TUM School of Engineering and Design
Technische Universität München

Parametric Projection-based Model Order Reduction for Crash Simulations

Mathias Lesjak

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 04.03.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 11.07.2024 angenommen.

Zusammenfassung

Crashsimulationen sind heutzutage ein unerlässliches Hilfsmittel für Ingenieure. Simulationen sparen teure Hardware in frühen Entwicklungsphasen und sie erhöhen das Vertrauen in das Crashdesign. Immer schnellere Produktzyklen und striktere Crashanforderungen erfordern auch schnellere Auslegungszyklen. Zudem erfordern Robustheitsstudien, Unsicherheitsanalysen und auch Optimierungsverfahren viele Simulationsauswertungen. Dabei spielen Robustheitsanalysen eine besonders wichtige Rolle, da sie das Vertrauen in das virtuelle Modell erhöhen und somit die Wahrscheinlichkeit eines Versagens von Crashversuchen in späten Entwicklungsphasen aufgrund von Unsicherheiten in Hardware und Testdurchführung reduzieren. Das Ziel von Modellreduktion ist das Bereitstellen von schnellen Modellen, um die zuvor genannten Methoden benutzen zu können. Modellreduktion lernt aus bereits generierten Daten, um schnelle Vorhersagen treffen zu können. Diese Arbeit beschäftigt sich mit der projektionsbasierten Modellreduktion, die datenbasiert ist, jedoch mit modifizierten physikalischen Gleichungen arbeitet.
Robustheitsstudien und Optimierung variieren die Parameter des Crashmodells, weshalb das reduzierte Modell auch parametrisch sein muss. Aktuell werden projektionsbasierte reduzierte Modelle für stark nichtlineare Probleme jedoch meist ohne Parametervariationen angewendet. Aufgrund dessen ist das Ziel dieser Arbeit parametrische Modellreduktionsmethoden für Crashprobleme zu entwickeln oder anzuwenden.
Diese Arbeit untersucht die Bandbreite an vorhandenen parametrischen projektionsbasierten Modellreduktionsmethoden, erweitert einige, und präsentiert ein funktionierendes reduziertes Modell basierend auf nichtlinearen Mannigfaltigkeiten. Beginnend mit der globalen linearen Methode, arbeiten wir uns zu lokalen linearen Methoden vor. Die Methode mit lokalen reduzierten Basen ist mit der energy-conserving sampling and weighting (ECSW) Hyperreduktionsmethode kompatibel. Deshalb erfüllt diese Methode die Voraussetzungen, um als Modellreduktionsmethode bei explizit zeitintegrierten, nichtlinearen Problemen eingesetzt zu werden. Die lokalen Basen werden durch ein Clusteringverfahren bestimmt. Wir stellen eine andere Clusteringmetrik im Crashkontext vor, welche die Charakteristiken von Modellen mit großen Deformationen berücksichtigt. Schließlich werden reduzierte Modelle basierend auf nichtlinearen Mannigfaltigkeiten vorgestellt. Sie versprechen reduzierte Modelle mit niedrigen latenten Dimensionen, und damit verbunden, kleineren reduzierten Netzen und größeren Rechenzeitverkürzungen.
Wir zeigen, dass globale lineare reduzierte Modelle im Grunde nicht geeignet sind für Crashanwendungen. Eine lineare Dimensionsreduktion findet in Daten mit großer Variation keine niedrigdimensionale Struktur. Die hohe Dimension der reduzierten Basis führt zu großen reduzierten Modellen mit verschwindender Rechenzeitverkürzung. Lokale reduzierte Modelle schaffen Abhilfe, indem sie die Lösung in lokale Bereiche, auch Cluster genannt, unterteilen. In den Clustern wird linear eine reduzierte Basis berechnet. Diese Basis ist niedrigdimensional, wodurch kleine reduzierte Modelle mit kleinen reduzierten Netzen in jedem Cluster möglich werden. Es ergeben sich Rechenzeitreduktionen von bis zu 32%. Diese Reduktion ergibt sich trotz der Auswertung von Elementen, welche mit Null gewichtet werden. Wir nutzen die parametrischen reduzierten Modelle außerdem, um Randbedingungen zu ersetzen, und beschleunigen so Crashmodelle, bei welchen ein Ersetzen durch fixe Randbedingungen scheitern würde. Abschließend zeigen wir ein reduziertes Modell basierend auf einem Autoencoder, mit einer reduzierten Netzgröße von 15% vom Gesamtnetz.
Zusammenfassend zeigen wir, dass reduzierte Modelle basierend auf lokalen Basen schnell und genau sind und als Ersatzmodelle für Randbedingungen eingesetzt werden können. Damit stellen wir erfolgreich parametrische, projektionsbasierte reduzierte Modelle für Crashmodelle vor. Außerdem stellen wir weitere Mechanismen zur Rechenzeitverkürzung vor, die weitere Forschung in diese Richtung motivieren.

Abstract

Today, crash simulations are an indispensable design tool for engineers. They save expensive hardware in the early development phase of a car and increase confidence in a crash design. Ever faster product cycles and stricter safety requirements also require a multitude of design cycles. In addition, optimization, uncertainty quantification, and robustness studies are multi-query methods that require numerous simulation evaluations. Robustness studies are particularly important as they increase confidence in the model and prevent failing crash tests at a late development phase due to hardware and test conduction uncertainties. Model Order Reduction (MOR) aims to provide fast models to enable these multi-query methods. MOR learns from already existing data and accelerates future predictions. Projection-based MOR restricts the solution to lie in a low-dimensional subspace and transforms the governing equations to predict the evolution in latent space. A data-driven dimensionality method determines the subspace.

Multi-query methods apply parameter variations to the model, which is why parametric Reduced-Order Models (ROMs) are required. So far, projection-based MOR was mainly limited to reproductive examples in highly nonlinear crash and impact problems. Therefore, this work aims to choose or develop MOR methods for parametric problems in crash.

This work evaluates the bandwidth of available methods for parametric projection-based MOR, extends specific methods, and presents a proof of concept (PoC) for MOR on nonlinear manifolds. Beginning with a global linear method, we move on to local linear approaches. The local reduced-order bases (lROB) method is compatible with energy-conserving sampling and weighting (ECSW) hyper-reduction. Thus, ECSW fulfills the prerequisites for MOR applied to explicitly solved nonlinear problems. lROB is extended by a clustering metric considering the characteristics of models exhibiting large deformations. Finally, projection-based MOR on nonlinear manifolds is presented for crash applications. Nonlinear manifolds promise even smaller latent dimensions and, associated with this, smaller reduced meshes and more significant speedups.

This thesis shows that global linear methods are in most cases unsuitable for MOR in crash. Parameter variations partially lead to significant variations in the output. A linear dimensional reduction cannot identify a low-rank structure in the data due to the large variance. Since hyper-reduction and transformations of the ROM scale with the reduced dimension, ROMs with negligible speedups result. ROMs based on lROB eliminate the large reduced dimensions by dividing the solution into subregions. Each subregion, also named cluster, is then assigned a lROB. The lROBs are low-dimensional, so hyper-reduction provides a small reduced mesh. We present computational time savings of 32%, even though elements weighted with zero must be evaluated due to their history dependency. We further utilize lROB ROMs as boundary condition replacement and accelerate models where a fixed boundary condition replacement fails. Finally, the PoC of a nonlinear manifold ROM based on an autoencoder motivates future research in this direction due to the hyper-reduction compatibility and a mesh size reduction of 85%.

To summarize, this work shows that ROMs based on lROB deliver fast and accurate ROMs that can be used as surrogates for boundary conditions. Therefore, we successfully introduce parametric projection-based ROMs for crash and impact problems. We outline further speedup potential, which promotes future research in this direction.

Acknowledgments

Many environmental factors influence a PhD thesis. Not only technical aspects but also social ones are crucial to successfully mastering a PhD journey. Even if this is already known during the work, it becomes more present towards the end, which is why I would like to thank all actors who directly or indirectly influenced the success of this work in the following.

First of all, I extend my deepest gratitude to my doctoral supervisor Prof. Dr.-Ing. habil. Fabian Duddeck for his scientific advice and the technical discussions. I would particularly like to thank him for the trust he has placed in me, for the freedom to choose my research focus, and for his support during difficult phases. His critical feedback has constantly improved the quality of my research and shaped my academic growth. Furthermore, I am grateful for the regular exchange appointments with the whole group, which fostered a sense of community among us.

I am also thankful to my BMW supervisor Dr. Michael Pfaffinger, for providing me with the opportunity to undertake this research position. He always found the right mix of constructive criticism and understanding, which has been immensely motivating. I extend my appreciation for his support within BMW. Thanks to his network and help, all research obstacles were quickly eliminated. Furthermore, I would like to thank my boss, Frank Bauer, for fostering an environment conducive to rigorous scientific research within our group.

I would like to thank Christopher Bach, who consistently supported me throughout my PhD. The regular exchanges provided invaluable opportunities for discussion and brainstorming about model order reduction. Even shortly before submission, discussions about the research results motivated and stimulated further thought.

My special thanks go to Ulrich Huber, with whom I had countless debug sessions and professional exchanges on computer science topics. His expertise, patience, and willingness to share insights have been instrumental in overcoming challenges and advancing the development of this thesis.

I am grateful for the great environment where I could conduct this research and work. Many friendships have arisen from shared PhD activities. I would especially like to mention Giada Colella, Zeyu Lian, Reza Barzanooni, and Marius Rees. The evenings and activities together were a wonderful distraction from research.

Furthermore, I owe a debt of gratitude to my best friends. Great time together and enriching conversations also helped me get through difficult research phases. Thank you Max, Conni, Kevin, Robin, and Alex.

From the bottom of my heart, I want to thank my family for their love, support, encouragement, and understanding. Without your safety, this work would never have been completed.

Contents

1 Introduction

1.1 Motivation

The virtual design of vehicles is gaining importance as product cycles are becoming ever faster, and in the context of passive safety, more complex tests have to be performed. Initially, the motivation of this thesis stems from Verification and Validation (V&V), nowadays often complemented by Uncertainty Quantification (UQ), which is a process to get confidence in the simulation models to prevent costs associated with not meeting the required product properties or avoiding the complete failure of the final product. We are concerned with the so-called model V&V in computational mechanics, where the final product is a computational model. Advanced validation metrics are applied to validate the model, which requires numerous simulation evaluations. Whenever many predictions of the model are necessary, the costs associated with constructing a reduced order model (ROM) may be negligible compared to the total computation costs. The concept of using already generated data/knowledge to accelerate future evaluations is the main idea of Model Order Reduction (MOR).
However, multi-query analyses also appear in other disciplines, such as optimization, robustness studies, or sensitivity analyses. In the context of passive safety, especially robustness studies are of interest. They investigate the behavior of the current design around a point in parameter space. This knowledge is essential as a single prediction of a virtual model only tells the behavior at the exact parameters. However, both models, virtual and hardware, involve uncertainties. The virtual model includes modeling assumptions, and the physical crash test involves uncertainties in the test conduction (e.g., impact velocity, impact location, etc.) as the vehicle itself is not yet from series production, which is why production-related deviations can occur. The behavior of the crash design around the parameter point is investigated using robustness studies to avoid small deviations leading to a failure of the crash test in later development stages. The required amount of simulations could also make MOR a helpful tool or even enable robustness studies for larger models, which would otherwise require too much computational resources.
In the context of optimization, MOR provides fast and cheap-to-solve models. As crash simulations are usually solved explicitly in time, no gradient information is available, and typically, population-based optimization algorithms are applied. They require many simulation evaluations. Therefore, MOR enables optimization or accelerates it.
A different use case for MOR in optimization is the partial reduction of simulation models. Often, only one part of a vehicle is optimized. The rest of the vehicle remains unchanged. It is computationally infeasible to optimize the single part based on the full vehicle model. One way to overcome this limitation is to cut the part free and apply displacements or equivalent loads. However, this approach lacks feedback on the overall vehicle. A partial reduction could combine computational speedup while maintaining feedback. In addition, parts with incompatible or confidential formulations could be excluded from MOR.
This work is mainly concerned with projection-based MOR (pMOR). pMOR is successfully applied to crash problems in reproductive situations [1]. However, these ROMs quickly fail when varying the parameters for strongly nonlinear crash problems. All of the previously stated methods share the need for parametric models. Simply including more training data and creating a global ROM does not yield an accurate and low-dimensional ROM [2]. Therefore, different approaches are investigated to construct parametric pROMs for crash problems.

1.2 Research Objectives

Initially, this work was intended as a continuation of Bach's thesis [1], wherein projection-based Model Order Reduction (pMOR) based on Proper Orthogonal Decomposition (POD) with Energy Conserving Sampling and Weighting (ECSW) hyper-reduction is successfully applied to exemplary crash problems. However, parameter variations have not been analyzed in depth, which is necessary for most applications of reduced-order modeling. Optimization and robustness studies are often applied in crash design and require numerous simulation evaluations. Without parametric reduced-order models (ROMs), these methods are unsuitable or require large computational resources and time. Hence, this research is concerned with creating parametric pROMs for highly nonlinear crash and impact models. The following research objectives are addressed to achieve the overall aim of parametric pMOR for crash:

- To develop fast pROMs, residual minimization is analyzed in detail to provide the necessary foundations for further investigation. The required mathematical construct is adapted to crash models, and the best methodology is identified among different strategies. Crash models usually consist of many shell elements that exhibit rotations and displacements. It is to determine whether a ROM that couples these degrees of freedom (DoF) or a separate ROM for each DoF is preferred. Building on the best ROM method, we conduct a study determining the preferred treatment of different degrees of freedom for parametric problems.

- pROMs with a single global linear reduced basis applied to a parametric crash box problem show a slow error decay. POD fails to identify a low-dimensional structure in the data. The high variance in the solution data results from the large nonlinearity of the model. ROMs with large reduced dimensions are the consequence, and effective hyper-reduction is barely possible. Therefore, local reduced-order bases (lROB) are identified as a suitable method for parametric pROMs. lROB is adapted to crash problems, and two clustering metrics are analyzed. Hyper-reduction is combined with lROB, and different strategies are identified. Using a parameter study, the influential hyper-parameters are identified to derive recommendations.

- Finally, nonlinear dimensionality reduction using an autoencoder (AE) is investigated. Nonlinear methods promise to resolve nonlinear correlations in the data and ensure a low-dimensional mapping. The low dimension enables effective hyper-reduction and, thereby, computational speedup. We adapt the nonlinear pMOR framework to crash problems. We investigate different AE variants to identify suitable architectures and hyper-parameters. ECSW is combined with an AE ROM to assess the potential speedup. In addition, a hyper-parameter study shows correlations and proposes design recommendations.

To summarize, we investigate currently available dimensionality reduction methods and divide them into three categories: global linear, locally linear, and global nonlinear. First, the mathematical details of pMOR are discussed to formulate the ROMs. Afterwards, for each dimensionality reduction category, the research objective is to transfer and extend the methodology to the problem-specific needs and assess the suitability of each method.

1.3 Preliminaries and Related Work

There is a need for fast simulation models in many scientific disciplines. Therefore, MOR is applied in various problems, such as heat transfer, fluid mechanics, chemistry, and solid dynamics [3]. Surrogate modeling can be categorized into three categories [4], as seen in Fig. 1.1. The models are hierarchical, data-fit, and projection-based ROMs. Further, the models can be distinguished by the required source code access. Non-intrusive methods do not require source code access, but intrusive methods do. pROMs are completely intrusive, whereas data-fit models are usually non-intrusive. However, hybrid methods exist, which again require access to the source code of the underlying solver.

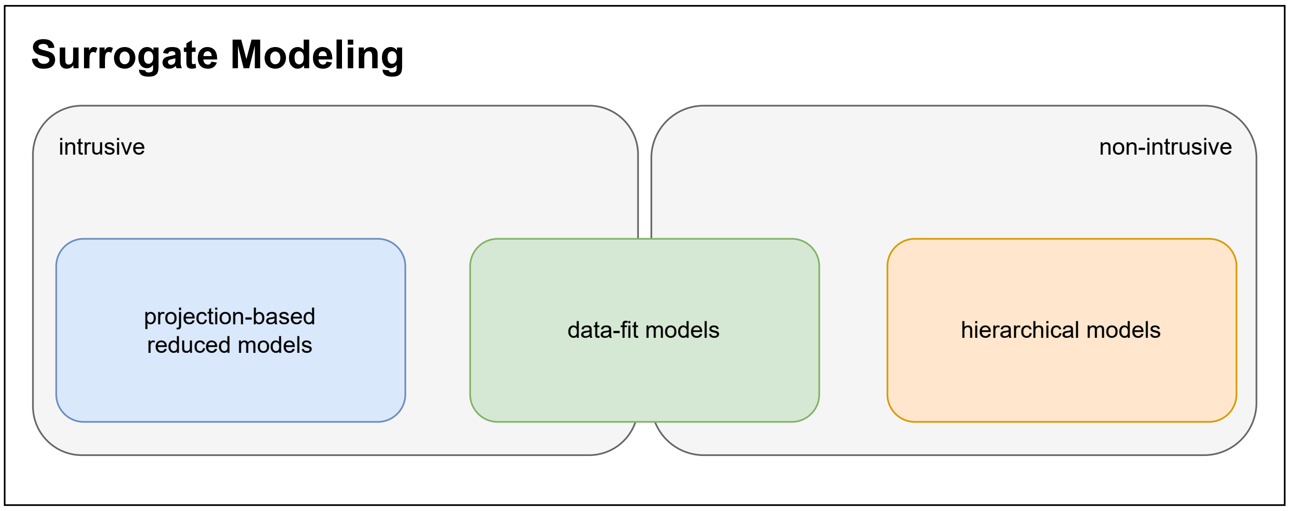


Figure 1.1 Division of surrogate modeling approaches.

Hierarchical models are physics-based models simplified using physical assumptions, coarser grids, alternative basis expansions, or looser residual tolerances. They are usually derived from higher-fidelity models and simplified using the mentioned assumptions. Fehr et al. [5] modeled a go kart frame by dividing it into nonlinearly deforming and linearly deforming parts. The linear parts are simplified using linear MOR, which saves computation time. Additional work investigates the identification of nonlinear and linear regions by considering different measures [6].
Data-fit models are usually purely data-driven models which learn a mapping from inputs to outputs. They utilize regression or interpolation techniques and need to be made aware of the physics. Popular methods are based on polynomials, radial bases functions [7], polynomial chaos expansion [8], Gaussian processes [2, 9–12], and Neural Networks (NN) [13–20].
A different strategy is to learn the dynamical system, which describes the underlying problem and not only the mapping from some input parameters to some outputs of interest. The sparse identification of nonlinear dynamics (SINDy) method was introduced by Brunton et al. [21] to identify the governing equations from known data. It assumes that the dynamics are sparse regarding function bases. The optimization algorithm promotes sparsity by using an L^1 regularization. The method is applied to low-dimensional dynamical systems and successfully identifies the weighting coefficients of the nonlinear contributions. A similar approach in pMOR exists, called Operator Inference [22]. The assumption of sparsity is not justified anymore. Therefore, the coefficients of the nonlinear terms are determined using a least squares fit. However, SINDy and Operator Inference provide the same nonlinearity in the dictionary of functions as the sought-after system exhibits. Operator Inference is designed for polynomial nonlinearities. Nevertheless, if the system is not of polynomial type, in certain cases it can be transformed (lifted) to a system with polynomial nonlinearity [23]. Similar approaches exist for SINDy. The appearance of the governing equations heavily depends on the chosen coordinates. To obtain a simple structure, [24] combines a NN with SINDy and trains them simultaneously. The NN identifies a suitable set of coordinates such that the SINDy model can identify the set of governing equations.
SINDy and Operator Inference are purely data-driven methods and do not impose any structure on the model. However, physical systems usually exhibit a structure. Now, if the data does not precisely fit the proposed structure of the models, the model may only fit the data and does not learn physical equations. This is shown by the fact that the identified systems suffer from instabilities. Additional physical information is used to stabilize the systems [25, 26]. Using physical constraints to steer data-driven methods towards stable dynamics and physical solutions is becoming increasingly popular and these methods are grouped under the term physics-informed machine learning [27]. By teaching NNs to fulfill conservation laws, they can even be used to solve partial differential equations (PDEs) [28]. Recent developments in NNs make it even possible to learn continuous operators instead of functions. The DeepONet [29], e.g. learns the

solution operator to simple dynamical systems.
Another dimensionality reduction method is the dynamic mode decomposition (DMD) which computes spatio-temporal modes [30–32] purely data-based. An extension of the DMD, which approximates the leading eigenvalues, eigenvectors, and modes of the Koopman operator, is proposed by Williams et al. [33]. While applications of DMD mainly focus on fluid mechanics problems, SINDy was applied to systems in solid mechanics, even with hysteresis [34–36]. An overview of system identification methods in structural engineering can be found in [37, 38]. Operator Inference was applied to mechanical systems with a particular structure. Kim et al. [39] showed that 3rd order polynomials can approximate the nonlinear force term for hyperelastic materials. Similar approaches can be found in [40] and even with hyper-reduction [41].

The type of MOR we are concerned with in this thesis is projection-based MOR (pMOR). pMOR can be divided into an offline and an online phase, as seen in Fig. 1.2. The offline phase includes the steps related to the creation of the ROM. The online phase is the fast evaluation of the created ROM. In the offline phase, the first step is creating the training data if a data-driven ROM method is used. There are also simulation-free reduction methods, which do not require training data. In the data-driven case, a dimensionality reduction is applied to the data to find a low-dimensional representation. This mapping is subsequently used to formulate the pROM, the last block of the offline phase in Fig. 1.2. As the dimensional reduction is usually not exact, a residual is formed when the mapping is inserted in the governing equations. Different strategies exist to treat the residual, which leads to different pROMs. An additional reduction step, called hyper-reduction must be applied for highly nonlinear equations. A precomputation of the reduced nonlinear term is only possible for certain types of nonlinearities.
Simulation-free methods do not need expensive model evaluations prior to the ROM construction. One

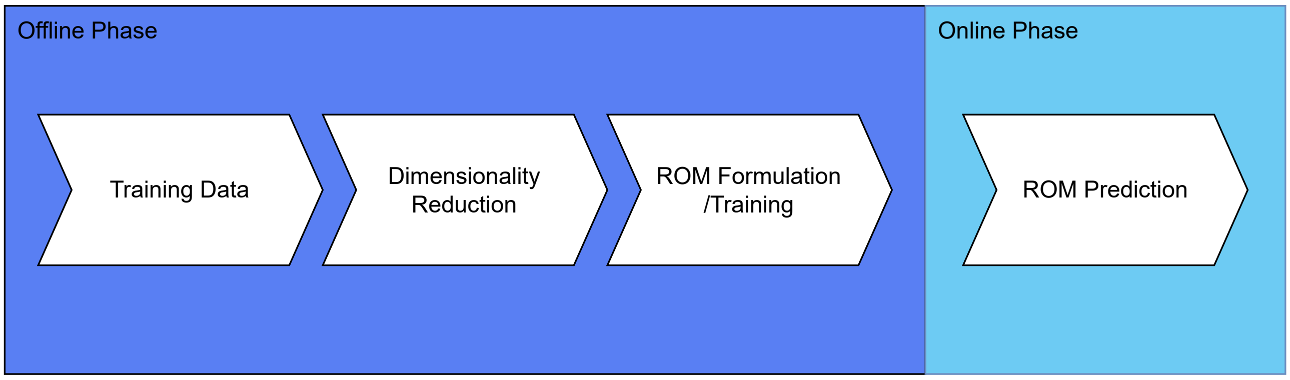


Figure 1.2 Generic MOR workflow.

of the earliest model reduction approaches for linear systems is to approximate the solution of the system with a reduced number of vibration modes. Several extensions such as static and modal derivatives exist [42–50] for nonlinear systems. An overview of modal derivatives, which express the sensitivities of a system, can be found in [51, 52]. An additional approach to reduce the complexity of large systems is substructuring. Component Mode Synthesis (CMS) [53] or modal coupling technique [54] divide a large model into substructures, which are treated individually. To solve the complete system, the substructures are coupled using interfaces whereby equilibrium is imposed. Further substructuring techniques widely used in linear structural analysis are the Craig-Bampton (CB) method [55] or the Rubin method [56].
In contrast to the simulation-free approaches, data-driven approaches use generated solution data to create a basis for expressing the new solution. This first step in the pROM creation is always the dimensional reduction and, associated with that, the Reduced-order Basis (ROB) creation. Several methods exist to compute the ROB. The most commonly used method is the Proper Orthogonal Decomposition (POD) [57]. POD is a linear method, mathematically equivalent to the Singular Value Decomposition (SVD), Principal Component Analysis (PCA) [58], or the Karhunen-Loève transform. However, algorithm differences exist, and the terms are used in different contexts. A rigorous discussion about comparing the three algorithms can be found in [59]. Also, a historical discussion of the SVD can be found in [60]. To the author's knowl-

edge, Sirovich [61] first used POD to identify coherent structures in turbulent fluid flows. Dimensional reduction relies on the fact that solutions of high-dimensional systems usually live in a lower-dimensional subspace [62]. Once the solution manifold is approximated, different methods exist to evaluate the approximation quality [63–65]. The truncated POD is the best possible rank k approximation of the training data regarding the Frobenius norm [66], where k is the dimension of the low-dimensional subspace. However, this does not necessarily mean it is the best basis to represent the dynamics of the problem, as shown by Moore, who introduced the Balanced Truncation (BT) method for linear systems [67]. Coordinates with low variance would be truncated by POD even if the system would be sensitive to them. Rowley proposed the balanced POD [68], which uses solution snapshots and information about the system's adjoint. This approach enabled the application of BT to high-dimensional systems, which would otherwise not be possible due to the computationally impractical evaluation of the Gramians. Extensions to nonlinear systems exist [69–73]. Phallipou et al. [74] showed that selecting POD modes according to the ordering of the eigenvalues can be suboptimal in explicit dynamics (crash applications).

In addition to the variance-based dimensionality reduction methods, extensions of them exist, or other/nonlinear manifold learning techniques are applied. Chelidze [75] proposed the smooth orthogonal decomposition which augments the classical POD with an additional requirement for smoothness to remove noisy measurements or [76] proposed a weighted POD, which assigns weights to certain snapshots. A snapshot of a specified variable is defined as the variable evaluated at a specified time, location and parameter combination. Further variants are the robust PCA [77] or the sparse PCA [78].
For certain problems such as convection-dominated problems, [79] or parametric nonlinear problems [2], a global linear dimensional reduction is generally not able to accurately describe the data in a low-dimensional space due to the slowly decaying Kolmogorov n-width [80]. Global means that all solution snapshots are assembled into one data matrix, also called snapshot matrix, and passed to the POD. Two methods based on POD exist to overcome the n-width limitation. The first approach applicable to linear parametric systems is manifold interpolation [81–84]. The Grassmanian [85–88] parameterizes all r-dimensional linear subspaces of an n-dimensional vector space. Utilizing the manifold concept allows to span a tangent space to the manifold, which is a space where interpolation can be performed as usual. After interpolation at the desired point, the point is projected back onto the manifold. This satisfies the orthogonality constraints that would not be met if the basis vectors were directly interpolated. However, the application to nonlinear systems that require hyper-reduction needs to be clarified. Therefore, a second approach that is directly compatible is considered.
The local ROB (lROB) approach [89] partitions the solution space into subregions, and for each subregion, a local basis is computed using POD. The clustering algorithm is the k-means clustering [90], which clusters solution snapshots based on their squared L^2 norm. Another aspect of the method is the fast identification of the closest subregion. Each subregion, called a cluster, has a cluster center (centroid). Calculating the distance in full space scales with the dimension of the Full Order Model (FOM) and thus prevents a speedup. Amsallem et al. [89], therefore, derived a sign function, which scales with the dimension of the ROM and indicates the need to change the cluster by a flipping sign. The lROB is compatible with Petrov-Galerkin ROMs and hyper-reduction [91, 92].
Other methods are designed to resolve more sophisticated manifold structures primarily in a global sense in contrast to the piecewise linear approximation. Popular nonlinear dimensionality reduction methods are the kernel PCA (kPCA) [93], locally linear embeddings (LLE) [94], isomaps [95], diffusion maps [96], and diffusion nets [97]. Different methods are compared in [98]. A popular NN-based method is the Autoencoder (AE). The first appearance of AE is hard to determine [99], however, [100] describes a novel feedforward NN architecture including the mathematical formalism. In MOR, AE appears in non-intrusive MOR [101–106] as well as in pMOR [79, 107]. AE-based pMOR also appears in the computer graphics community [108, 109], which could be due to the missing reference simulation. Since the goal is only a realistic-looking deformation and does not necessarily have to be physical, the requirements here are lower, and the AE has a clear advantage over a POD-based modeling.
A different nonlinear approach is quadratic manifolds (QM) [110, 111]. The linear approximation obtained

by POD is extended by a term that depends quadratically on the reduced state. QM is compatible with hyper-reduction [112], used for Operator Inference [113] and utilized to mitigate the Kolmogorov barrier in nonlinear pMOR [114].

Once the proper basis is found, the pROM has to be formulated by inserting the mapping in the governing equations. Thereby, the trial basis is defined, or simply put, the admissible displacements are defined. The next step is to define how the arising residual is treated. Suppose the generated residual is enforced to be orthogonal to a subspace spanned by the test basis, and the test and trial bases are equal. In that case, we obtain a Bubnov-Galerkin scheme, often labeled Galerkin scheme in literature. This differs from a Petrov-Galerkin scheme, where the test and trial bases differ. How different ROM formulations can be interpreted regarding their corresponding residual minimization problems is discussed in [115]. Also, a detailed discussion about time-continuous optimality and time-discrete optimality and which time integration methods destroy the optimality criterion can be found there.
Projection-based MOR is especially successful (in terms of speedup) in the case of linear ODEs or special cases in which the reduced operators can be built a priori. The construction of the reduced operators scales with the size of the high-dimensional model (HDM) but needs to be done only once if they are pre-computable. Typical examples are linear time-invariant systems (LTI) in control and structural vibration. If the high-dimensional operators must be evaluated at each time step, additional reduction methods termed hyper-reduction must be utilized. Especially in crash applications, the semi-discretized PDEs are a highly nonlinear system of ODEs that do not fulfill the mentioned operator properties.
Hyper-reduction methods can be categorized into nodal vector approximation methods, which try to interpolate the finite element force vector, and integral approximation methods, considering the finite element force vector as domain integral. The nodal vector approximation methods are based on the work of Everson and Sirovich [116] for gappy data. All methods try to approximate the original vector using only a small number of entries of the snapshots. How to choose the indices of the entries is described by various methods, such as the Empirical Interpolation Method (EIM) [117] and its discrete counterpart the Discrete Empirical Interpolation Method (DEIM) [118], a DEIM version that further reduces the computational time as it works on the unassembled quantities and does not need to consider adjacent quantities [119, 120], the Best Points Interpolation Method (BPIM) [121], and the discrete BPIM [122]. Further approaches exist, such as polynomial tensors [123] or the Gauss-Newton with approximated tensors (GNAT) method. However, polynomial tensors are only applicable to certain problems. The collocation-based methods do not construct the low-dimensional approximation directly but evaluate the balance equations only at the collocation points [124].
The second category relates to integral approximation approaches; they can be further divided into methods that interpolate the integrand and cubature methods, which interpolate the whole integral as the weighted sum of the integrand evaluated at sampling points. Interpolation of the integrand follows the same principles as the vector approximation methods and is presented in [125–128]. The methods mentioned above sometimes lack numerical stability for second-order nonlinear differential equations as they are not structure-preserving. Therefore, cubature-based hyper-reduction methods have been developed. Optimized cubature methods work on the unassembled quantities and effectively speedup computation due to the reduced mesh size. The first method of this kind was proposed by An et al. [129] in the field of computer graphics. Later, the method was adopted by the computational mechanics community by [130, 131] and termed the Energy-Conserving Sampling and Weighting method (ECSW) [132–134]. Hernández et al. [135] introduced the empirical cubature method related to ECSW but operates on the integration points instead of working on the element level. A comparison of established hyper-reduction methods can be found in [136, 137]. In the context of explicit dynamics, ECSW is the most popular technique due to structure preservation and associated with its stability properties [131]. Structure preservation of MOR for mechanical systems was further investigated by [138]. Similarly to the non-intrusive methods, the stability of pROMs [139] can be improved by a postprocessing step [140]. In addition, the stable time step of explicit time integration is usually enlarged by the projection step [130, 141].
In addition, we want to mention iterative approaches such as a priori model reduction [142–144] and the

a priori hyper-reduction (APHR) [145] which both incrementally update the ROM if more data becomes available. Finally, it is to say that models in crash usually contain internal variables that require special attention [124], and contact algorithms consume a large portion of the total simulation time. The reduction of contact forces still needs to be explored [146, 147].

We end this section with an economic perspective on MOR. This consideration can be applied to all

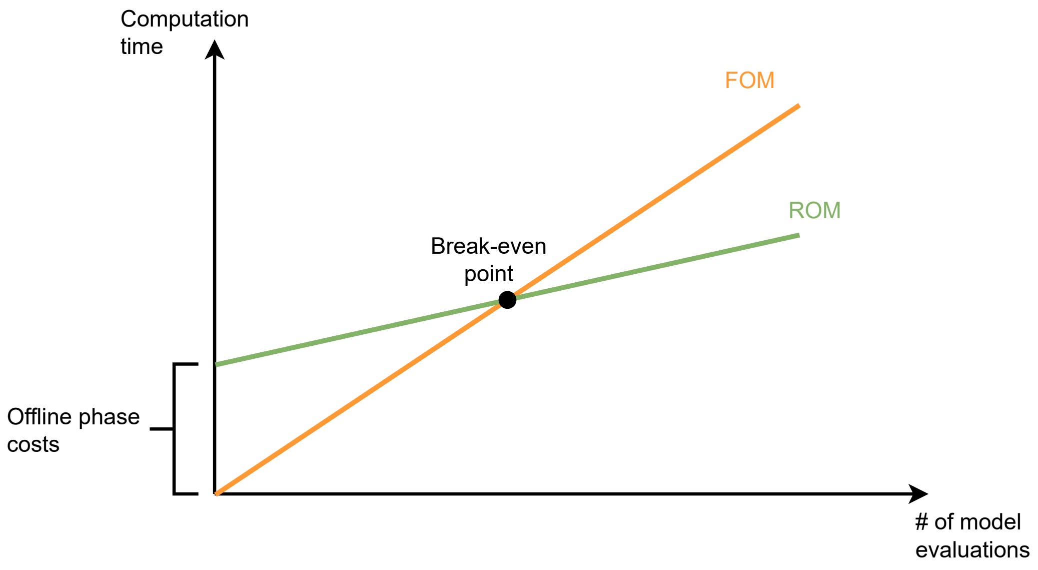


Figure 1.3 Economic perspective to model order reduction.

previously discussed MOR methods. Fig. 1.3 compares the total computation time for an increasing number of model evaluations for a FOM (in orange) and a ROM (in green). Compared to the FOM, the line for the ROM does not start at zero. Instead, the offline phase cost appears, which is the initial inevitable "investment". At the break-even point, the total computation time of the FOM and the ROM is equal, and for further evaluations, the ROM is faster in total. Non-intrusive methods usually have higher offline phase costs, as a machine learning (ML) model must be trained. However, querying the model is very fast; thus, it is suitable for methods requiring many evaluations, such as population-based optimization algorithms. In contrast, pMOR has a cheaper offline phase consisting of dimensionality reduction and a clustering algorithm in the case of IROB. For nonlinear systems, the optimization associated with hyper-reduction must be added. Since the pROM is still a system of ordinary differential equations (ODEs), which is solved using a time integration scheme, the reduction in computation time is not as immense as in the non-intrusive case. Depending on the requirements of the multi-query method regarding computation time of the model and solution resolution (entire field or just specific quantities), the suitable method can be selected. This thesis concerns pMOR as it is physics-based and provides the same resolution as the underlying finite element method (FEM) model. Also, the possibility of replacing certain parts with a reduced model and optimizing other parts was a challenging question where pMOR is a promising approach.

1.4 Contributions of this Work

The contributions of this work are listed in the following:

- We rigorously derive projection-based ROMs with a particular focus on residual minimization for nonlinear second-order systems. Different ROMs are formulated which optimize the arising residual regarding different semi-norms. The ROMs are applied to two exemplary crash problems and the best ROM is identified for the following chapters.

- We apply pMOR in a parametric setting and study the effect on the dimensionality reduction as well as the ROM construction. Further, the influence of the different variable types, namely displacements and rotations, is studied and a recommended ROM construction regarding the degrees of freedom treatment is proposed.
- The identified barriers of reduced order modeling for parametric highly nonlinear systems are tackled by introducing local reduced-order bases for second-order systems with explicit time integration. The chosen method best satisfies the requirements from explicit FEM, e.g., compatibility with hyper-reduction. Due to the three-point stencil in explicit time integration, we propose a velocity transformation to transform the velocity to the new local basis in case of a basis change.
- lROB based on spherical clustering (slROB) is introduced due to observing system states with a nearly monotonically increasing L^2 norm. As regular lROB clustering is based on the squared L^2 norm, solutions are clustered along the deformation, which is similar to the time axis. Inspired by the clustering of documents of varying length but similar content, spherical clustering is adjusted to solid mechanics problems.
- lROB, as well as slROB, are combined with ECSW hyper-reduction. We propose a global-local hyper-reduction strategy. This strategy uses a global mesh with varying weights and allows changing clusters and weights without reactivating elements with missing history.
- Motivated by the need to optimize single components in full-vehicle models, we exemplify the possibility of reducing single parts in a full model. Our proof of concept (PoC) allows even stronger assumptions as we reduce only certain sections of a crash box model, which equal parts of the model. Further, the compatibility with ECSW hyper-reduction is demonstrated.
- We formulate a ROM based on AE, which is a NN-based nonlinear dimensionality reduction. We train the AE regarding different error definitions and evaluate the accuracy compared to the linear ROMs. To the author's knowledge, this is the first application of an AE in pMOR for crash.

1.5 Outline

The thesis is structured in the following way. In Chapter 2, we present the governing equations and formulate a POD-based pROM. The different mechanisms to achieve speedup are discussed and the following chapters are motivated. We discuss POD and describe how to utilize SVD to create the ROB. Residual minimization is introduced in the context of second-order nonlinear equations, different pROMs are formulated, and results are shown for a pipe whip and a crash box example. Parameter variations are introduced, and the resulting Kolmogorov n-width barrier is illustrated. The behavior of displacements and rotations is studied, and a recommended treatment is deduced. Lastly, the concept of ECSW hyper-reduction is introduced. In Chapter 3, the idea of lROB is explained, and the method is applied to the parametric problem defined in the previous chapter. The crash box problem is analyzed, and features of crash problems are highlighted. lROB is modified using spherical k-means clustering, and slROB is presented. ECSW is combined with lROB and slROB. The methods are demonstrated using the crash box example. In Chapter 4, we conduct a PoC of a partial reduction. The crash box is divided into four rings, leaving one ring unreduced. The geometry of the unreduced ring is modified, and the accuracy of the surrounding ROM is measured. In Chapter 5, pROM based on nonlinear dimensionality reduction is introduced, NN and AE are explained, and successful examples are presented. Finally, in Chapter 6, the findings are summarized, and an outlook for future research is given.

2 Residual Minimization

Chapter 2 introduces residual minimization in the context of second-order ODEs resulting from finite element approximations in solid mechanics. Section 2.1 summarizes the governing equations beginning from the strong form arising from continuum mechanics, reaching the finite element approximation and, finally, the time integration. The resulting equations from Section 2.1 are the starting point for the pMOR framework, which is why the main mechanisms for computational speedup are presented next in Section 2.2. Finally, pMOR is introduced in Section 2.3. pMOR restricts the solution to lie in a lower-dimensional subspace. If the conservation equation can no longer be fulfilled exactly, a residual is formed. We propose different residual minimization strategies in the remainder of Section 2.3, which yield different ROMs. The ROMs differ in computational complexity, e.g., the inversion of the reduced mass matrix of the Autoencoder ROM in Chapter 5 could be dispensed with. In addition, mathematical relations are derived and discussed, and three different pROMs are summarized. Section 2.4 introduces error measures to assess the ROMs and applies the identified ROMs of the previous section to two exemplary crash problems. We choose the best ROM and use it for all other methods in this work.

2.1 Governing Equations

2.1.1 Strong Form

In this work, we consider the deformation of structures which can be modeled using continuum assumptions. The governing equations are the results of continuum mechanics and consist of conservation laws for mass, momentum, and energy. In crash applications, temperature is usually not considered and the energy balance is not solved. Also in this chapter we therefore limit the introduction to mass and momentum conservation. We present a summary of the equations relevant to this work. A detailed derivation can be found in [148].
Mass conservation reads:

$$\rho_0 = \rho(\mathbf{X}, t) J(\mathbf{X}, t), \tag{2.1}$$

where $\mathbf{X}$ is the material position, ρ is the density, ρ_0 is the material density, t is the time and $J = \det\mathbf{F}$ is the determinant of the deformation gradient $\mathbf{F}$. The deformation gradient is defined as:

$$\mathbf{F}(\mathbf{X}, t) = \frac{\mathrm{d}\mathbf{x}(\mathbf{X}, t)}{\mathrm{d}\mathbf{X}}, \tag{2.2}$$

where $\mathbf{x}(\mathbf{X}, t)$ is the current position. The linear momentum balance reads:

$$\rho \frac{D\mathbf{v}}{Dt} = \rho\dot{\mathbf{v}} = \nabla \cdot \boldsymbol{\sigma} + \rho\mathbf{b}, \tag{2.3}$$

where $\frac{D\cdot}{Dt}$ denotes the material derivative, $\mathbf{v}$ is the velocity field, $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\mathbf{b}$ are volume forces. The balance of angular momentum is satisfied by imposing symmetry to the Cauchy stress tensor:

$$\boldsymbol{\sigma}^T = \boldsymbol{\sigma}. \tag{2.4}$$

The solution $\mathbf{x}(\mathbf{X}, t)$ to these equations fulfills the governing equations for every point in space and time. Therefore, this formulation is called strong form.

2.1.2 Weak Form

To solve the previously defined set of partial differential equations (PDEs), the strong form is first transformed into a weak form that weakens the requirements for differentiability of the solution and dictates that the solution does not need to satisfy the equations in an "absolute" sense anymore but in some precisely defined sense. The weak form is obtained by multiplying the strong form with a test function $\delta\mathbf{v}$, integrating it over the current domain Ω and applying the Gauss-Green divergence theorem to shift one derivative from the solution onto the test function. The test function can be thought of as velocity, giving the terms units of power. The resulting equation, which is also known as the principle of virtual power [149], reads:

$$\underbrace{\int_\Omega \delta\mathbf{v}\cdot\rho\dot{\mathbf{v}}\mathrm{d}\Omega}_{\delta P^{\text{kin}}} + \underbrace{\int_\Omega \frac{\partial\delta\mathbf{v}}{\partial\mathbf{x}} : \boldsymbol{\sigma}\mathrm{d}\Omega}_{\delta P^{\text{int}}} \underbrace{- \int_\Omega \delta\mathbf{v}\cdot\rho\mathbf{b}\mathrm{d}\Omega - \sum_{j=1}^{n_{\text{SD}}}\int_{\Gamma_{t_j}} \delta v_j \bar{t}_j \mathrm{d}\Gamma}_{\delta P^{\text{ext}}} = 0, \tag{2.5}$$

with the kinetic virtual power δP^{kin}, the internal virtual power δP^{int} and the external virtual power δP^{ext}. The traction is $\bar{\mathbf{t}}$ and the traction boundaries are Γ_{t_j} with the number of spatial dimensions n_{SD}. The space of trial functions is defined as [148]:

$$v_i(\mathbf{X},t) \in \{v_i | v_i \in \mathcal{C}^0, v_i = \bar{v}_i \text{ on } \Gamma_{v_i}\}, \tag{2.6}$$

where $\bar{v}_i$ is the velocity boundary condition on the velocity boundary Γ_{v_i}. This space of kinematically admissible velocities is chosen to satisfy the velocity boundary conditions and to satisfy compatibility [148]. The space of test functions is chosen as:

$$\delta v_i(\mathbf{X}) \in \{\delta v_i | \delta v_i \in \mathcal{C}^0, \delta v_i = 0 \text{ on } \Gamma_{v_i}\}. \tag{2.7}$$

Both, test and trial function are infinite-dimensional, which is why the weak form and the strong form are equivalent. However, in computational frameworks, a finite dimensional function space is required. The transfer to a finite-dimensional space is described in the next subsection.

2.1.3 Finite Element Approximation

To be able to solve eq. (2.5) numerically, a finite-dimensional function space must be chosen. We introduce the finite element approximation by dividing the whole domain Ω into n_e subdomains Ω_e with:

$$\Omega = \bigcup_e^{n_e} \Omega_e. \tag{2.8}$$

The current position of the n_N nodes is approximated with shape functions N_I:

$$\mathbf{x}(\mathbf{X},t) = N_I(\mathbf{X})\mathbf{x}_I(t), \quad \text{for } I \in [1, n_N]. \tag{2.9}$$

Summation over repeated indices is implied. At node I, the coordinate vector is defined as:

$$\mathbf{x}_I(t) = \begin{bmatrix} x_{1I} \\ x_{2I} \\ \vdots \\ x_{n_{\text{SD}}I} \end{bmatrix} = x_{iI} \quad \text{for } I \in [1, n_N],\ i \in [1, n_{\text{SD}}]. \tag{2.10}$$

The velocity is the time derivative of eq. (2.9) and reads:

$$\mathbf{v}(\mathbf{X},t) = N_I(\mathbf{X})\mathbf{v}_I(t). \tag{2.11}$$

The test function is not a function of time and is approximated using the same shape functions as for the trial function:

$$\delta\mathbf{v}(\mathbf{X}) = N_I(\mathbf{X})\delta\mathbf{v}_I. \tag{2.12}$$

Inserting the finite element (FE) approximations into eq. (2.5) yields the set of discrete equations [148]:

$$\delta v_{iI}(\underbrace{\delta_{ij}\int_{\Omega}\rho N_I N_J \mathrm{d}\Omega}_{M_{ijIJ}}\,\dot v_{jJ} + \underbrace{\int_{\Omega}\frac{\partial N_I}{\partial x_j}\sigma_{ji}\mathrm{d}\Omega}_{f^{\text{int}}_{iI}} - \underbrace{\int_{\Omega}N_I\rho b_i\mathrm{d}\Omega - \int_{\Gamma_{t_i}}N_I\bar t_i\mathrm{d}\Gamma}_{f^{\text{ext}}_{iI}}) = 0 \quad \forall \delta v_{iI} \notin \Gamma_{v_i}. \tag{2.13}$$

As the scalar values δv_{iI} are arbitrary, the inner part of eq. (2.13) must be zero. As the focus of this thesis is MOR, we simplify the notation and write eq. (2.13) as:

$$\mathbf{M}(\mu)\ddot x + f_{int}(t, x, \dot x, \mu) = f_{ext}(t, x, \mu), \tag{2.14}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is the mass matrix, $f_{int} \in \mathbb{R}^n$ and $f_{ext} \in \mathbb{R}^n$ are the internal and external force vector, respectively, and $\mu \in \mathbb{R}^p$ is the parameter vector containing the p varied parameters. The number of DoF in the system is n reduced by the number of constraints n_{cstr}:

$$n = n_{\text{SD}} * n_N - n_{\text{cstr}}. \tag{2.15}$$

Typical car structures are thin-walled and described by shell formulations in the FEM models. Therefore, x contains translatory and rotatory DoF. Often the force terms are summarized and the equations are simply written as:

$$\mathbf{M}\ddot x + f = 0, \tag{2.16}$$

with:

$$f(t, x, \dot x, \mu) = f_{int} - f_{ext}. \tag{2.17}$$

2.1.4 Time Integration

A time integration scheme is applied to discretize eq. (2.14) in time. It is very common to use an explicit time integration scheme in crash applications due to the high dynamics and short simulation times in these scenarios. In commercial solvers, central difference scheme is often used to solve systems of second-order ODEs. The central difference scheme as depicted in Fig. 2.1, reads as follows [148, 150]:

$$\ddot x|_{t=t_n} = \ddot x^n \approx \frac{\dot x^{n+\frac{1}{2}} - \dot x^{n-\frac{1}{2}}}{\Delta t_x}; \tag{2.18a}$$

$$\dot x|_{t=t_{n+\frac{1}{2}}} = \dot x^{n+\frac{1}{2}} \approx \frac{x^{n+1} - x^n}{\Delta t_2} \tag{2.18b}$$

where

$$\Delta t_x = \frac{\Delta t_1 + \Delta t_2}{2}. \tag{2.19}$$

For a constant time step, the acceleration at time $t = t_n$ can be expressed as:

$$\ddot x^n \approx \frac{x^{n+1} - 2x^n + x^{n-1}}{\Delta t^2} \tag{2.20}$$

and consequently, the new displacement is obtained as:

$$x^{n+1} = 2x^n - x^{n-1} - \Delta t^2\mathbf{M}^{-1}f. \tag{2.21}$$

In LS-Dyna[1], the stable time step is calculated and may change over time. Therefore, the update is performed as follows: These equations are repeatedly evaluated until the termination time is reached. It is important to note that the mass matrix $\mathbf{M}$ is a diagonal matrix with mass and rotational inertia on the diagonal entries due to mass lumping. Therefore, inversion of the mass matrix is computationally cheap.

[1]Version: smp d R12, Revision: R12-4012-g7c66dc5b4e

Algorithm 1 Explicit time integration

Input: Initial conditions $x^0, \dot{x}^{\frac{1}{2}}$, boundary conditions, termination time t_{end} and additional parameters μ
Output: Temporal evolution of model $x^{t_n}, \dot{x}^{t_{n+\frac{1}{2}}}$

```
1: t = 0
2: while $t \leq t_{\text{end}}$ do
3:     $f^n(t^n, x^n, \dot{x}^{n-\frac{1}{2}}, \mu) \leftarrow$ calculate force
4:     Compute new acceleration $\ddot{x}^n = \mathbf{M}^{-1} f^n$
5:     $\dot{x}^{n+\frac{1}{2}} = \dot{x}^{n-\frac{1}{2}} + \Delta t_x \ddot{x}^n$
6:     $x^{n+1} = x^n + \Delta t_2 \dot{x}^{n+\frac{1}{2}}$
7:     $t = t + \Delta t_2$
8: end while
```

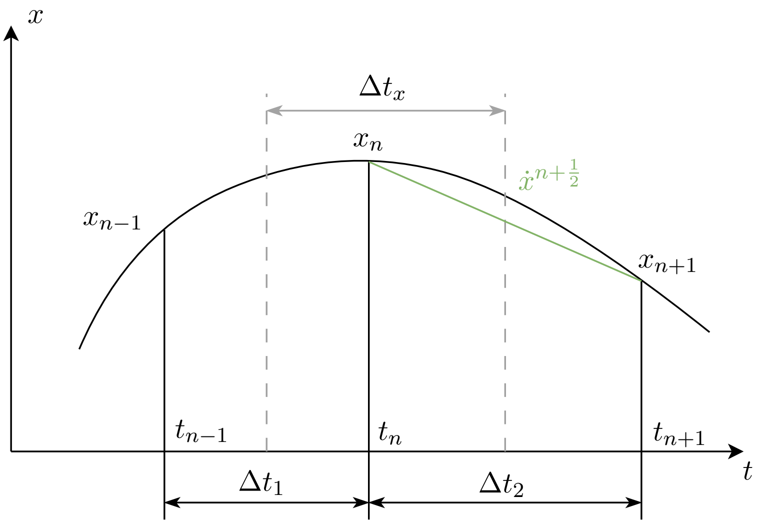


Figure 2.1 Central difference time integration with varying time step.

2.2 Problem Setting and Achievable Speedup

In this section we discuss the computational framework in which we implement an intrusive MOR scheme and discuss the associated bottlenecks. We start by explaining a time integration loop, highlighting difference between a commercial solver and an academic one and comment on computational efforts of the single operations.

Figure 2.2 illustrates the algorithm to proceed one step forward in time as it is typically performed by an explicit nonlinear Finite Element (FE) solver such as Abaqus, LS-Dyna, PamCrash, or Radioss. The unreduced version can be seen on the left side of the dashed line and the reduced version on the right side. First, we consider the unreduced case where all yellow rhombi are left negatively. In the beginning, it is checked whether the termination time has already been reached. If not, the forces are calculated based on the current deformations, velocities, and history variables. The force calculation is the most expensive step of one explicit integration cycle as the time update itself is inexpensive as compared to implicit schemes. Once the forces are known, they are converted to accelerations and the time update returns the velocities and displacements at the new time. Finally, time is incremented and the new state is set as current state.

In this cycle MOR has three possibilities to achieve speed-up:

- Smaller problem size: Projection-based MOR projects the governing equations into a lower dimensional space. This strongly reduces the dimension of the problem. Instead of solving for the state $x \in \mathbb{R}^n$ we solve for the reduced state $\hat{x} \in \mathbb{R}^k$ with $k \ll n$. The reduced dimension k is the sum of the reduced dimension of the displacements d and the reduced dimension of the rotations r. The explicit time update directly benefits from the reduced dimension, however, not as much as implicit schemes do as the explicit update is cheap anyway. One implicit integration cycle involves solving a system of equations iteratively. The repeated use of the small dimension results in an even larger advantage. Usually, the nonlinear force term must be evaluated in physical/full space except some special case depending on the type of nonlinearity. Material models and contact algorithms are usually formulated in physical coordinates. Also in crash applications, forces are not directly computable from reduced quantities and transformations have to be performed in every integration cycle. The transformation consists of a projection and a backprojection, as can be seen in Figure 2.2 on the right side of the dashed line. Both, the projection and the backprojection are associated with computational complexity of $\mathcal{O}(k * n)$ for a linear mapping between full and reduced space. In summary, the transformation costs dominate for pROMs in explicit FE whereas the reduced problem size, due to the more complex time update can dominate for implicit simulations.

- Larger time step: To still obtain a speed-up by just using projection, e.g. when no access is given to the force calculation routines in the solver to apply hyper-reduction, Bach et al. [141] has shown that projection onto the reduced basis filters out the higher modes. More precisely, the largest eigenvalue of the unreduced system will be always larger than the largest eigenvalue of the corresponding ROM. As the critical time step of the central difference scheme is in a simplified manner proportional to $\frac{1}{\sqrt{\lambda}}$ with λ the largest eigenvalue of the system, the critical time step of the FOM is smaller than the critical time step of the ROM. Also Farhat et al. [130] reduced a model of a circular Taylor bar achieving a speed-up of 118 mainly due to a larger admissible time step.

- Hyper-reduction: If access is given to the force calculation routines, hyper-reduction can be applied. Hyper-reduction avoids evaluating the full high-dimensional force term and instead evaluates only selected entries. The selection is an additional optimization problem in the offline phase. ECSW [130] is the most popular hyper-reduction method for solid dynamics problems as it has shown prior stability properties. This is mainly due to preservation of the Lagrangian structure of the underlying problem [131]. The idea for ECSW originated in fluid structure interaction (FSI) problems, where the fluid mesh is usually much finer than the solid mesh. Instead of imposing force balance directly, the difference of the work done by the forces of the fine mesh and of the coarse mesh is optimized. This idea has been transferred to FE models where the original mesh is replaced by a reduced mesh.

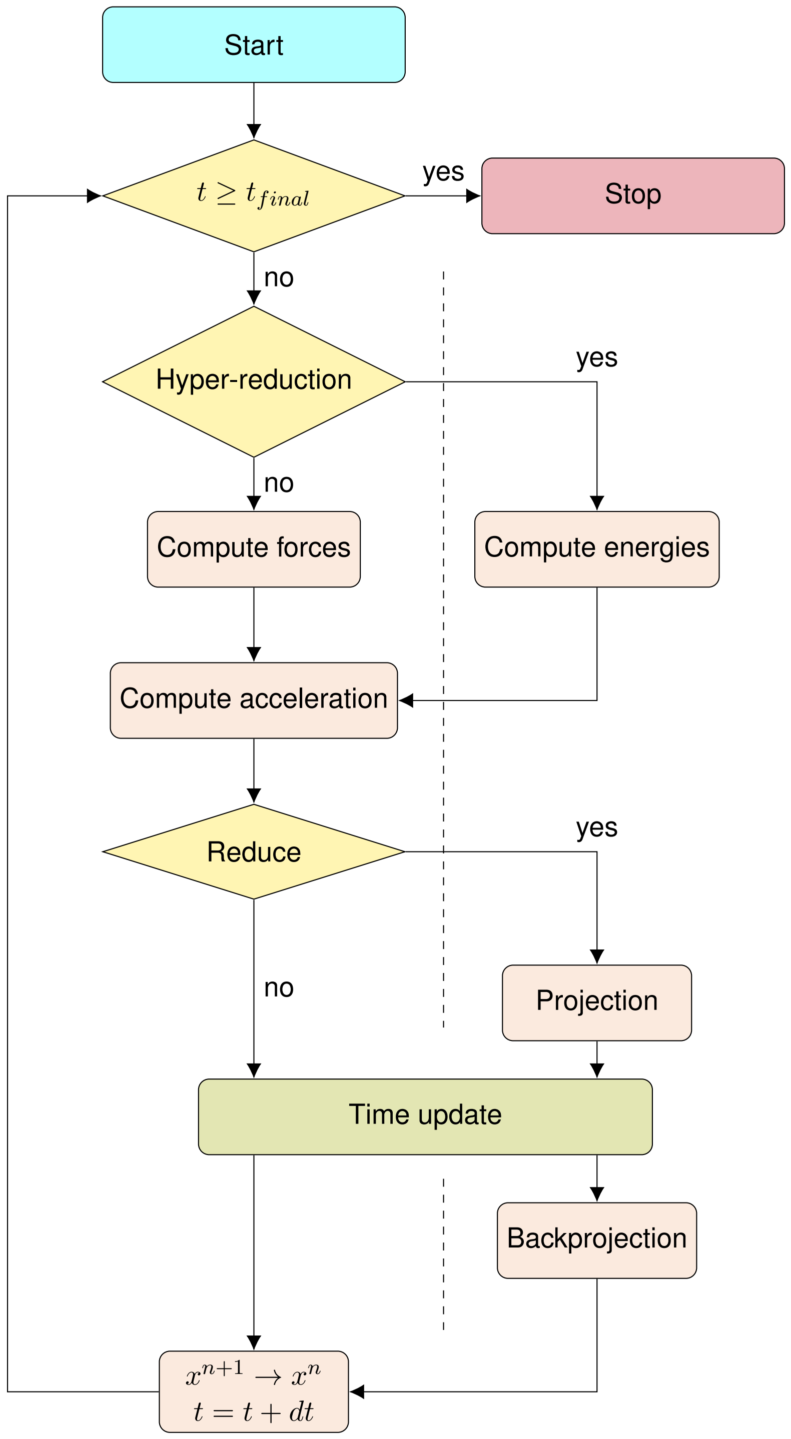


Figure 2.2 Computational graph of a simulation with and without MOR. The steps performed in the FE solver without MOR are shown on the left hand side of the vertical dashed line. The modifications due to MOR are shown on the right hand side of the dashed line. The FOM and the ROM share the same time update. However, the ROM requires additional projection steps and the force calculation is modified based on energies, as described by ECSW.

2.3 Projection-based Model Order Reduction

After discussing the general speedup mechanisms, we derive projection-based MOR related to residual minimization [115] in this section. The necessary foundations for MOR are introduced, on which the rest of the work builds. In this section of the thesis, we treat displacements and rotations separately. Therefore the high-dimensional state is $x \in \mathbb{R}^n$ with n the dimension of either displacements and rotations and k

the corresponding reduced dimension. However, both types of DoF are treated equally, this is why we consider only displacements here.

2.3.1 Subspace Projection

We introduce the projection-based MOR framework without going into the details for the beginning. The first step to construct a ROM is to use a mapping $\Gamma : \mathbb{R}^k \to \mathbb{R}^n$ that maps a low-dimensional state $\hat{x} \in \mathbb{R}^k$ to a high-dimensional state:

$$x \approx \tilde{x} = \Gamma(\hat{x}) \in \mathbb{R}^n. \tag{2.22}$$

We restrict our self in the first chapter of this work to linear mappings although the main motivation behind the different ROMs comes from the difference in solving them for linear and nonlinear mappings. However, all necessary points can be explained using a linear method and POD is currently most popular to compute such a mapping. Using a linear mapping, eq. (2.22) reads:

$$x \approx \tilde{x} = \mathbf{\Phi}\hat{x} = \hat{x}_i(t)\phi_i \quad \text{, for } i = 1, \dots, k, \tag{2.23}$$

where summation over a pairwise appearing index is implied. The high-dimensional state x is decomposed into a sum of temporal coefficients $\hat{x}_i$ and spatial modes $\phi_i \in \mathbb{R}^n$. The sum can be also expressed as a matrix vector product with $\mathbf{\Phi} \in \mathbb{R}^{n \times k}$ the matrix containing the k basis vectors.
The mapping eq. (2.23) is differentiated twice with respect to time and inserted in the governing eq. (2.16):

$$r_1 = \mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + f. \tag{2.24}$$

By inserting an approximation, a residual r_1 is formed. The next step, often referred to as Galerkin projection, is to enforce the residual to be orthogonal to the space spanned by the columns of $\mathbf{\Phi}$. In other words, we solve the right hand side of eq. (2.24) exactly in the reduced space.

$$\mathbf{\Phi}^T r_1 \overset{!}{=} 0 \tag{2.25a}$$
$$\implies \mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T f = \tilde{\mathbf{M}}\ddot{\hat{x}} + \mathbf{\Phi}^T f = 0. \tag{2.25b}$$

In the resulting set of equations $\tilde{\mathbf{M}} = \mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi} \in \mathbb{R}^{k \times k}$ denotes the reduced mass matrix which is no longer a diagonal matrix. This is the standard ROM used in literature for second-order mechanical systems.
However, from an implementation point of view or in combination with nonlinear dimensionality reduction, it may be preferable to work directly with accelerations instead of forces. We try to derive a ROM where accelerations are appearing naturally without the need to invert the reduced mass matrix. First, we transform eq. (2.16) to standard form by multiplying the whole equation with the inverse of the mass matrix:

$$\ddot{x} + \mathbf{M}^{-1} f = 0. \tag{2.26}$$

We insert again the twice differentiated mapping eq. (2.23) and obtain a new residual r_2:

$$r_2 = \mathbf{\Phi}\ddot{\hat{x}} + \mathbf{M}^{-1} f. \tag{2.27}$$

Although different residuals, by multiplying eq. (2.27) with $\mathbf{M}$, the two residuals can be related.

$$r_1 = \mathbf{M} r_2. \tag{2.28}$$

We again enforce the residual r_2 to be orthogonal to the column space of $\mathbf{\Phi}$ and obtain a second ROM which directly works on accelerations:

$$\mathbf{\Phi}^T r_2 \overset{!}{=} 0 \tag{2.29a}$$
$$\implies \mathbf{\Phi}^T\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T\mathbf{M}^{-1} f = \ddot{\hat{x}} + \mathbf{\Phi}^T\ddot{x} = 0. \tag{2.29b}$$

Solving both ROMs results in different solutions and further investigation is needed to show the mathematical differences. The next section highlights the relation of both ROMs to the solution of an optimization problem.

2.3.2 Residual Minimization

Now we relate the previously presented ROMs to residual minimization problems. We first consider time continuous residuals and show that eq. (2.29b) can be interpreted as the solution of an optimization problem of the L^2 norm of residual $\|r_2\|_2$. This derivation is analog to the derivation of the analytical solution of the least squares problem. We state the following optimization problem as proposed by [115]:

$$\ddot{\tilde{x}} = \arg \min_{a \in \mathsf{Ran}(\mathbf{\Phi})} \left\| a + \mathbf{M}^{-1} f(\mathbf{\Phi} x, \mathbf{\Phi} \dot{x}, \mu, t) \right\|_2^2 . \tag{2.30}$$

Since $\ddot{\tilde{x}} = \mathbf{\Phi} \ddot{\hat{x}}$, we can rewrite eq. (2.30) and obtain:

$$\ddot{\hat{x}} = \arg \min_{\hat{a}} \left\| \mathbf{\Phi} \hat{a} + \mathbf{M}^{-1} f \right\|_2^2 . \tag{2.31}$$

We define $g(\hat{a})$ as:

$$\begin{aligned} g(\hat{a}) &= \left\| \mathbf{\Phi} \hat{a} + \mathbf{M}^{-1} f \right\|_2^2 = (\mathbf{\Phi} \hat{a} + \mathbf{M}^{-1} f)^T (\mathbf{\Phi} \hat{a} + \mathbf{M}^{-1} f) \\ &= \hat{a}^T \mathbf{\Phi}^T \mathbf{\Phi} \hat{a} + 2 \hat{a} \mathbf{\Phi}^T \mathbf{M}^{-1} f + f^T \mathbf{M}^{-T} \mathbf{M}^{-1} f \end{aligned} \tag{2.32}$$

and find the minimum of g by solving for the stationary point:

$$\begin{aligned} \frac{\mathrm{d}g}{\mathrm{d}\hat{a}} &= 2 \mathbf{\Phi}^T \mathbf{\Phi} \hat{a} + 2 \mathbf{\Phi}^T \mathbf{M}^{-1} f \overset{!}{=} 0 \\ \implies & \mathbf{\Phi}^T \mathbf{\Phi} \hat{a} + \mathbf{\Phi}^T \mathbf{M}^{-1} f = 0. \end{aligned} \tag{2.33}$$

The solution that solves the time continuous minimization problem of r_2 is equal to ROM2 eq. (2.29b) with $\ddot{x} = \mathbf{M}^{-1} f$:

$$\ddot{\hat{x}} = \hat{a} = -\mathbf{\Phi}^T \mathbf{M}^{-1} f. \tag{2.34}$$

Right now, we have shown that Galerkin projection optimizes the time continuous residual. This result applies to general nonlinear first-order systems of the form $\dot{x} = f(x, t, \mu)$ [115].
Does this result also hold for ROM1 eq. (2.16), which is most often used in literature? To answer this question, we follow the exact same procedure as before and state an L^2 optimization problem, but now for r_1:

$$\ddot{\tilde{x}} = \arg \min_{a \in \mathsf{Ran}(\mathbf{\Phi})} \| \mathbf{M} a + f(\mathbf{\Phi} x, \mathbf{\Phi} \dot{x}, \mu, t) \|_2^2 . \tag{2.35}$$

Since $\ddot{\tilde{x}} = \mathbf{\Phi} \ddot{\hat{x}}$, we can rewrite eq. (2.30) and obtain:

$$\ddot{\hat{x}} = \arg \min_{\hat{a}} \| \mathbf{M} \mathbf{\Phi} \hat{a} + f \|_2^2 . \tag{2.36}$$

We define $g(\hat{a})$ as:

$$\begin{aligned} g(\hat{a}) &= \| \mathbf{M} \mathbf{\Phi} \hat{a} + f \|_2^2 = (\mathbf{M} \mathbf{\Phi} \hat{a} + f)^T (\mathbf{M} \mathbf{\Phi} \hat{a} + f) \\ &= \hat{a}^T \mathbf{\Phi}^T \mathbf{M}^T \mathbf{M} \mathbf{\Phi} \hat{a} + 2 \hat{a} \mathbf{\Phi}^T \mathbf{M}^T f + f^T f \end{aligned} \tag{2.37}$$

and find the minimum of g by solving for the stationary point:

$$\begin{aligned} \frac{\mathrm{d}g}{\mathrm{d}\hat{a}} &= 2 \mathbf{\Phi}^T \mathbf{M}^T \mathbf{M} \mathbf{\Phi} \hat{a} + 2 \mathbf{\Phi}^T \mathbf{M}^T f \overset{!}{=} 0 \\ \implies & \mathbf{\Phi}^T \mathbf{M}^T \mathbf{M} \mathbf{\Phi} \hat{a} + \mathbf{\Phi}^T \mathbf{M}^T f = 0. \end{aligned} \tag{2.38}$$

ROM1 as derived in Section 2.3.1 does not correspond to the solution of the associated L^2 norm optimization problem of the time continuous residual. The question immediately arises to which optimization

problem does the solution of ROM1 correspond then?
To find out, we define a new norm:

$$\|x\|_{\mathbf{A}} = \sqrt{x^T \mathbf{A} x}, \tag{2.39}$$

where $\mathbf{A}$ is a symmetric positive definite matrix. We evaluate the objective function g of eq. (2.37) using the new norm: We define $g(\hat{a})$ as:

$$\begin{aligned} g(\hat{a}) &= \|\mathbf{M\Phi}\hat{a} + f\|^2_{\mathbf{A}} = (\mathbf{M\Phi}\hat{a} + f)^T \mathbf{A} (\mathbf{M\Phi}\hat{a} + f) \\ &= \hat{a}^T \mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} \mathbf{M\Phi}\hat{a} + 2\hat{a}\mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} f + f^T \mathbf{A} f. \end{aligned} \tag{2.40}$$

We find the minimum of g by solving for the stationary point:

$$\frac{\mathrm{d}g}{\mathrm{d}\hat{a}} = 2\mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} \mathbf{M\Phi}\hat{a} + 2\mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} f \overset{!}{=} 0 \tag{2.41a}$$

$$\implies \mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} \mathbf{M\Phi}\hat{a} + \mathbf{\Phi}^T \mathbf{M}^T \mathbf{A} f = 0. \tag{2.41b}$$

By comparing eq. (2.41b) with ROM1 we identify $\mathbf{A}$ as:

$$\mathbf{A} = \mathbf{A}^T = \mathbf{M}^{-1}. \tag{2.42}$$

Therefore, ROM1 is the solution of the optimization problem:

$$\ddot{\hat{x}} = \arg\min_{\hat{a}} \|\mathbf{M\Phi}\hat{a} + f\|^2_{\mathbf{M}^{-1}}. \tag{2.43}$$

2.3.3 Commutativity of Projection and Time Discretization

Now, we show that Galerkin projection and time integration are commutable. First, we apply the time integration scheme on ROM1 to obtain the fully discretized reduced equations. Second, we apply Galerkin projection to the time discretized FOM equations and show equivalence to the first approach. A general discussion for first-order ODE systems in standard form can be found in [115]. The time discrete residual r_1^n is obtained by evaluating r_1 at time $t = t_n$ and is directly enforced to be orthogonal to $\mathbf{\Phi}$:

$$\hat{r}_1|_{t=t^n} = \hat{r}_1^n = \mathbf{\Phi}^T r_1^n \overset{!}{=} 0 \tag{2.44a}$$

$$\hat{r}_1^n = \tilde{\mathbf{M}}\ddot{\hat{x}}^n + \mathbf{\Phi}^T \tilde{f}^n = 0, \tag{2.44b}$$

where:

$$\tilde{f}^n = f(t_n, \mathbf{\Phi}\hat{x}^n, \mathbf{\Phi}\dot{\hat{x}}^{n-\frac{1}{2}}, \mu). \tag{2.45}$$

We now insert the central difference scheme eq. (2.20) in eq. (2.44b) and obtain the implicit formula for the reduced displacement at the next time step $\hat{u}^{n+1}$:

$$\hat{r}_1^n(\hat{u}^{n+1}) = \frac{\tilde{\mathbf{M}}}{\Delta t^2}(\hat{u}^{n+1} - 2\hat{x}^n + \hat{x}^{n-1}) + \mathbf{\Phi}^T \tilde{f}^n = 0. \tag{2.46}$$

In contrast, we now begin with the fully discretized version of eq. (2.16):

$$\frac{\mathbf{M}}{\Delta t^2}(x^{n+1} - 2x^n + x^{n-1}) + \underbrace{f(t_n, x^n, \dot{x}^{n-\frac{1}{2}}, \mu)}_{f^n} = 0. \tag{2.47}$$

In analogy to the continuous case, we insert the linear mapping eq. (2.23) and obtain the time discrete residual r_1^n:

$$r_1^n = \frac{\mathbf{M\Phi}}{\Delta t^2}(x^{n+1} - 2x^n + x^{n-1}) + \tilde{f}^n. \tag{2.48}$$

We enforce the high-dimensional residual r_1^n to be orthogonal to the space spanned by the columns of $\mathbf{\Phi}$ and obtain:

$$\mathbf{\Phi}^T r_1^n = \hat{r}_1^n(\hat{u}^{n+1}) = \frac{\tilde{\mathbf{M}}}{\Delta t^2}(\hat{u}^{n+1} - 2x^n + x^{n-1}) + \mathbf{\Phi}^T \tilde{f}^n \overset{!}{=} 0. \tag{2.49}$$

For both approaches, the same ROM is obtained. The proof can be performed for ROM2 analogously. A derivation for general first-order systems with linear multistep schemes and Runge-Kutta schemes can be found in Carlberg et al. [115].

2.3.4 Least-squares Petrov-Galerkin

Until now, we formulated the residual optimization problems explicitly for time continuous residuals. However, it needs to be shown that time discretization does not destroy the optimality property. A ROM that is a solution to an optimization problem of the time discrete residual was introduced by Carlberg et al. [151] in 2011 and named least-squares Petrov-Galerkin (LSPG). Here, we recall the fundamental result of [115] that LSPG and Galerkin is equivalent for explicit time integration schemes by proving it for the central difference method applied to a second-order ODE. We define a similar optimization problem as in Section 2.3.2, however, we now consider the time discrete residual:

$$\tilde{x}^{n+1} = \arg \min_{u^{n+1} \in \text{Ran}(\mathbf{\Phi})} \left\| r_1^n(u^{n+1}) \right\|_2^2 . \tag{2.50}$$

We can use the linear mapping eq. (2.23) again to solve directly for the reduced quantities:

$$\hat{x}^{n+1} = \arg \min_{\hat{u}^{n+1} \in \mathbb{R}^k} \left\| r_1^n(\hat{u}^{n+1}) \right\|_2^2 = \left\| \frac{\mathbf{M\Phi}}{\Delta t^2}(x^{n+1} - 2x^n + x^{n-1}) + \tilde{f}^n \right\|_2^2 . \tag{2.51}$$

Solving eq. (2.51) yields the same result as the time discretized ROM obtained from optimization problem eq. (2.37):

$$\frac{1}{\Delta t^2}\mathbf{\Phi}^T\mathbf{M}^T\mathbf{M\Phi}(x^{n+1} - 2x^n + x^{n-1}) + \mathbf{\Phi}^T\mathbf{M}^T\tilde{f}^n = 0. \tag{2.52}$$

It is to note again that this is not equivalent to the result obtained by inserting the POD approach in eq. (2.16) and applying Galerkin projection on the residual. The optimal solution differs by the additional term $\mathbf{M}^T$. In contrast, also at the fully discrete level, ROM2 is the minimizer of the associated minimization problem of r_2^n and analog to the time discretized Galerkin ROM2 derived in Section 2.3.2.

2.3.5 Equivalence of ROMs using Mass Orthonormalized and Orthonormal Bases

Often in literature, mass orthonormalization of the reduced basis is used. The question arises whether this ROM represents a new version or is it equal to an already discussed version? We show here that a ROM using a mass orthonormalized basis is equivalent to ROM1.
We begin by introducing the idea of mass orthonormalization. It can be convenient if forces are the input of the time update. No inversion of reduced mass matrix and no conversion from forces to accelerations is necessary then. The definition of a mass orthonormalized basis $\mathbf{\Phi}_M \in \mathbb{R}^{n \times k}$ is implicitly given by:

$$\mathbf{\Phi}_M^T\mathbf{M\Phi}_M = \mathbf{I}, \tag{2.53}$$

where $\mathbf{I} \in \mathbb{R}^{k \times k}$ denotes the identity matrix. The orthonormalized basis $\mathbf{\Phi}_M$ can be found by computing the Cholesky-decomposition of the reduced mass matrix $\tilde{\mathbf{M}}$ and inverting the lower triangular matrix $\mathbf{R}$ afterwards.

$$\mathbf{\Phi}^T\mathbf{M\Phi} = \mathbf{R}^T\mathbf{R}. \tag{2.54}$$

Then, the mass orthonormalized basis reads:

$$\mathbf{\Phi}_M = \mathbf{\Phi R}^{-1}. \tag{2.55}$$

The proof that this new basis fulfills the desired properties is straightforward:

$$\begin{aligned} \mathbf{\Phi}_M^T\mathbf{M\Phi}_M &= (\mathbf{\Phi R}^{-1})^T\mathbf{M\Phi R}^{-1} \\ &= \mathbf{R}^{-T}\underbrace{\mathbf{\Phi}^T\mathbf{M\Phi}}_{\mathbf{R}^T\mathbf{R}}\mathbf{R}^{-1} = \mathbf{I}. \end{aligned} \tag{2.56}$$

Using this new basis, the new linear mapping between the reduced and physical space reads:

$$\tilde{x} = \mathbf{\Phi}_M\hat{x}_M = \mathbf{\Phi}\underbrace{\mathbf{R}^{-1}\hat{x}_M}_{\hat{x}}, \tag{2.57}$$

where $\hat{x}_M$ is a new reduced variable. Since both mappings approximate the same high-dimensional state, a relation between the reduced quantities with and without mass orthonormalization can be found:

$$\hat{x} = \mathbf{R}^{-1}\hat{x}_M. \tag{2.58}$$

Using all the previously derived properties, we show equality of mass orthonormalized ROM with ROM1. First, we derive the ROM for the mass orthonormalized case and show afterwards equivalence with ROM1. Differentiation of the linear mapping eq. (2.57) with respect to time and inserting it in the governing equation (2.16) yields again residual r_1:

$$\mathbf{M}\mathbf{\Phi}_M\ddot{\hat{x}}_M + f = r_1. \tag{2.59}$$

Here again, we enforce r_1 to be orthogonal to the subspace spanned by the columns of $\mathbf{\Phi}_M$:

$$\mathbf{\Phi}_M^T r_1 \overset{!}{=} 0 \tag{2.60a}$$

$$\implies \mathbf{\Phi}_M^T\mathbf{M}\mathbf{\Phi}_M\ddot{\hat{x}}_M + \mathbf{\Phi}_M^T f = \ddot{\hat{x}}_M + \mathbf{\Phi}_M^T f = 0. \tag{2.60b}$$

Now we show that this ROM (eq. (2.60b)) is equivalent to ROM1 (eq. (2.25b)). We insert eq. (2.55) into the ROM:

$$\ddot{\hat{x}}_M + \mathbf{R}^{-T}\mathbf{\Phi}^T f = \mathbf{R}^{-T}\mathbf{\Phi}^T r_1 = 0 \tag{2.61}$$

and use eq. (2.58) and multiply with $\mathbf{R}^T$ from the left side:

$$\mathbf{R}^T\mathbf{R}\ddot{\hat{x}} + \mathbf{R}^T\mathbf{R}^{-T}\mathbf{\Phi}^T f = \mathbf{R}^T\mathbf{R}^{-T}\mathbf{\Phi}^T r_1 = 0. \tag{2.62}$$

Finally, we use eq. (2.54) and obtain ROM1:

$$\mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T f = \mathbf{\Phi}^T r_1 = 0. \tag{2.63}$$

2.3.6 ROM Summary

Table 2.1 summarizes the most important results from Section 2.3.2. At this point, we consider solely time continuous results as we have shown that explicit central difference time integration does not destroy optimality. ROM1_opti is summarized in Table 2.1a. It is the optimal solution of the time continuous residual minimization problem regarding the L^2 norm. However, it is neither physically motivated nor equivalent to the Galerkin projection of r_1. That is why we asked for the minimization problem related to the ROM obtained from Galerkin projection of r_1. The associated minimization problem is stated in Table 2.1b and denoted ROM1. A semi norm using the inverse of the mass matrix is used to formulate the minimization problem. In addition, ROM1 is equal to the ROM obtained by using a mass orthonormalized version.
Finally, a third ROM labeled ROM2 is defined by transforming the governing equation to standard form first. After inserting the low-dimensional approximation, residual r_2 is projected onto $\mathbf{\Phi}$. The Galerkin ROM is simultaneously the solution to the optimization problem of r_2 regarding L^2 norm. However, more interestingly, it is also the solution to an optimization problem of r_1 regarding a semi norm using the square of the inverse of the mass matrix.
To summarize, we defined three ROMs weighting the residual r_1 differently strong. The accuracy of the individual ROM is investigated in the results section 2.4 and the results are interpreted and commented in the conclusion 2.5.

Governing equation	$\mathbf{M}\ddot{x} + f = 0$
Residual	$r_1 = \mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + f$
Minimization problem	$\min \lVert r_1 \rVert_2^2$
Solution	$\mathbf{\Phi}^T\mathbf{M}^T\mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T\mathbf{M}^T f = 0$

(a) ROM1_opti

Governing equation	$\mathbf{M}\ddot{x} + f = 0$
Residual	$r_1 = \mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + f$
Minimization problem	$\min \lVert r_1 \rVert_{\mathbf{M}^{-1}}^2$
Solution	$\mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T f = 0$

(b) ROM1

Governing equation	$\ddot{x} + \mathbf{M}^{-1} f = 0$
Residual	$r_2 = \mathbf{\Phi}\ddot{\hat{x}} + \mathbf{M}^{-1} f = \mathbf{M}^{-1} r_1$
Minimization problem	$\min \lVert r_2 \rVert_2^2 = \min \lVert r_1 \rVert_{(\mathbf{M}^{-1})^2}^2$
Solution	$\ddot{\hat{x}} + \mathbf{\Phi}^T\ddot{x} = 0$

(c) ROM2

Table 2.1 Summary of all different ROM versions. Note that f is evaluated at $f(\hat{x}, \dot{\hat{x}})$

2.3.7 POD

So far, it is not clear how to compute the ROB $\mathbf{\Phi}$. In this section, we utilize POD to effectively reduce displacements as well as rotations, which typically appear in the context of crash simulations. It is to note that reproductive examples are considered. Parameter variations lead to a slow decaying Kolmogorov n-width and other methods must be utilized, which is shown later in this thesis. Cars consist of many sheet metal parts which are modeled in the FE model as shell elements. These elements reduce the total number of DoFs to solve for as assumptions about the kinematics and kinetics are introduced. However, additional rotational DoFs are inserted which are of different type as translational DoFs. The following section is structured as follows: first, we introduce POD and define an offline error to judge accuracy of POD approximation. Second, we comment on how to utilize Singular Value Decomposition (SVD) to compute a ROB. POD computes a modal decomposition of a field $u(x,t) \in \mathbb{R}^n$ into temporal coefficients $a_i(t) \in \mathbb{R}$ and spatial modes $\phi_i \in \mathbb{R}^n$:

$$u(x,t) = \sum_{i=1}^{M} a_i(t)\phi_i(x), \tag{2.64}$$

where M is the number of modes to be used. The idea is to replace the sum over all M modes by a sum over the k dominant modes with $k \ll M$. The modes are selected to be optimal regarding the mean squared error ϵ:

$$\min_{\phi_i} \epsilon(k) = \langle \lVert u - u(k) \rVert_2^2 \rangle, \tag{2.65}$$

where $u(k)$ denotes the reconstruction of u using the first k modes and $\langle\cdot\rangle$ is the average, as defined in eq. (2.79). Without further derivation, we claim that the solution to optimization problem eq. (2.65) is the eigenvalue problem:

$$\mathbf{C}\phi_i = \lambda_i \phi_i. \tag{2.66}$$

The covariance matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$ is defined using the snapshot matrix $\mathbf{U} \in \mathbb{R}^{n\times N_s}$ and N_s the number of collected solution snapshots $u^{t_i} \in \mathbb{R}^n$:

$$\mathbf{C} = \frac{1}{N_s - 1}\mathbf{U}\mathbf{U}^T, \tag{2.67}$$

with

$$\mathbf{U} = \begin{bmatrix} u^{t_1}, & u^{t_2}, & \dots, & u^{t_{N_s}} \end{bmatrix} \tag{2.68}$$

and therefore

$$\mathbf{C} = \frac{1}{N_s - 1} \begin{bmatrix} \sum_{i=1}^{N_s} u_1^{t_i} u_1^{t_i} & \sum_{i=1}^{N_s} u_1^{t_i} u_2^{t_i} & \dots & \sum_{i=1}^{N_s} u_1^{t_i} u_n^{t_i} \\ \sum_{i=1}^{N_s} u_2^{t_i} u_1^{t_i} & \sum_{i=1}^{N_s} u_2^{t_i} u_2^{t_i} & \dots & \sum_{i=1}^{N_s} u_2^{t_i} u_n^{t_i} \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{N_s} u_n^{t_i} u_1^{t_i} & \sum_{i=1}^{N_s} u_n^{t_i} u_2^{t_i} & \dots & \sum_{i=1}^{N_s} u_n^{t_i} u_n^{t_i} \end{bmatrix}. \tag{2.69}$$

Once eigenvalue problem eq. (2.66) is solved, the eigenvectors ϕ_i can be arranged in a matrix $\mathbf{\Phi}_{full}$ ordered correspondingly to the eigenvalues λ_i in descending order $\lambda_{i+1} \leq \lambda_i \leq \lambda_{i-1}$:

$$\mathbf{\Phi}_{full} = \begin{bmatrix} \phi_1, & \phi_2, & \dots, & \phi_n \end{bmatrix}. \tag{2.70}$$

To achieve a dimensional reduction, the least important basis vectors are truncated and a ROB $\mathbf{\Phi}$ can be defined using the truncation rank k:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1, & \phi_2, & \dots, & \phi_k \end{bmatrix}. \tag{2.71}$$

We choose the following error norm to select a reasonable k:

$$\epsilon^2(k) = \frac{\left\| \mathbf{U} - \tilde{\mathbf{U}} \right\|_F^2}{\|\mathbf{U}\|_F^2} = \frac{\left\| \mathbf{U} - \mathbf{\Phi}(k)\mathbf{\Phi}^T(k)\mathbf{U} \right\|_F^2}{\|\mathbf{U}\|_F^2} = \frac{\sum_{i=k+1}^{N_s} \lambda_i}{\sum_{i=1}^{N_s} \lambda_i}, \tag{2.72}$$

where $\|\|_F$ denotes the Frobenius norm. Once the approximation rank is determined, the temporal coefficients can be computed by projecting the high-dimensional state u onto the basis vectors ϕ_i:

$$a_j = \langle u, \phi_j \rangle \quad \text{for} \quad j \in 1, \dots, k. \tag{2.73}$$

It is to note, that $\langle \cdot, \cdot \rangle$ denotes a scalar product in eq. (2.73), whereas in eq. (2.65) it denotes the averaging operator. For the n-dimensional case, eq. (2.73) transforms to:

$$a^{t_i} = a(t_i) = \mathbf{\Phi}^T u^{t_i}, \tag{2.74}$$

where $a^{t_i} = a(t_i) \in \mathbb{R}^k$ is also called the reduced state at time t_i. All high-dimensional states in time can be transformed to a reduced state in a single step by using matrix notation:

$$\mathbf{A} = \begin{bmatrix} a(t_1), & a(t_2), & \dots, & a(t_{N_s}) \end{bmatrix} = \mathbf{\Phi}^T \mathbf{U}. \tag{2.75}$$

At the end, we want to highlight the similarity to SVD as it is often used in literature to compute the ROB. The SVD of the snapshot matrix $\mathbf{U}$ is given by:

$$\mathbf{U} = \mathbf{\Phi}_{full} \mathbf{\Sigma} \mathbf{Z}^T. \tag{2.76}$$

The left singular vectors are equal the orthogonal basis vectors of the ROB, $\mathbf{\Sigma} \in \mathbb{R}^{n \times N_s}$ contains the singular values σ_i which are related to the eigenvalues of the covariance matrix by:

$$\lambda_i = \frac{\sigma_i^2}{N_s - 1}. \tag{2.77}$$

The right singular values are contained in $\mathbf{Z}^T \in \mathbb{R}^{N_s \times N_s}$. Both $\mathbf{\Phi}_{full}$ and $\mathbf{Z}^T$ are orthonormal matrices containing the eigenvectors of the matrices $\mathbf{U}\mathbf{U}^T$ and $\mathbf{U}^T\mathbf{U}$, respectively.

2.4 Results

In this section, we introduce two examples with relevant properties of typical crash load cases to apply the different ROMs. First we consider a pipe whip example involving shell elements, contact and plastic deformation. Second we consider the popular crash box example involving shell elements, contact, self contact and plasticity. We apply the three different ROM versions summarized in Table 2.1 and evaluate their accuracy based on two error measures.

2.4.1 Online Error Measure

To judge the quality of the obtained ROM solutions, we first need to introduce suitable error measures. The online error measures consider purely displacements. Rotations or other types of DoF are not considered to evaluate the online accuracy of the created ROMs, although considered for the offline error. The first considered online error is time dependent and the second error is related to the temporal average of the first one. Depending on the problem, a different error metric is more appropriate.
We begin to define the time dependent error:

$$\epsilon(t) = \frac{\|u(t) - \tilde{u}(t)\|_2}{\langle \|u(t)\|_2^2 \rangle^{\frac{1}{2}}}, \tag{2.78}$$

where $u(t) \in \mathbb{R}^n$ denotes the reference solution and $\tilde{u}(t) \in \mathbb{R}^n$ the ROM solution. Here, $\langle \cdot \rangle$ is the temporal averaging operator defined as:

$$\langle u(t) \rangle = \sum_{i=1}^{n} \frac{u(t_i)}{n}. \tag{2.79}$$

Crash phenomena usually happen during a small temporal duration and do not exhibit a periodic solution like an undamped cantilever beam. Therefore, it makes sense to also define an error that compares two solutions in a temporal averaged sense. We achieve this by temporal averaging the numerator and obtain:

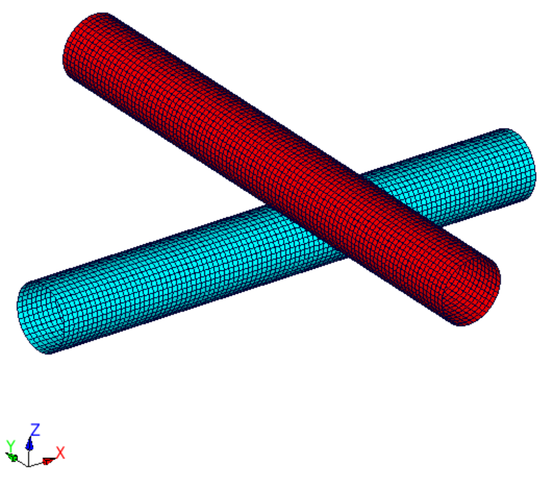
$$\epsilon_{\text{GRE}} = \frac{\langle \|u(t) - \tilde{u}(t)\|_2^2 \rangle^{\frac{1}{2}}}{\langle \|u(t)\|_2^2 \rangle^{\frac{1}{2}}}. \tag{2.80}$$

If the number of timesteps is the same, we obtain the global relative error (GRE), as defined by [130]:

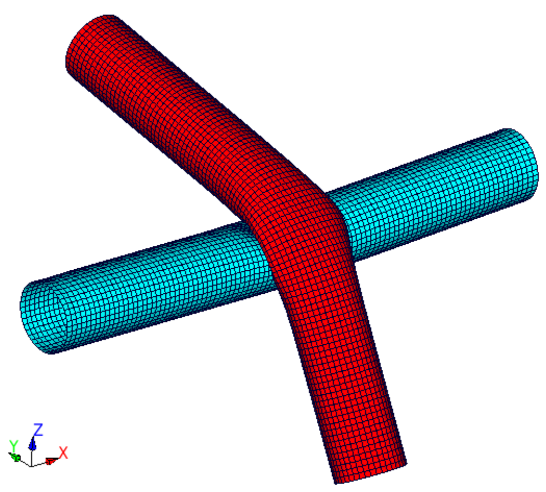
$$\epsilon_{\text{GRE}} = \frac{\sqrt{\sum_{i=1}^{N_s} (u(t_i) - \tilde{u}(t_i))^T (u(t_i) - \tilde{u}(t_i))}}{\sqrt{\sum_{i=1}^{N_s} u(t_i)^T u(t_i)}}. \tag{2.81}$$

2.4.2 Pipe Whip

Model Description



(a) Initial configuration at $t = 1$ ms



(b) Deformed configuration at $t = 15$ ms

Figure 2.3 Pipe whip model in initial configuration (a) and deformed configuration (b).

Here we consider the pipe whip example taken from LS-Dyna examples homepage. The model can be seen in the initial configuration in Fig. 2.3a and in the deformed configuration in Fig. 2.3b at the termination

time of $15\,\mathrm{ms}$. The example consists of two parts colored in red and cyan and labeled tube 1 and tube 2, respectively. Both tubes are modeled with fully integrated shell elements implemented with the enhanced assumed strain (EAS) algorithm and standard LS-Dyna viscous form hourglass control. Each tube with a length of $1270\,\mathrm{mm}$ and a midsurface diameter of $57.3\,\mathrm{mm}$ is discretized with 4000 elements. The thickness of both tubes is $10.97\,\mathrm{mm}$. The second-order central difference time integration scheme is chosen to step forward in time. The time step for the reference solution is chosen to be 90% of the stable timestep. In the ROM simulations, the time step is fixed to $0.0015\,\mathrm{ms}$ to ensure comparability between different ROMs. Single point constraints (SPCs) are applied to the nodes at the end of tube 2 which block motion in all 6 DoFs. The nodes at the end of tube 1 in positive y-direction are defined as rigid body nodes. As initial condition, an initial velocity is applied to the whole part. The velocity of each node is defined by a rigid body rotation with a rotation axis parallel to the x-axis passing through the midpoint of the circle defined by the rigid body nodes of tube 1. The initial angular velocity is $75\,\mathrm{s}^{-1}$.
Both parts are modeled with an elasto-plastic material with kinematic hardening. The mass density is $7823.59\,\mathrm{kg\,m}^{-3}$, the Poisson ratio is 0.3, the Young's modulus equals $2.07 \times 10^5\,\mathrm{N\,mm}^{-2}$, the yield stress is $310.5\,\mathrm{N\,mm}^{-2}$, and the tangent modulus equals $0\,\mathrm{N\,mm}^{-2}$. A vanishing tangent modulus represents a material without hardening.

Offline Accuracy

In this section we present the POD approximation error $\epsilon^2(k)$ as defined by eq. (2.72) for the pipe whip example. The error in Fig. 2.4 is plotted on a logarithmic y axis for increasing number of used modes k. Nodal displacement output is collected every $0.01\,\mathrm{ms}$. Hence, 1501 snapshots are collected along $15\,\mathrm{ms}$ simulation time and passed to the SVD. Numpy's [152] SVD algorithm was utilized to compute the ROB and the corresponding eigenvalues for error visualization. However, if the truncation rank is known previously, we recommend randomized algorithms such as facebook's Principal Component Analysis, which is considerably faster.
We highlight two things. First, displacements as well as rotations possess a low-rank structure, as can be seen by the quickly decaying error. That is, both quantities can be represented using less dimensions and the preliminaries to construct a ROM are satisfied. However, it can be observed that rotational DoFs are harder to reduce than translational ones. Whether this observation is problem specific or general cannot be answered at this point. However, we show that this behavior also applies to a crash box example with varying wall thickness.

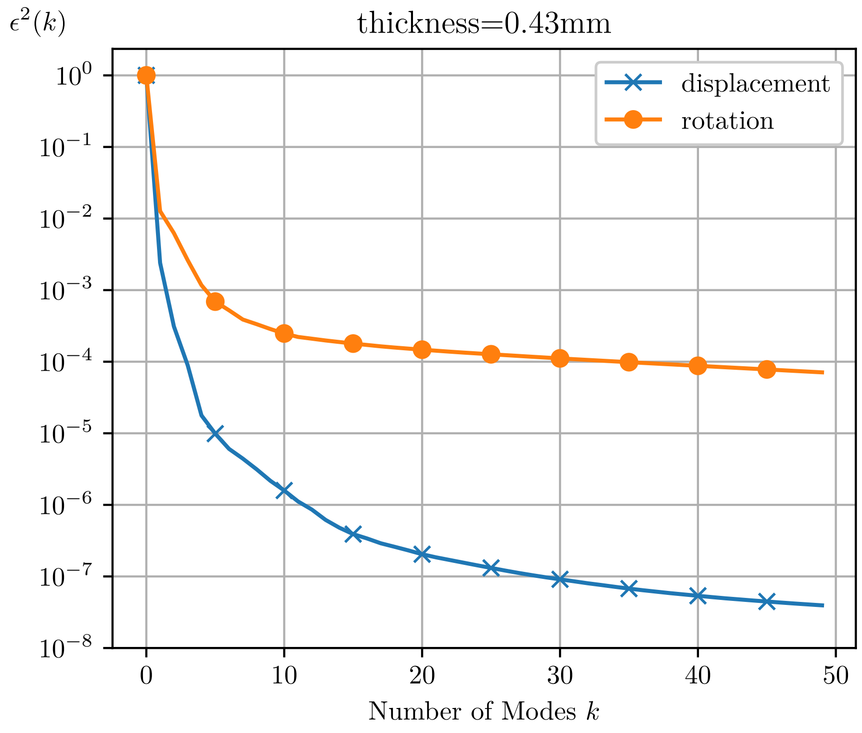


Figure 2.4 POD error of the pipe whip example for increasing number of modes for a fixed tube thickness of t=0.43 mm.

ROM Comparison

In this section, we use the resulting ROB from SVD and construct three different ROMs as summarized in Table 2.1. We label the ROMs accordingly ROM1_opti, ROM1 and ROM2. Fig. 2.5 shows the temporal error eq. (2.78) for all three ROMs. Each subplot corresponds to a ROM constructed using a ROB with a certain number k of basis vectors (modes). Each row shows the same y axis but different ranges are applied for better visibility. The error curves share the same general behavior. The error starts at zero, increases over time and decreases at the end. Only ROM2 deviates from this behavior for 25, 30 and 35 modes, where a first drop in error can be seen before it increases again. The decrease of the error at the end is likely due to elastic spring back of the tube once it has deformed. In this region, the solutions are getting closer again due to the sign change in velocity.
That the error decreases for an increasing number of modes is only guaranteed for ROM1. It shows the most consistent behavior and the smoothest error decay for an increasing number of modes. The accuracy of ROM1 lies in between ROM1_opti and ROM2 until 20 modes, as can be also seen in Fig. 2.6 where the averaged error, as defined in eq. (2.81), is plotted against an increasing number of modes. ROM1_opti has the lowest error for less than 25 modes. ROM2 has the lowest accuracy among all numbers of modes. Only for 40 modes, ROM2 has approximately the same error than ROM1. Fig. 2.6 summarizes all previous results by plotting GRE against an increasing number of modes. ROM1 has a consistent decrease in error with an increasing number of modes. With less than 25 modes, ROM2 performs worse and ROM1_opti performs best. Using more than 25 modes the trend changes and ROM1_opti has the lowest accuracy and ROM2 has the highest GRE except for 35 modes.

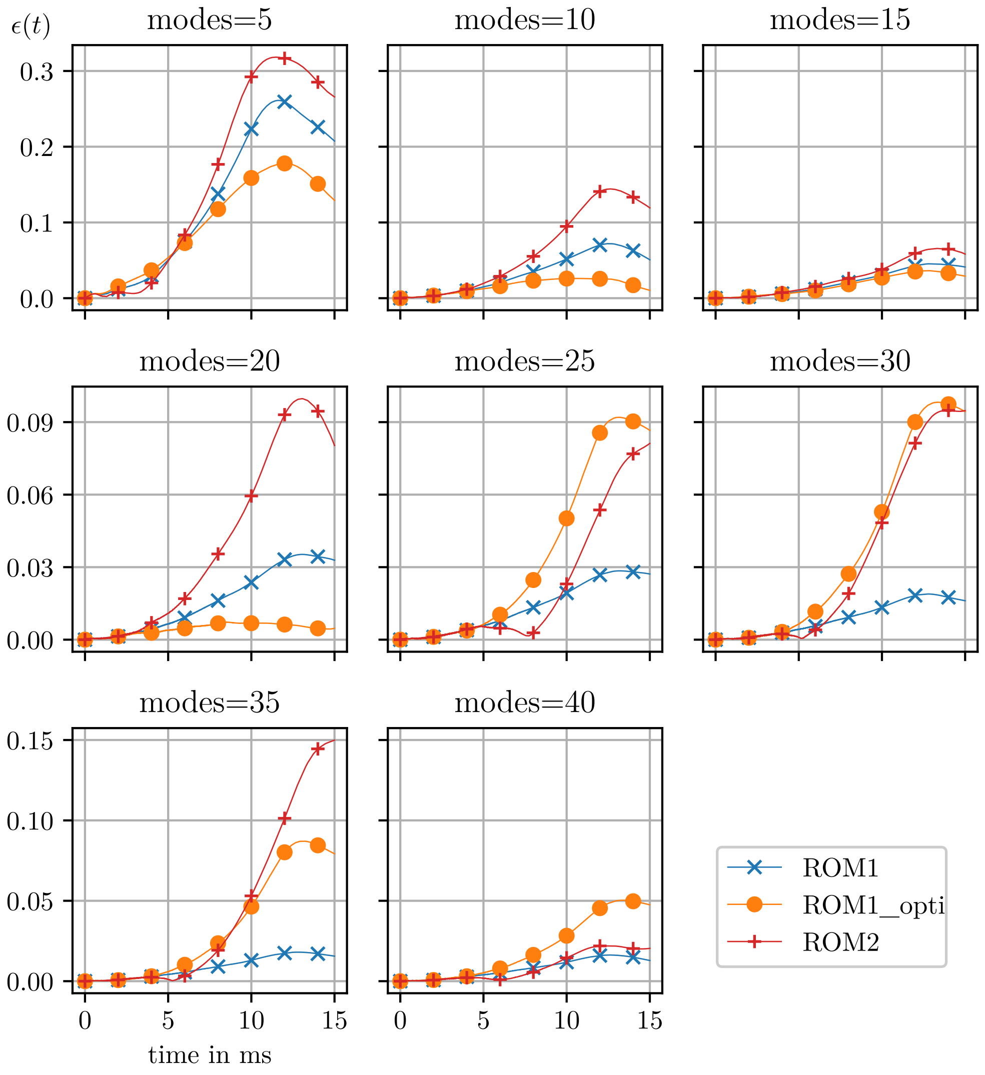


Figure 2.5 Temporal error of the pipe whip example plotted against time for increasing number of modes for a fixed tube thickness of t=0.43 mm.

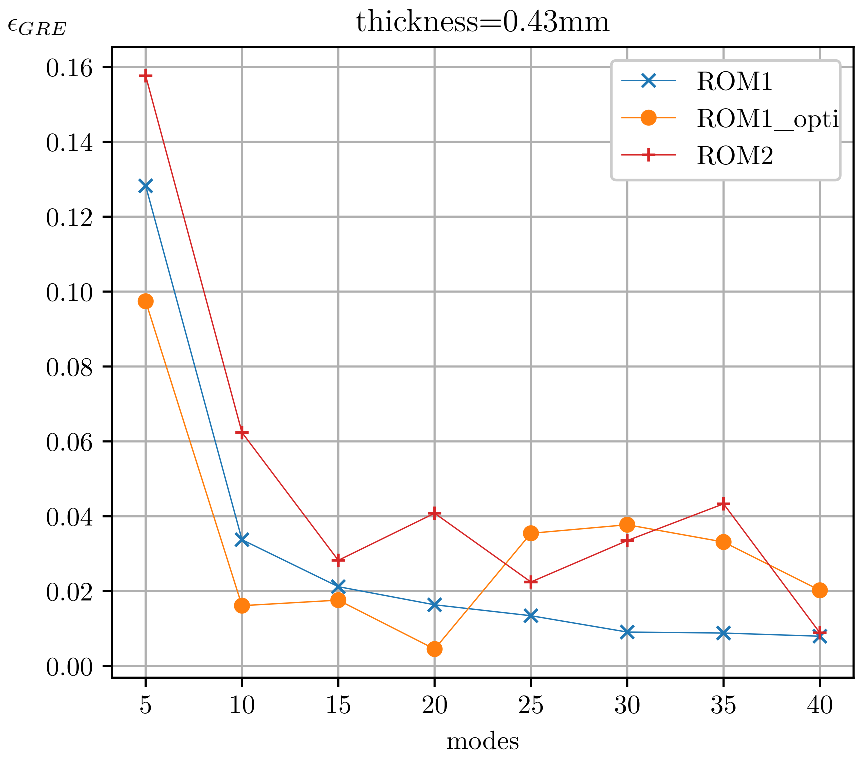
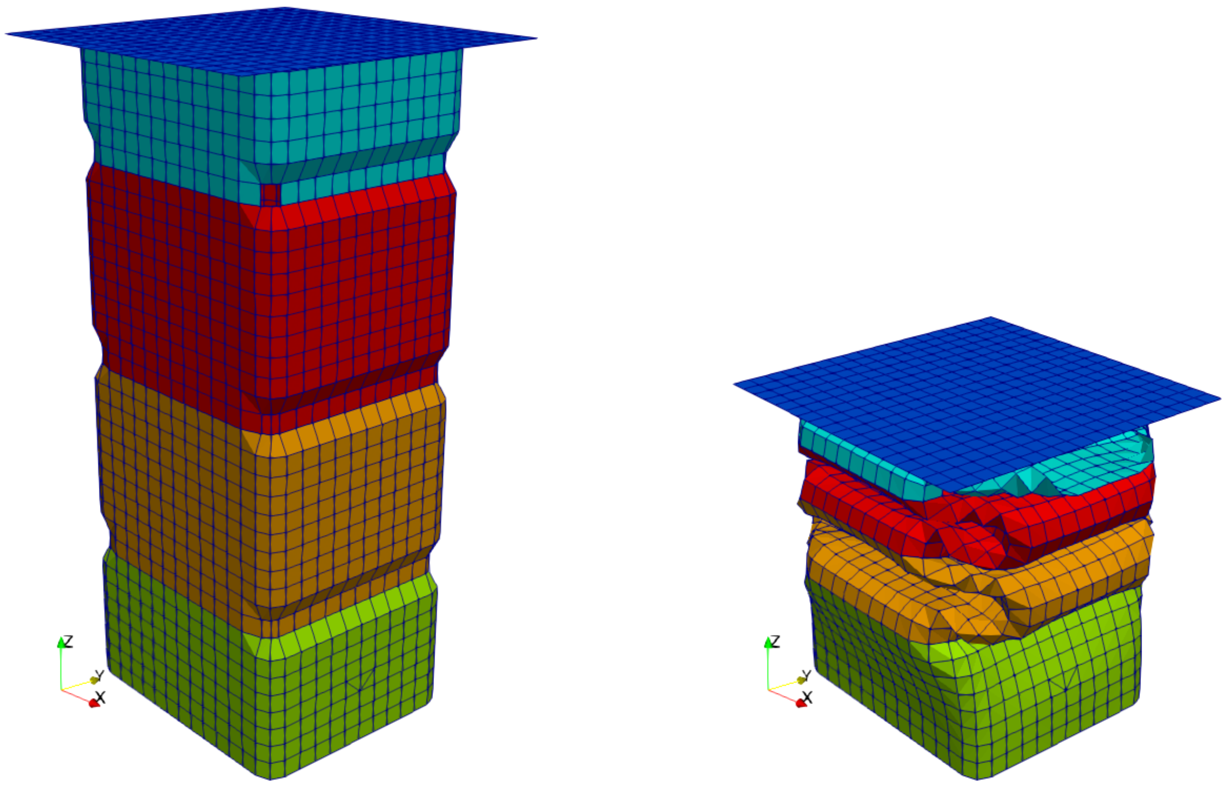


Figure 2.6 GRE of pipe whip example plotted against increasing number of modes for a fixed tube thickness of t=0.43 mm.

2.4.3 Crash Box

Model Description



(a) Initial configuration at $t = 1$ ms   **(b)** Deformed configuration at $t = 35$ ms

Figure 2.7 Crash box model in initial configuration (a) and deformed configuration (b).

Next, we consider a crash box example, which is established as standard example for crash analysis. The initial and deformed configurations for a crash box with $2\,\mathrm{mm}$ wall thickness can be seen in Fig. 2.7a and 2.7b, respectively.
A rigid plate with an initial velocity of $11.11\,\mathrm{m\,s^{-1}}$ in negative z-direction and a mass of $157\,\mathrm{kg}$ impacts a deformable tube. The tube consists of 4 parts, colored differently, where the wall thickness can be varied individually. However, in this study all 4 sections share the same wall thickness that is varied between $1.2\,\mathrm{mm}$ and $2.6\,\mathrm{mm}$ in $0.2\,\mathrm{mm}$ steps. The dimensions of the crash box's cross-section with respect to the midsurfaces is $116\,\mathrm{mm}{\times}96\,\mathrm{mm}$ and the height of the tube is approximately $272.5\,\mathrm{mm}$. The crash box is modeled using 1860 quadrilateral and 8 triangular shell elements of LS-Dyna type 16, which are fully integrated shell elements implemented with EAS algorithm and standard LS-Dyna viscous form hourglass control. The material model is *MAT_24 which is a piece-wise linear J2-based plasticity model. The mass density is $7830\,\mathrm{kg\,m^{-3}}$, the Poisson ratio is 0.3, the Young's modulus equals $2.0 \times 10^5\,\mathrm{N\,mm^{-2}}$, the initial static yield stress is $366\,\mathrm{N\,mm^{-2}}$ and the values for the plastic stress strain curve are given in Table 2.2. Strain rate dependencies are treated with the Cowper-Symonds model with C = 40 and p = 5. The tube is fixed at the bottom with single point constraints (SPCs) prohibiting movement in all 3 translational and rotational DoFs. Contact is handled by the LS-DYNA Keyword *CONTACT_AUTOMATIC_SINGLE_SURFACE which implements a single surface contact where contact is checked between all parts as well as self contact is checked for each part. The static and dynamic friction coefficients are 0.08. The end time is 35 ms and automatic time step computation with security factor of 0.9 is chosen for the reference solution. All ROMs are computed with a fixed time step of 5×10^{-4} ms.

equivalent plastic strain	yield stress$\mathrm{N\,mm^{-2}}$
0.000	366
0.025	424
0.049	476
0.072	507
0.095	529
0.118	546
0.140	559
0.182	584

Table 2.2 Evolution of the yield stress dependent on the equivalent plastic strain, as given by the LS-Dyna examples homepage.

Offline Accuracy

In this section we present the POD approximation error $\epsilon(k)$ as defined by eq. (2.72) for a crash box example with varying wall thickness. The error in Fig. 2.8 is plotted on a logarithmic y-axis for increasing number of used modes k. We collect 3501 displacement snapshots during $35\,\mathrm{ms}$ simulation time and use again Numpy's [152] SVD standard algorithm to compute the ROB. It is to note that only the tube without the rigid plate is considered for POD. In analogy to the pipe whip example, displacements as well as rotations possess a low-rank structure and the error quickly decays with increasing number of modes. However, it can be seen that the variance in the dataset of displacements can be captured using less modes than there would be necessary for the rotational DoFs. This trend is determined here by experience, but not universally proven. It is likely due to the manifold character of rotational DoFs mentioned by Farhat et al. [130]. In addition, no connection can be established between the wall thickness and the divergence of displacement and rotation curve in Fig. 2.8.

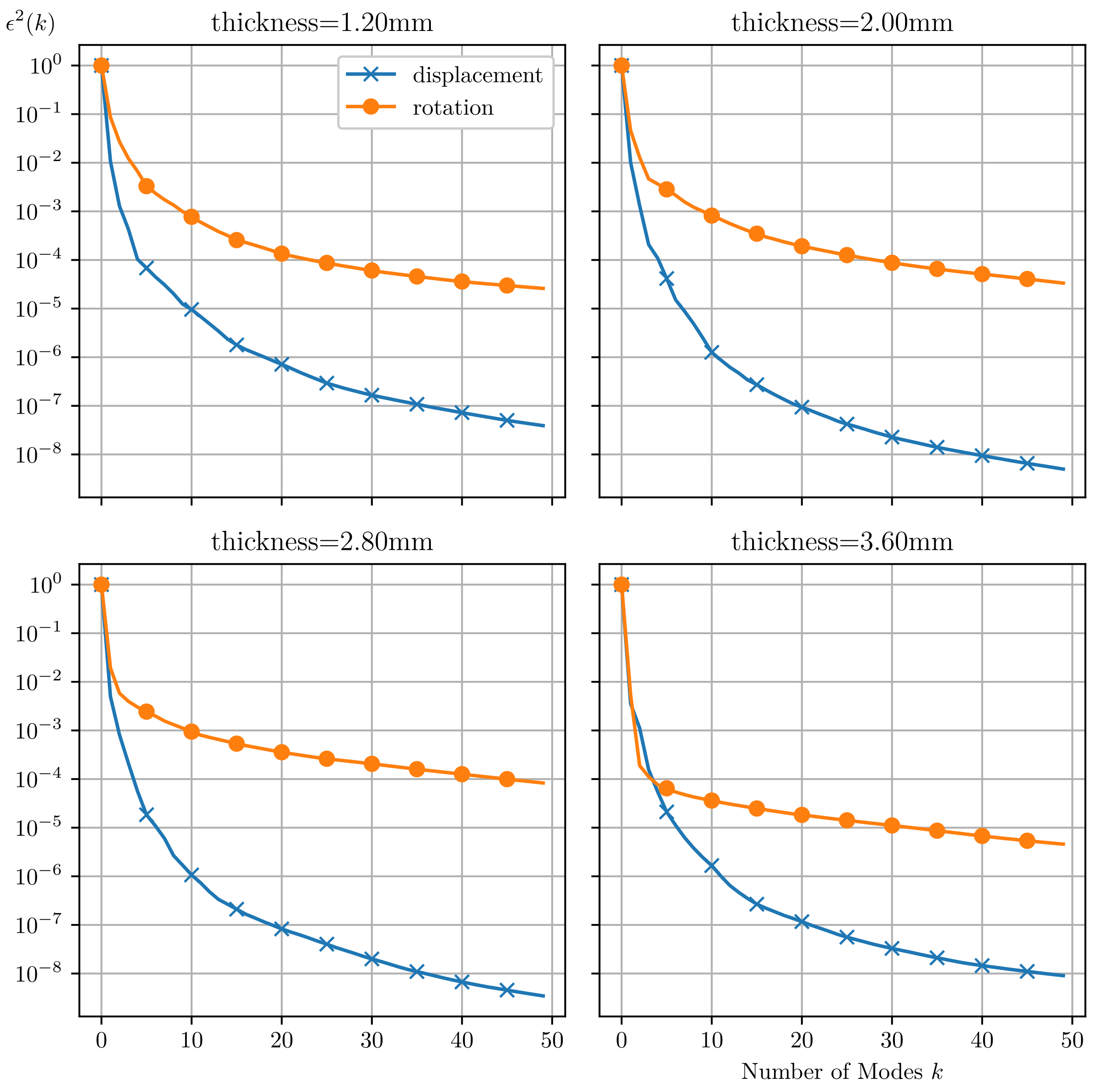
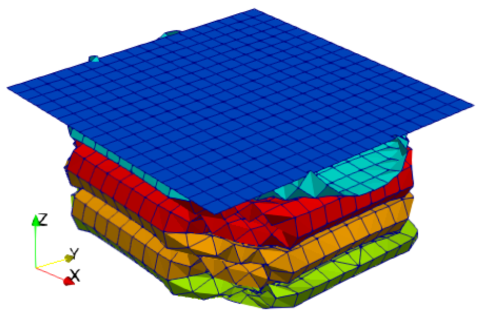


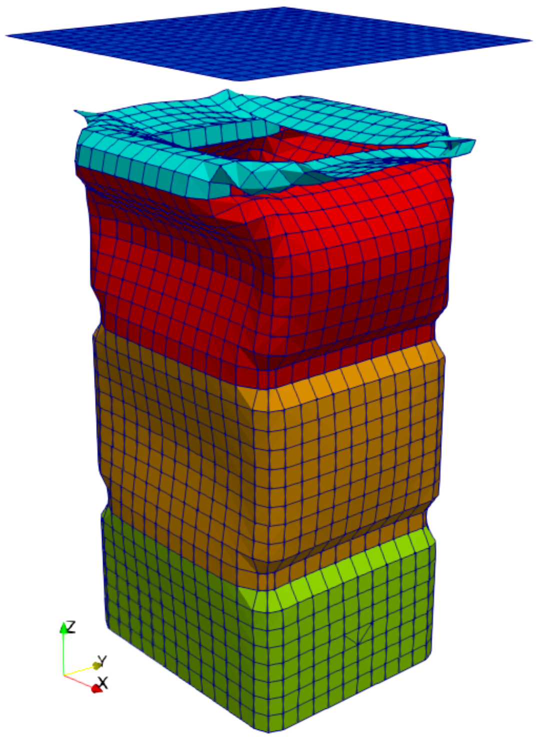
Figure 2.8 POD error for the crash box example for varying wall thickness.

ROM Comparison

We now adopt the ROB from the previous section and construct three different ROMs which minimize different PDE residuals in different ways as described in section 2.3.2 and summarized in Table 2.1. To make statements about the influence of the wall thickness of the crash box on the accuracy of the ROM methods, the wall thickness is varied from 1.2 mm to 3.6 mm in 0.2 mm steps. The difference in total deformation can be seen in Fig. 2.9a and Fig. 2.9b where both figures show the deformed configuration at $t = 35$ ms for a tube with the lowest and the highest wall thickness, respectively.



(a) Crash box with wall thickness 1.2 mm at t = 35 ms.



(b) Crash box with wall thickness 3.6 mm at t = 35 ms.

Figure 2.9

Fig. 2.10 shows the GRE as defined in eq. (2.81) plotted against an increasing number k of modes for different wall thicknesses. ROM1 shows a monotonous increase in accuracy for increasing number of modes consistently among all wall thicknesses. ROM2 and ROM1_opti cannot guarantee this desirable behavior. Especially for thin tubes, all three ROMs yield similar accuracy. For 3 mm and thicker tubes, ROM1 and ROM2 are similarly accurate. ROM1_opti behaves too stiff in this thickness region and yields the least accurate results. In addition, an increase in accuracy with increasing number of modes cannot be guaranteed. For 5 modes, ROM2 is often the most accurate model. In general, the increase in accuracy for increasing modes is less for thicker tubes.

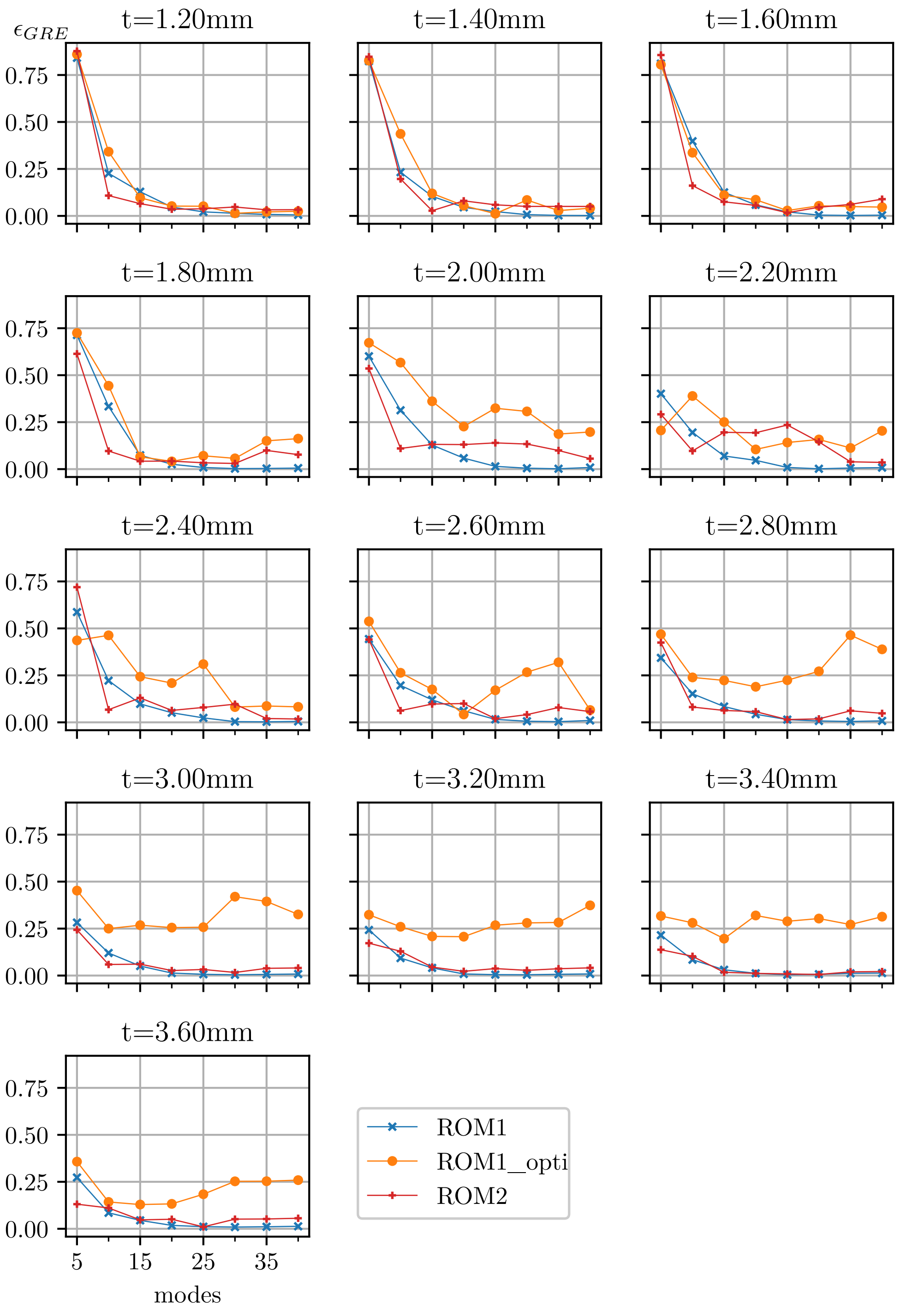


Figure 2.10 Generalized relative error of three different ROM architectures plotted against an increasing number of modes for the crash box example with increasing wall thickness.

2.5 Conclusion

In this chapter, we discuss pMOR for nonlinear finite element models in crash and impact applications. We discuss the mechanisms to achieve speedup and mainly study different residual minimization problems.
Two crash models are used to investigate POD dimension reduction and residual minimization. We consider a pipe whip example with a fixed tube thickness and a crash box example with varying wall thickness. FE models in crash applications usually contain shell assumptions, which is why we discuss POD and compare the low rank structure of displacements and rotations. We experimentally show that rotational DoFs are harder to reduce, most likely due to their manifold character. The pipe whip example as well as all different crash box examples show this behavior.
In the pipe whip example, all ROMs show the trend of a decreasing GRE for an increasing number of modes. However, only ROM1 strictly shows this behavior also for higher number of modes. For very low modes, ROM1_opti works best, which corresponds to the L^2 optimization of the residual. By further weighting the residual with the inverse of the mass, which yields ROM1 and ROM2, the accuracy decreases. These ROMs give more importance to residuals at nodes with a small mass. As the nodal mass is usually smaller than 1, the residual is weighted with a number larger than 1 and the residual is constraint tighter than in the L^2 case. The same applies to ROM2 in an even stronger manner, where the residual is weighted with $\frac{1}{\text{mass}^2}$. However, ROM2 has the lowest accuracy among all ROMs. ROM1 mirrors the principle of virtual work. Hence, we conclude this is the reason that it shows the most desirable behavior. That is, by increasing the number of modes in the ROM, the GRE consistently decreases.
The crash box example adds another level of complexity by including folding and self contact. In addition, the thickness variation leads to different degrees of deformation. Also here, ROM1's GRE consistently decreases by adding additional modes in the ROM construction. The decrease in error is steeper for thinner tubes than for thicker tubes. Thin-walled tubes deform in all regions of the tube, whereas thick tubes deform only locally at the impact region. With thin walls, the weight of the residual matters little and all three ROMs have about the same accuracy. As we move to thicker walls, deformation becomes more locally and an equal weighting in an L^2 sense of all nodes in the tube does not seem appropriate anymore. ROM1 and ROM2 yield approximately the same accuracy for thick tubes. In some low mode regions, ROM2 with a stronger weighting performs slightly better.
To summarize, ROMs are sensitive regarding the chosen error norm. ROM1 exhibits the most consistent error decrease. This is due to the physical motivation behind ROM1. It follows the principle of virtual work as the linear basis can be interpreted as virtual displacements and reduced quantities have the units of physical work. This is also shown in Chapter 5. Since ROM1 is the only ROM exhibiting a consistent behavior, we use ROM1 for all following chapters. Despite the results of this study, more work should investigate how the chosen residual minimization influences the stable time step of the explicit time integration and how the chosen time step influences the ROM results. Accuracy variations due to the chosen time-step could be observed for ROM2 and ROM1_opti. Also, instabilities for simple beam examples were observed for those ROMs. Therefore, the influence of the contact algorithm on the ROM stability is of interest. Finally, how ECSW interacts with different ROMs should be investigated, since the reduced force term does not represent a virtual work in all ROMs and hyper-reduction is the only possibility to achieve speedup in explicit nonlinear pMOR.

3 Local Reduced-order Bases

This chapter introduces pROMs for parameter variations. They are necessary for parametric applications, such as optimization or robustness studies. We begin by describing the varied parameters and evaluating the offline accuracy of POD. Next, we choose a global strategy for ROM1, which is the best ROM formulation according to Chapter 2. We further discuss the treatment of displacements and rotations and decide to choose a combined treatment to introduce lROB. First, lROB based on k-means clustering is introduced. Subsequently, an adaption based on spherical k-means clustering is proposed. Finally, results are compared, hyper-reduction is applied, and a speedup for the crash box example is calculated.

3.1 Towards Parametric ROMs

3.1.1 Parameter Variations

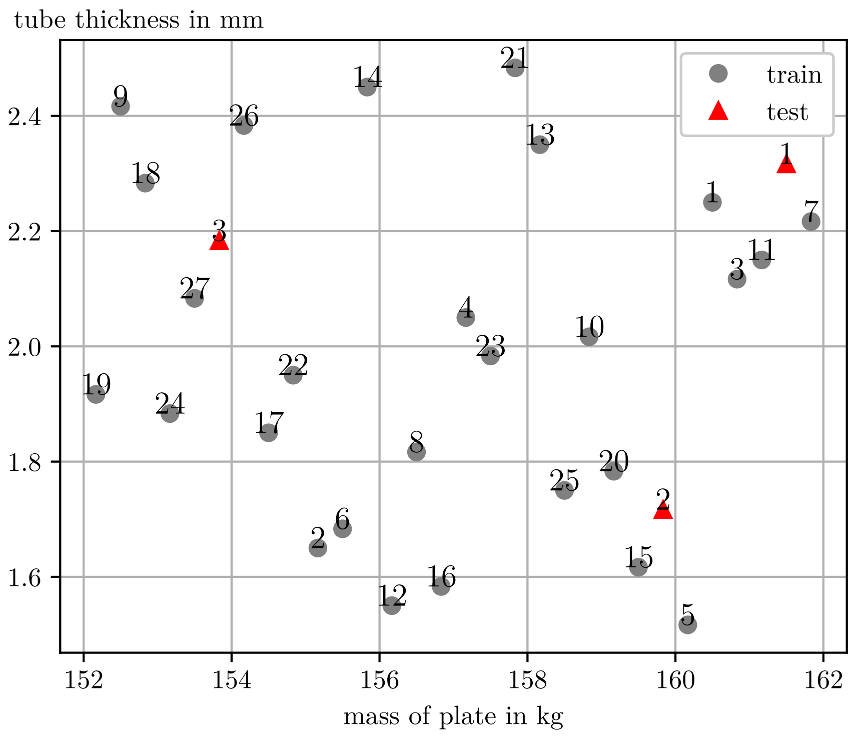


Figure 3.1 Sampling points for parameter variations obtained by Latin Hypercube sampling (LHS). The 30 points are divided into 27 training points (black dots) and 3 test points (red triangles).

Fig. 3.1 shows the 30 sampling points obtained by LHS. The varied parameters are the tube thickness and the mass of the plate. The tube thickness is sampled between 1.5 mm and 2.5 mm and the mass of the plate is sampled between 152 kg and 162 kg. The sampling points are divided into 27 training points and 3 test points. The training points are used to generate the training data and to train the model. The training data is generated with a non-modified version of LS-Dyna[1]. More precisely, pMOR training data is

[1]Version: smp d R12, Revision: R12-4012-g7c66dc5b4e

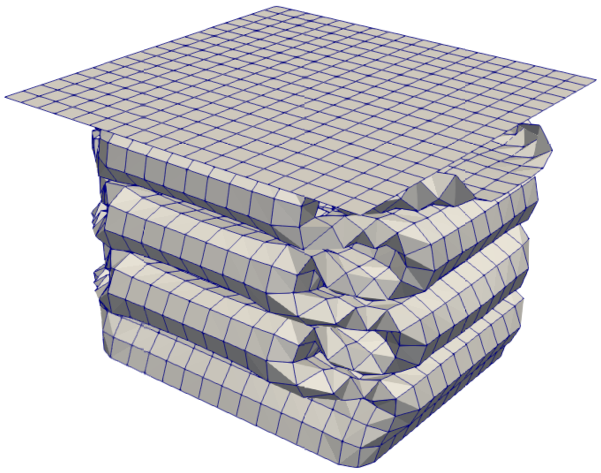


Figure 3.2 Highly deforming crash box at t =35 ms.

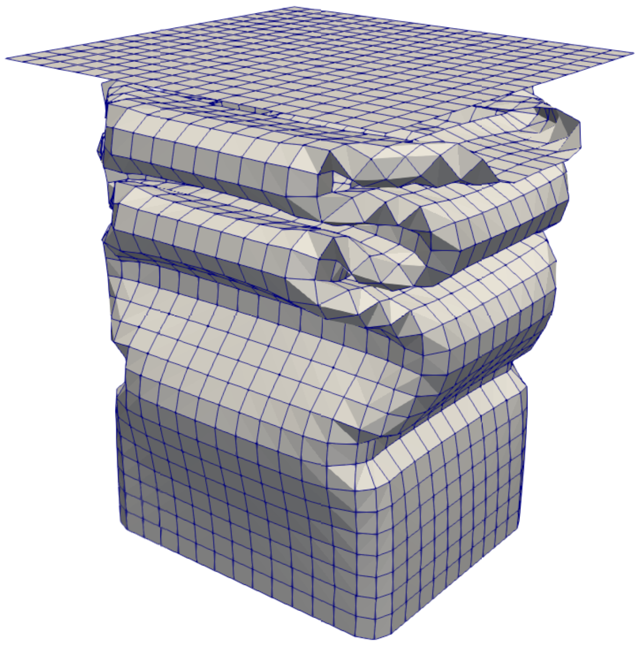


Figure 3.3 Crash box with low total deformation at t =35 ms.

used to create the ROB, or later the clusters and IROBs. The test points are chosen to represent a crash box with a high total deformation and a crash box with a low total deformation. The high deformation case corresponds to the test point with a low wall thickness and a high mass of the plate. The low deformation case corresponds to a crash box with high wall thickness and a low plate mass. The total deformation can be seen in Fig. 3.2 and Fig. 3.3. The parameters depicted in Fig. 3.1 are summarized in Tab. 3.1. The highly deforming test case depicted in Fig. 3.2 corresponds to test simulation number 2 in Table 3.1, and the low deforming test case shown in Fig. 3.3 corresponds to test simulation number 3 in Table 3.1. As test simulation number 1 shows a total deformation between test cases 2 and 3, it is not considered in the following evaluations. In addition, it is mentioned that the highly deforming test case usually poses the more difficult scenario. This observation is empirical and the author believes it is due to the cumulative nature of the error during a longer deformation path. The training data is collected and assembled to the global snapshot matrix $\mathbf{U}_{\text{glob}} \in \mathbb{R}^{n \times N_p N_s}$, which concatenates the temporal training data of all parameter variations:

$$\mathbf{U}_{\text{glob}} = \left[\mathbf{U}_1, \quad \mathbf{U}_2, \quad \dots, \quad \mathbf{U}_{N_p}\right], \tag{3.1}$$

where $\mathbf{U}_i \in \mathbb{R}^{n \times N_s}$ is the snapshot matrix of the simulation corresponding to training parameter i and N_p is the total number of training parameters. Each snapshot matrix $\mathbf{U}_i$ for the parametric case is equivalently defined as the snapshot matrix for the reproductive case eq. (2.68):

$$\mathbf{U}_i = \left[u_i^{t_1}, \quad u_i^{t_2}, \quad \dots, \quad u_i^{t_{N_s}}\right], \tag{3.2}$$

where $u_i^{t_j} \in \mathbb{R}^n$ is the solution snapshot of training simulation i at time t_j. Next, the global snapshot matrix is used to compute a ROB.

3.1.2 Treatment of DoF Types

So far, displacements and rotations have been treated separately. However, a combined treatment that uses correlations between the different DoF types can also be used. Therefore, the following two ways to reduce the system state are discussed in detail. First, we look at the separated treatment, which is the

Table 3.1 Numeric values of sampling points.

Training Simulation	Mass of the plate kg	Tube thickness mm
1	160.500	2.250
2	155.167	1.650
3	160.833	2.117
4	157.167	2.050
5	160.167	1.517
6	155.500	1.683
7	161.833	2.217
8	156.500	1.817
9	152.500	2.417
10	158.833	2.017
11	161.167	2.150
12	156.167	1.550
13	158.167	2.350
14	155.833	2.450
15	159.500	1.617
16	156.833	1.583
17	154.500	1.850
18	152.833	2.283
19	152.167	1.917
20	159.167	1.783
21	157.833	2.483
22	154.833	1.950
23	157.500	1.983
24	153.167	1.883
25	158.500	1.750
26	154.167	2.383
27	153.500	2.083
Test Simulation	**Mass of the plate kg**	**Tube thickness mm**
1	161.500	2.317
2	159.833	1.717
3	153.833	2.183

previously used method. As shown in Fig. 3.4, a separate basis is created for displacements and rotations. The portion of the high-dimensional state vector containing displacement DoF x_{disp} is approximated as:

$$x_{\mathrm{disp}} = \mathbf{\Phi}_{\mathrm{disp}}\hat{x}_{\mathrm{disp}}, \tag{3.3}$$

and the portion containing rotations DoF x_{rot} is approximated as:

$$x_{\mathrm{rot}} = \mathbf{\Phi}_{\mathrm{rot}}\hat{x}_{\mathrm{rot}}. \tag{3.4}$$

The reduced state vector for displacements is $\hat{x}_{\mathrm{disp}} \in \mathbb{R}^{k_{\mathrm{disp}}}$, where k_{disp} is the reduced dimension corresponding to displacements and $\hat{x}_{\mathrm{rot}} \in \mathbb{R}^{k_{\mathrm{rot}}}$ is the reduced state vector for rotations with the reduced dimension k_{rot}. The total dimension of the ROM is therefore:

$$k = k_{\mathrm{disp}} + k_{\mathrm{rot}}. \tag{3.5}$$

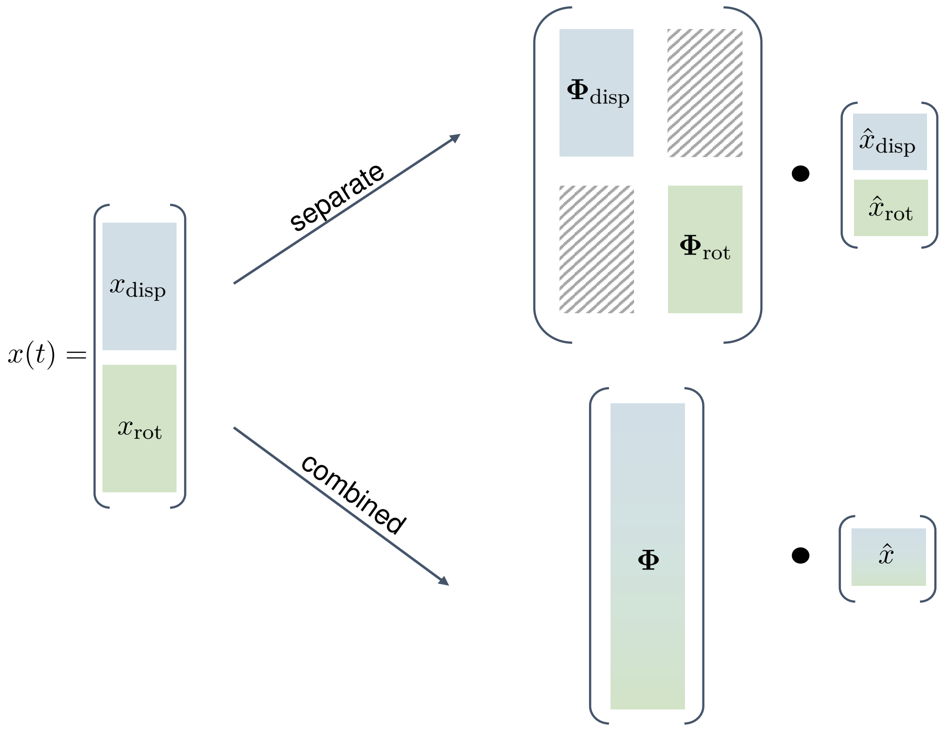


Figure 3.4 Two approaches to reduce displacements and rotations.

A combined treatment, as shown in the second row of Fig. 3.4, simply concatenates both types of DoF and approximates the entire state x as:

$$x = \mathbf{\Phi}\hat{x}, \tag{3.6}$$

where the dimension of the reduced state $\hat{x}$ is k. The combined treatment increases the size of the dimensionality reduction problem. However, the resulting ROM is smaller as correlations between displacements and rotations are considered and summarized to one reduced state. When displacements and rotations are separated, the size of the dimensionality reduction problems is reduced. However, the resulting ROM is usually higher-dimensional than the combined ROM. It was not observed in this thesis that displacements and rotations are fully uncorrelated, such that the sum of the separate bases is smaller than the combined basis. In addition, more hyper-parameters need to be adjusted. Compared to the combined ROM, two reduced dimensions, two hyper-reduction tolerances, and later, when using local reduced-bases, two numbers of clusters must be chosen. Which strategy delivers more accurate results will be assessed in the following. Furthermore, all evaluations aim to show the necessity of local ROMs.

3.1.3 Offline Accuracy

Error Evaluation

First, the approximation error eq. (2.72) is investigated for an increasing number of dimensions k for different scenarios. Fig. 3.5 depicts the approximation error of the reduced bases for the highly deforming test case, the low deformation test case, and a global basis for all training simulations. In each scenario, a separate and a combined reduced basis is calculated. In all cases, the error decreases with increasing dimensions. The error decays fastest for displacement bases and is the lowest for the single-parameter scenarios. The low deformation displacement basis is the most accurate basis for low dimensions. For larger dimensions (> 150), the accuracies of the single-parameter displacement bases converge. Reduced bases for rotations approximate the data less accurately than those for the displacements. The least accurate basis is the global basis for rotations. In general, global approximations are less accurate than single-parameter approximations. This is due to the increase in variance in the data set, which makes it harder for the dimensionality reduction to identify a low-rank structure in the data. The combined basis is less accurate than the displacement basis but more accurate than the respective rotation basis in all scenarios. We conclude that a combined treatment is preferred from an offline error perspective. Also, the error is higher for the global/parametric basis compared to the single-parameter cases. To achieve the same error in the parametric case, the ROB must use more dimensions, which counteracts the computational speedup. Finally, it is important to note that the discussed accuracy measure does not guarantee an accurate ROM. Hence, the online error is assessed additionally.

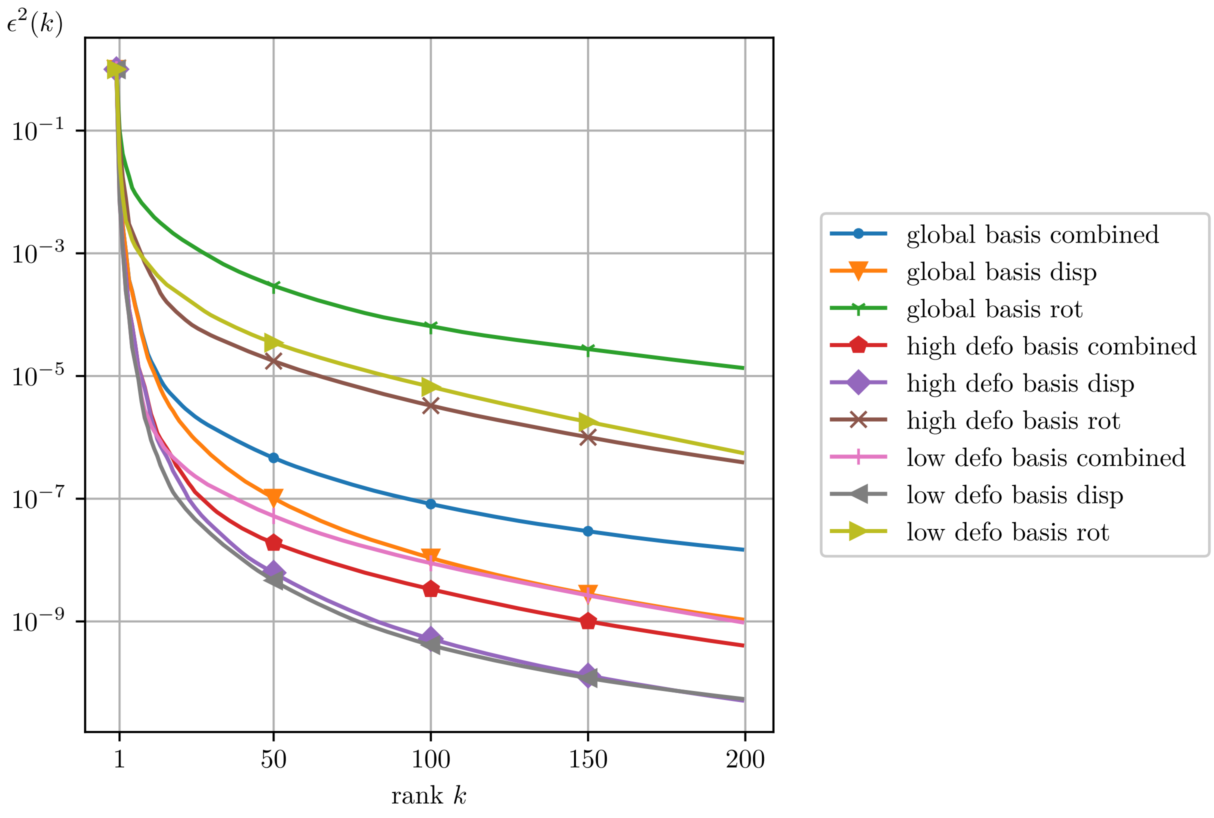


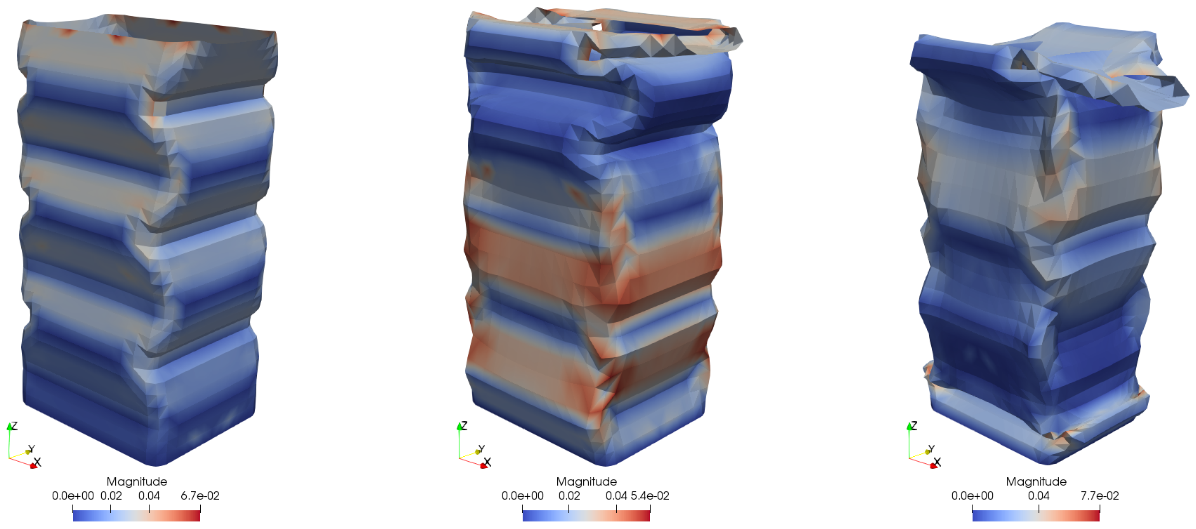
Figure 3.5 Approximation error $\epsilon^2(k)$ depending on the number of modes k to reconstruct the training data.

Visualization of Basis Vectors

The basis vectors, which compose the ROB, can be visualized. The displacement basis $\mathbf{\Phi}_{\text{disp}}$ or the part of the combined basis $\mathbf{\Phi}$ associated with the displacement DoF can be viewed as a displacement field. Figs. 3.6 - 3.9 show the deformation mode as deformed geometry in each case. The scalar coloring represents the magnitude of the rotation mode. Let $u \in \mathbb{R}^n$ be an arbitrary deformation state. Then, the state can be expressed in terms of reduced quantities (eq.(2.23)). The state u is a superposition of weighted mode contributions $u_i \in \mathbb{R}^n$. The contribution of the first three dominant modes is therefore visualized in the following plots. The deformation state is calculated as follows:

$$u_i = \alpha_i \phi_i \text{ for } i = 1, 2, 3. \tag{3.7}$$

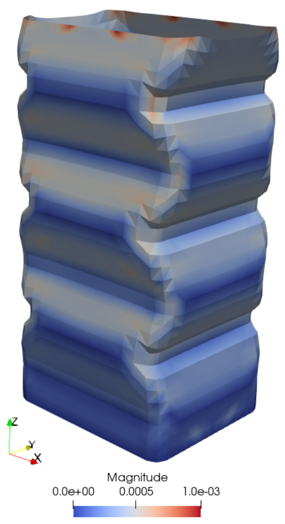
The scaling factor $\alpha_i = 1000$ is chosen to ensure good visualization of the displacement modes. For the rotation modes, no scaling is applied $\alpha_i = 1$. Fig. 3.6 visualizes the first three modes of the ROB for the high deformation test case. The first mode represents the most dominant buckling shape. The higher the mode number, the higher the frequencies in the spatial deformations. Fig. 3.7 visualizes the deformation modes for the high deformation test case calculated in a combined POD. While the deformation shapes are similar, the rotation modes differ for the second and third modes. Especially top and bottom regions behave



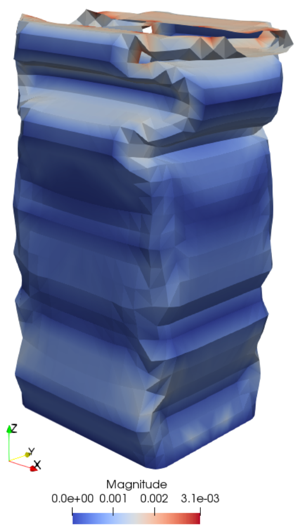
(a) ϕ_1 **(b)** ϕ_2 **(c)** ϕ_3

Figure 3.6 First three basis vectors of a POD-based ROM with a separate basis for displacements and rotations, trained using the highly deforming test simulation.

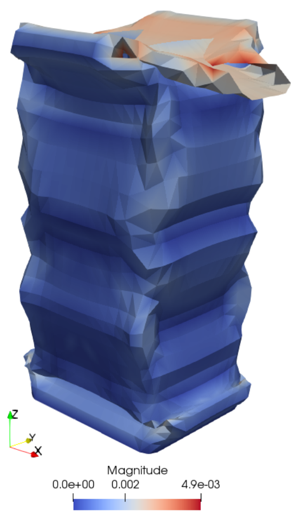
differently. The main deformation modes are similar in the parametric case for the separate and combined POD. Also, in the parametric case, mainly modes for rotations differ. As in Fig. 3.5 depicted, the first modes account for a large portion of the variance in the data set and cause a fast error decay. However, the more simulation data must be approximated at once, the more modes must be used to accurately reconstruct the solution. Especially in the parametric case, this is seen in the slow error decay. Already mode 5 (Fig. 3.10) cannot be related to a "visual" deformation shape anymore; however, it is needed to fully reconstruct a specific solution via linear superposition. In the parametric case, many different deformations can occur during the simulation, which is why many modes need to be retained in the ROB.



(a) ϕ_1

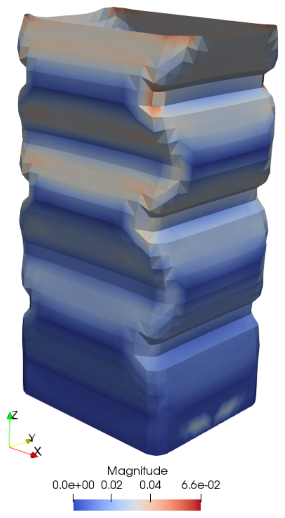


(b) ϕ_2

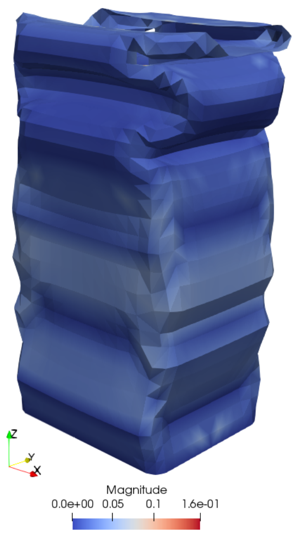


(c) ϕ_3

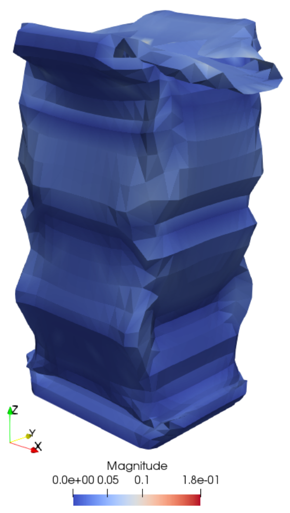
Figure 3.7 First three basis vectors of a POD-based ROM with combined reduction trained using the highly deforming test simulation.



(a) ϕ_1

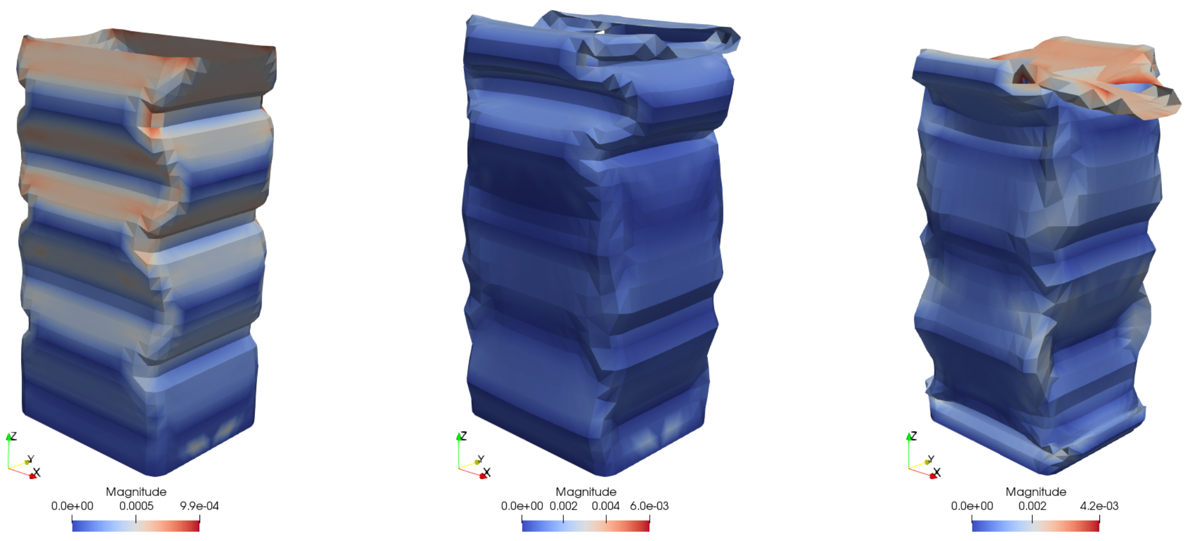


(b) ϕ_2



(c) ϕ_3

Figure 3.8 First three basis vectors of a POD-based ROM with a separate basis for displacements and rotations, trained using all training simulations.



(a) ϕ_1 **(b)** ϕ_2 **(c)** ϕ_3

Figure 3.9 First three basis vectors of a POD-based ROM with combined reduction, trained using all training simulations.

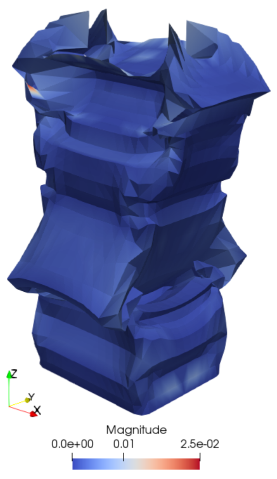


Figure 3.10 Fifth deformation mode ϕ_5 of a POD-based ROM.

3.1.4 Online Accuracy

Next, we assess the resulting online error of the ROM for an increasing number of dimensions k. Two cases are considered. The first case is a reproductive example of the highly deforming test case. That is, the ROB is computed based on the snapshots of the same simulation, which is afterward re-simulated. The second example uses a global basis calculated using all training simulations. Fig. 3.11 shows the displacement-based temporal error for the simulation time 35 ms for an increasing number of dimensions k in the reproductive case. The temporal error quickly decreases with increasing k, which can also be seen in Fig. 3.12. In Fig. 3.12 the temporal average error ϵ is shown for an increasing number of dimensions k for the combined ROM, a ROM where solely displacements are reduced, and a ROM where only rotations are reduced and displacements are reduced with $k = 200$ for the respective DoF to ensure nearly-zero error induced by displacements. A rapid error decrease is observable for all three ROMs. However, already at $k = 40$, the combined ROM yields the same error as the displacement-only ROM. Also, in the online phase, the combined ROM can reduce the slow error decay associated with the rotations. At approximately

$k = 90$, the combined ROM yields the same error as ROM with $k = 90$ for the rotation DoF and $k = 200$ for the displacement DoF, resulting in a 290-dimensional ROM.

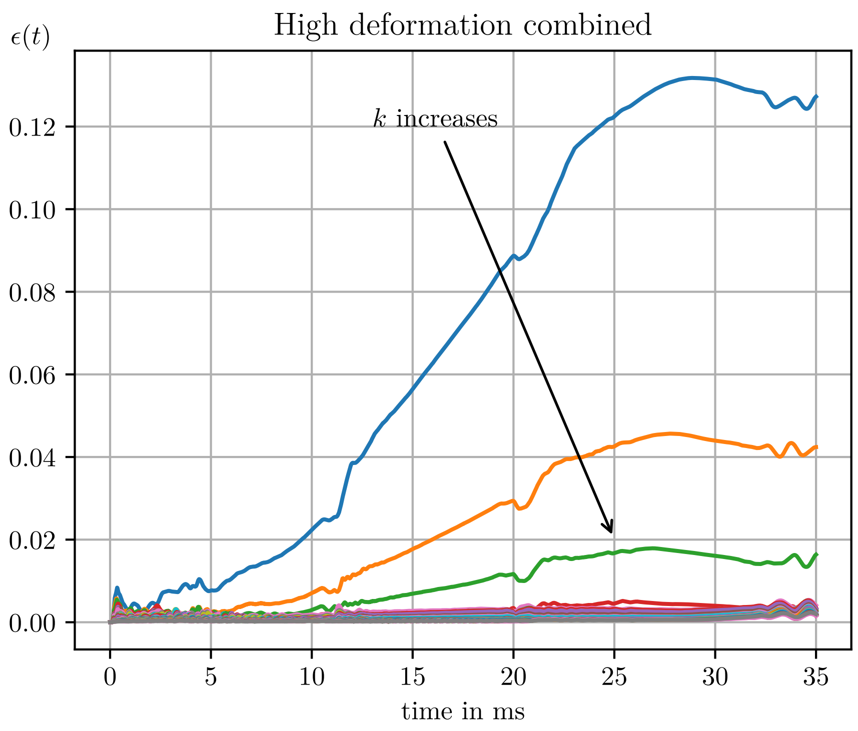


Figure 3.11 Time dependent error $\epsilon(t)$ for an increasing number of modes k using a combined basis in a reproductive example.

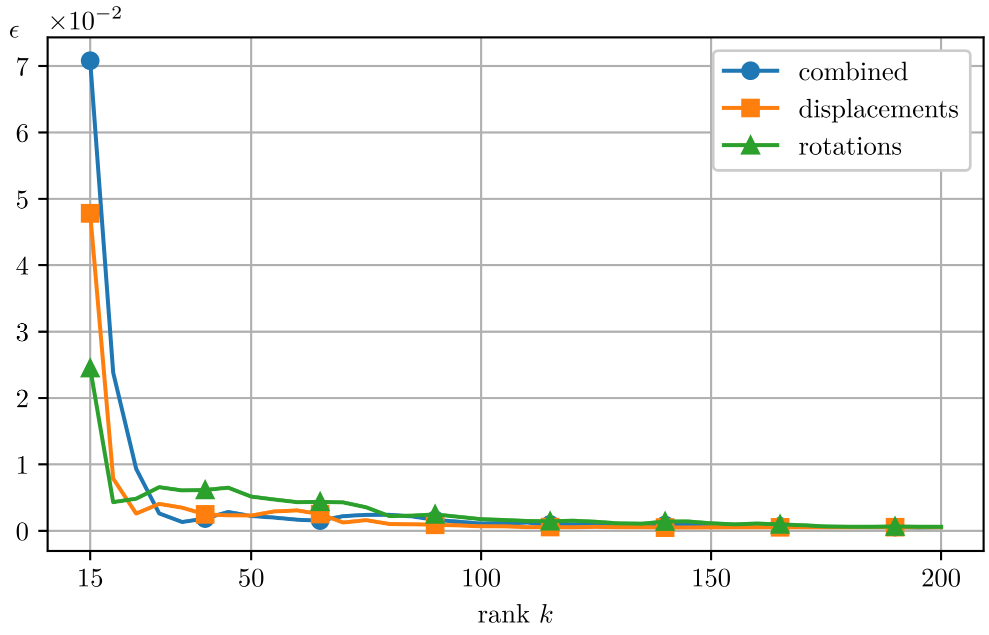


Figure 3.12 Average error ϵ for an increasing number of modes k using a ROM with separate bases, and a ROM with a combined basis, applied to a reproductive example.

Next, the global basis is used to simulate the highly deforming test case. Also, the temporal error is shown for an increasing number of dimensions k for the ROM with combined reduction. Compared to the reproductive example, the error starts at higher values, and a slower decrease in error with increasing dimensions can be observed in Fig. 3.13. The error behavior is also mirrored in Fig. 3.14, where the temporal average error is plotted against the approximation rank k for the combined ROM, the displacement ROM, and rotation ROM as defined in the reproductive example before. In analogy to the offline phase, the displacement-only ROM has the fastest error decay and the lowest error in total. The rotations exhibit a slower error decrease, especially for larger approximation ranks. However, the combined ROM ensures comparable error to the separate ROMs with approximately 65 dimensions.

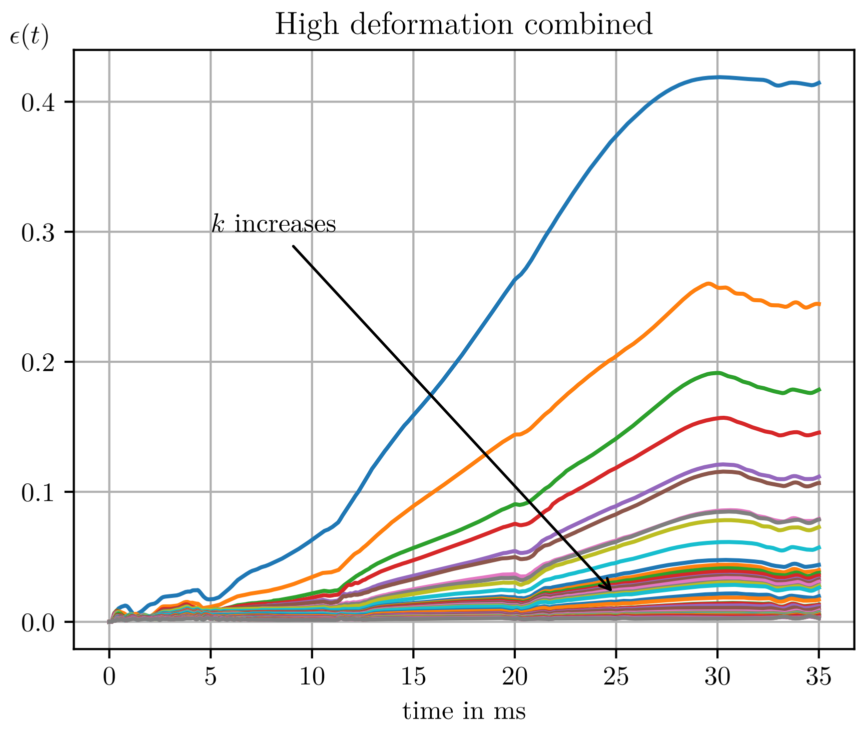


Figure 3.13 Time dependent error $\epsilon(t)$ for an increasing number of used modes k, using a combined basis with a global approach of the training data.

In conclusion, the combined reduction is beneficial, yielding low error at low dimensions. In addition, the creation of the ROM in the offline phase is simplified by reducing the number of adjustable parameters. The global basis has a slower error decay, which results in higher-dimensional ROMs. The ROM dimension influences the hyper-reduction. High-dimensional ROMs result in higher-dimensional reduced meshes, which is negative for computational speedup. Therefore, a different approach needs to be chosen for parametric pROMs. The approach must be suitable for ECSW hyper-reduction, as this is one primary mechanism to achieve speedup in nonlinear explicit simulations.

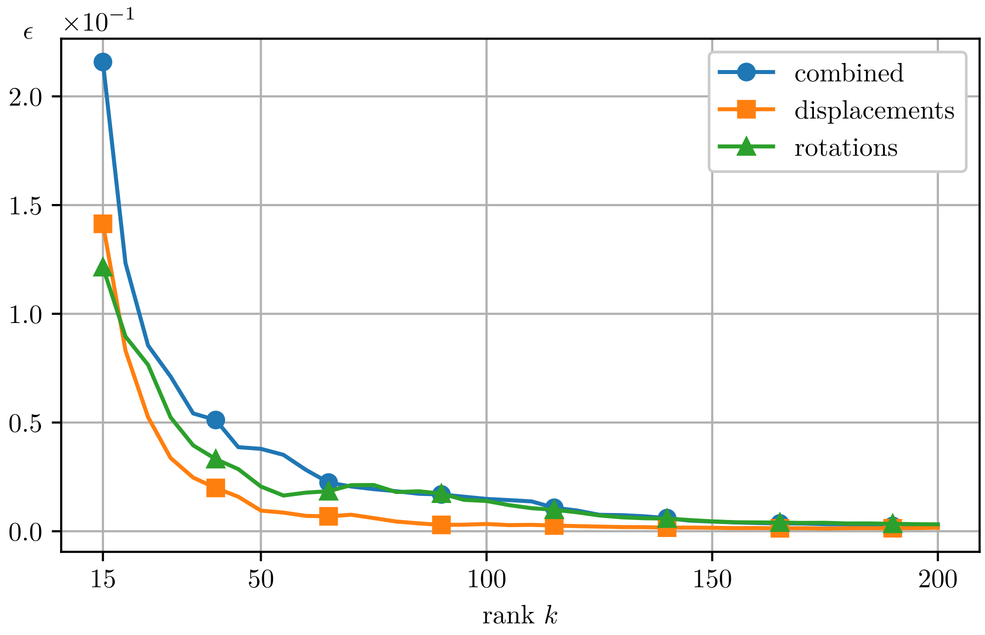


Figure 3.14 Average error ϵ for an increasing number of modes k, using a ROM with separate bases and a combined basis. The bases are calculated using all training data.

3.2 Local Reduced-order Bases

In this section, we introduce the necessary theory of lROB. Motivated by the observations made in the previous section about global ROMs, we choose lROB to realize projection-based ROMs for parametric systems in crash. A poor approximation quality of a global ROB for ROMs of nonlinear systems was already observed in literature [153]. A ROM based on a global basis often lacks robustness regarding parameter variations [81, 154–156]. Trajectories of parametric nonlinear systems can transverse or localize around different regions in state space, which implies that local ROB will better approximate the solution manifold at the considered point. In fact, the reproductive example in the previous section can be considered a local approximation. The reproductive ROM was accurate with fewer dimensions, while the global ROM required more dimensions to achieve the same accuracy.

The idea of local ROBs is illustrated in Fig. 3.15 for the 3-dimensional case. The dashed line is a one-dimensional manifold embedded in three-dimensional space. Further, the one-dimensional manifold also lives in a two-dimensional plane. On the left hand side, a global one-dimensional basis $\mathbf{\Phi}$ is used to approximate the manifold. It can be seen that the orthogonal projection $\hat{x}$ of the state x onto the one-dimensional basis results in a significant approximation error. At least two dimensions must be used to accurately describe the original manifold without introducing significant error. On the right hand side of Fig. 3.15, the original manifold is approximated using two local subspaces $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$. In this simple example, it can be seen that using two local subspaces, the current state $x(t)$ can be described by the associated local reduced-order basis $\mathbf{\Phi}_i$ and reduced state $q^i(t)$. The resulting ROM is closer to the intrinsic dimension of the original manifold, which is one in this case. The local ROM will likely result in a more accurate and lower-dimensional ROM than the global ROM. Fig. 3.16 illustrates the global approach for the n-dimensional case. The ROB $\mathbf{\Phi}$ spans a hyperplane and approximates the manifold $\mathcal{M}$. The global ROB cannot ensure to provide a good approximation to the manifold without increasing the dimensionality of the ROB. Instead, the manifold can be divided into subregions $\mathbf{\Phi}_i$, as seen in Fig. 3.17. The local subspaces are tailored to the region and accurately describe the state $x(t)$, and the evolution $\dot{x}(t) = f(x(t))$ is captured without significant error during the projection step $\mathbf{\Phi}_i^T f$.

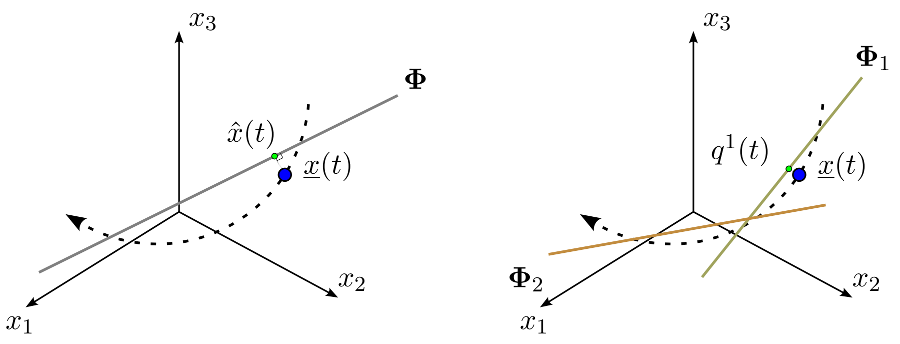


Figure 3.15 Global ROB approach (on the left side) versus local ROB approach (on the right side) in two dimensions.

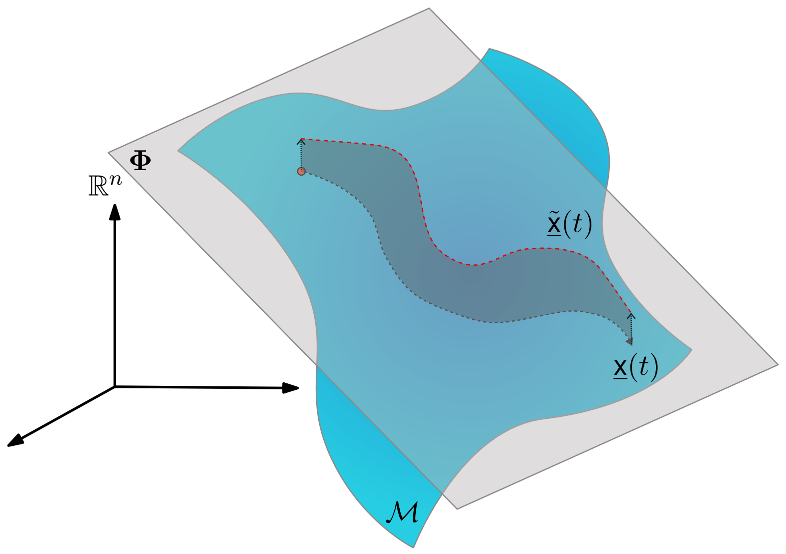

$\mathbf{\Phi}$
$\mathbb{R}^n$
$\tilde{\underline{\mathrm{x}}}(t)$
$\underline{\mathrm{x}}(t)$
$\mathcal{M}$

Figure 3.16 Global ROB describing a hyper plane.

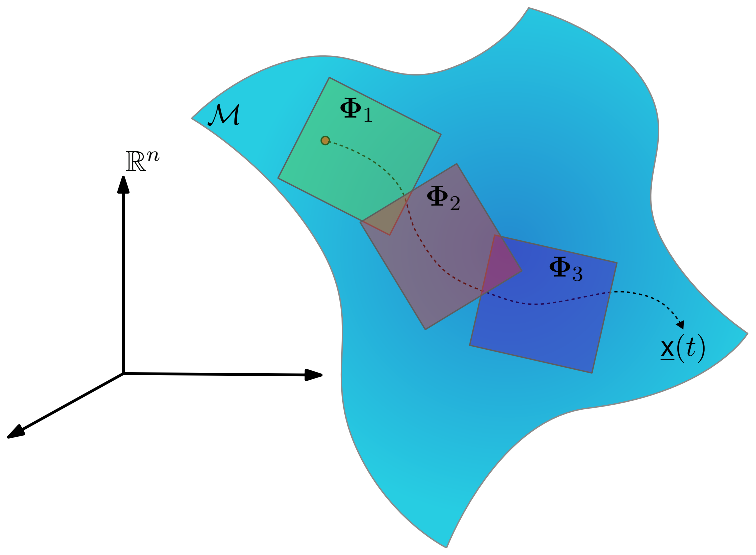


Figure 3.17 Local ROB approximating the solution manifold $\mathcal{M}$.

3.2.1 MOR Based on lROB

We formulate the lROB approach for second-order systems with explicit two-step time integration. Compared to the standard algorithm proposed by Amsallem et al. [140], an additional transformation step is required if the solution moves from one subdomain to another. We recall eq. (2.25b) to provide a basis for the introduction of IROB:

$$\mathbf{\Phi}^T\mathbf{M}(\mu)\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T f(t, \mathbf{\Phi}\hat{x}, \mathbf{\Phi}\dot{\hat{x}}, \mu) = 0.$$

The ROM predicts the evolution of the reduced state $\hat{x}$ using a ROB $\mathbf{\Phi}$, which remains unchanged during the entire simulation. Compared to the ROMs presented in Chapter 2, IROB approximates the increment Δx_{n+1} the instead of the states x_n, x_{n+1} itself. For notational reasons, the time index is written as a subscript for the remainder of this work. Thus, the increment can be written as:

$$\Delta x_{n+1} = x_{n+1} - x_n = \mathbf{\Phi}_n \Delta\hat{x}^{k_n}_{n+1}, \tag{3.8}$$

where $\mathbf{\Phi}_n$ is the reduced basis corresponding to the cluster chosen at t_n and $\Delta\hat{x}^{k_n}_{n+1}$ is the associated reduced increment with reduced dimension k_n. The velocities $\dot{x}_{n+\frac{1}{2}}$, $\dot{x}_{n-\frac{1}{2}}$ and reduced velocities $\dot{\hat{x}}^{k_n}_{n+\frac{1}{2}}$, $\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}$ can be related to the increments by dividing eq. (3.8) with the corresponding time step:

$$\dot{x}_{n+\frac{1}{2}} = \frac{x_{n+1} - x_n}{\Delta t_2} = \mathbf{\Phi}_n \frac{\Delta\hat{x}^{k_n}_{n+1}}{\Delta t_2} = \mathbf{\Phi}_n \dot{\hat{x}}^{k_n}_{n+\frac{1}{2}}, \tag{3.9a}$$

$$\dot{x}_{n-\frac{1}{2}} = \frac{x_n - x_{n-1}}{\Delta t_1} = \mathbf{\Phi}_{n-1} \frac{\Delta\hat{x}^{k_{n-1}}_{n}}{\Delta t_1} = \mathbf{\Phi}_{n-1} \dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}. \tag{3.9b}$$

Next, the governing equation eq. (2.16) is considered at time t_n and the temporal discretization of the acceleration eq. (2.18) is inserted:

$$\mathbf{M}\Delta t_x^{-1}(\dot{x}_{n+\frac{1}{2}} - \dot{x}_{n-\frac{1}{2}}) + f_n = 0. \tag{3.10}$$

The approximation of the velocities eq. (3.9) can now be inserted in eq. (3.10), which forms the high-dimensional residual $r_n(\dot{\hat{x}}^{k_n}_{n+\frac{1}{2}})$:

$$r_n(\dot{\hat{x}}^{k_n}_{n+\frac{1}{2}}) = \Delta t_x^{-1}\mathbf{M}(\mathbf{\Phi}_n \dot{\hat{x}}^{k_n}_{n+\frac{1}{2}} - \mathbf{\Phi}_{n-1}\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}) + \tilde{f}_n, \tag{3.11}$$

where $\Delta t_x = \frac{1}{2}(\Delta t_1 + \Delta t_2)$ and $\tilde{f}_n$ is the force vector evaluated at the ROM solution $\tilde{x}_n, \dot{\tilde{x}}_{n-\frac{1}{2}}$. We enforce the residual to be orthogonal to the current basis $\mathbf{\Phi}_n$:

$$\mathbf{\Phi}_n^T r_n \overset{!}{=} 0. \tag{3.12}$$

The resulting ROM is:

$$\mathbf{\Phi}_n^T\mathbf{M}(\mathbf{\Phi}_n \dot{\hat{x}}^{k_n}_{n+\frac{1}{2}} - \mathbf{\Phi}_{n-1}\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}) + \Delta t_x \mathbf{\Phi}_n^T \tilde{f}_n = 0. \tag{3.13}$$

Eq. (3.13) still scales with the dimension of the HDM. Not only does the projection of the nonlinear force vector $\mathbf{\Phi}_n^T \tilde{f}_n$ involve high dimensions, but also the multiplication of the ROB with the mass matrix $\mathbf{M}\mathbf{\Phi}_n$, $\mathbf{M}\mathbf{\Phi}_{n-1}$. The reduction of the nonlinear force term is achieved using hyper-reduction, which will be discussed in Subsection 3.2.4. Addressing the multiplication of the ROB with the mass matrix, two approaches are possible to avoid high-dimensional operations. The first approach is performed offline. All combinations of reduced mass matrices are computed:

$$\tilde{\mathbf{M}}_{ij} = \mathbf{\Phi}_i^T \mathbf{M} \mathbf{\Phi}_j \in \mathbb{R}^{k_i \times k_j} \quad i, j \in [1, n_c], \tag{3.14}$$

where n_c is the number subdomains, also called clusters. However, we follow a different approach and transform the velocity:

$$\dot{x}_{n-\frac{1}{2}} = \mathbf{\Phi}_{n-1}\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}} \approx \mathbf{\Phi}_n \underbrace{\mathbf{\Phi}_n^T \mathbf{\Phi}_{n-1}\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}}_{\dot{\hat{x}}^{k_n}_{n-\frac{1}{2}}}. \tag{3.15}$$

The reduced velocity at time $t = t_{n-\frac{1}{2}}$ is now transformed to the current cluster at $t = t_n$. The transformation is assumed to introduce no significant error as the cluster borders are assumed to be close to each other. In addition, a parameter will be introduced that controls the overlap of the clusters to ensure a smooth transition between different clusters. For the ROM, two cases must be distinguished:

1. The solution stays in the cluster and no cluster change appears. That is, $\mathbf{\Phi}_n = \mathbf{\Phi}_{n-1}$ and $k_n = k_{n-1}$. Due to the orthonormality of the ROB
$$\mathbf{\Phi}_n^T \mathbf{\Phi}_n = \mathbf{I}, \tag{3.16}$$
eq. (3.13) transforms to the standard ROM regarding the currently chosen basis Φ_n:
$$\underbrace{\mathbf{\Phi}_n^T \mathbf{M} \mathbf{\Phi}_n}_{\tilde{\mathbf{M}}_n}(\dot{\hat{x}}^{k_n}_{n+\frac{1}{2}} - \dot{\hat{x}}^{k_n}_{n-\frac{1}{2}}) + \Delta t_x \mathbf{\Phi}_n^T \tilde{f}_n = 0. \tag{3.17}$$
The new velocity is thus obtained as:
$$\dot{\hat{x}}^{k_n}_{n+\frac{1}{2}} = \dot{\hat{x}}^{k_n}_{n-\frac{1}{2}} + \Delta t_x \tilde{\mathbf{M}}_n^{-1} \mathbf{\Phi}_n^T \tilde{f}_n = 0. \tag{3.18}$$

2. The solution transverses from one subregion to another. The ROBs $\mathbf{\Phi}_n \neq \mathbf{\Phi}_{n-1}$ and the associated dimensions $k_n \neq k_{n-1}$ differ. Therefore, the previous reduced velocity $\dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}$ must be transformed to the current cluster first:
$$\dot{\hat{x}}^{k_n}_{n-\frac{1}{2}} = \mathbf{\Phi}_n^T \mathbf{\Phi}_{n-1} \dot{\hat{x}}^{k_{n-1}}_{n-\frac{1}{2}}. \tag{3.19}$$
Once the velocity is transformed, the new reduced velocity is obtained as in the first case using eq. (3.18).

Once the new velocity is known, the displacement increment can also be calculated:
$$\Delta \hat{x}^{k_n}_{n+1} = \Delta t_2 \mathbf{\Phi}_n \dot{\hat{x}}^{k_n}_{n+\frac{1}{2}} \tag{3.20}$$
and
$$x_{n+1} = x_n + \mathbf{\Phi}_n \Delta \hat{x}^{k_n}_{n+1}. \tag{3.21}$$
At this point, the ROM formulation is complete. Now, it is still to be determined how the subdomains are formed and how it can be determined quickly in which subdomain the model is located. These tasks are addressed in the next sections for the k-means-based ROM, as proposed by [140], and for the new ROM based on spherical k-means.

3.2.2 K-means Clustering-based IROB

Clustering and Reduced Basis Generation

Now that the model has been derived, the solution snapshots must be divided into subregions/clusters. The difficulty of this task is to choose a proper clustering metric. In the first part of this chapter, we recall IROB based on k-means clustering [90, 157] and subsequently apply it to a typical crash problem. k-means clustering minimizes the squared L^2 distance of the samples/snapshots to the cluster centers $m_j \in \mathbb{R}^n$:
$$\{\pi_j^*\}_{j=1}^{n_C} = \arg \min_{\{\pi_j\}_{j=1}^{n_C}} \sum_{j=1}^{n_C} \sum_{u \in \pi_j} \|u - m_j\|^2 , \tag{3.22}$$
where π_j, $j = 1, \dots, n_c$ are the n_c subsets. The union of all subsets contains all $N_p * N_s$ snapshots stored in the global snapshot matrix $\mathbf{U}_{\text{glob}}$:
$$\bigcup_{j=1}^{n_C} \pi_j = \{u_1^{t_1}, \dots, u_{N_p}^{t_{N_s}}\}, \tag{3.23}$$
and the subsets are disjoint:
$$\bigcap_{j=1}^{n_C} \pi_j = \emptyset. \tag{3.24}$$

The cluster centers are the average of the snapshots contained in one cluster:

$$m_j = \frac{1}{|\pi_j|} \sum_{u \in \pi_j} u. \tag{3.25}$$

In addition, they can be arranged in a matrix $\mathbf{m} \in \mathbb{R}^{n \times n_c}$:

$$\mathbf{m} = \begin{bmatrix} m_1, & m_2, & \dots, & m_{n_c} \end{bmatrix}. \tag{3.26}$$

Optimization problem eq. (3.22) can be solved iteratively by Lloyd's algorithm [157]: The snapshots are

Algorithm 2 Lloyd's algorithm to partition a dataset into n_c disjoint subsets π_j, $j = 1, \dots, n_c$ regarding a specified distance metric $S(x, y)$. In k-means $S = \|x - y\|_2^2$.

Input: Tolerance ϵ, data matrix $\mathbf{U}_{\text{glob}}$ containing the solution snapshots, predefined number of clusters n_c, maximum iterations n_{max}.

Output:n_c subsets π_j containing the assigned solution snapshots, the cluster centroids m_j for $j = 1, \dots, n_c$.

```
1: Randomly choose $n_c$ snapshots as initial cluster centers $m_j^{(0)}$, for $j = 1, \dots, n_c$
2: $t = 0$
3: $\epsilon^{(0)} = \epsilon + 1$
4: **while** $t < n_{max}$ and $\epsilon^{(t)} > \epsilon$ **do**
5:     **for** $j \in \{1, \dots, n_c\}$ **do**
6:         $$\pi_i^{(t)} = \left\{ u_r^{t_s} \;\middle|\; \left\| u_r^{t_s} - m_i^{(t)} \right\|^2 \le \left\| u_r^{t_s} - m_j^{(t)} \right\|^2, \text{ for } s = 1, \dots, N_s, \, r = 1, \dots, N_p \right\} \tag{3.27}$$
7:     **end for**
8:     **for** $j \in \{1, \dots, n_c\}$ **do**
9:         $m_j^{(t+1)} = \frac{1}{|\pi_j^{(t)}|} \sum_{u \in \pi_j^{(t)}} u$
10:    **end for**
11:    $\epsilon^{(t+1)} = \left\| \mathbf{m}^{(t+1)} - \mathbf{m}^{(t)} \right\|_F$
12:    $t = t + 1$
13: **end while**
14: $\pi_j \leftarrow \pi_j^{(t)}$, for $j = 1, \dots, n_c$
15: $m_j \leftarrow m_j^{(t)}$, for $j = 1, \dots, n_c$
16: **return** $\pi_j$, $m_j$
```

assigned to clusters, and the centroids of the clusters are calculated. An overlap between the subregions is introduced to ensure a smooth transition of the solution from one cluster to the other. The clusters are, therefore, no longer distinct. The overlap is realized by adding the r closest snapshots regarding the specified metric to each cluster. The final assignment of the snapshots is termed:

$$\pi_j^{\text{fin}}, \quad \text{for } j = 1, \dots, n_c \tag{3.28}$$

The final sets are used to assemble the local snapshot matrices $\mathbf{X}_j$:

$$\mathbf{X}_j = \begin{bmatrix} u_1, u_i, \dots, u_{n_j} \end{bmatrix} \in \mathbb{R}^{n \times n_j}, \quad u_i \in \pi_j^{\text{fin}} \tag{3.29}$$

with

$$n_j = |\pi_j^{\text{fin}}|. \tag{3.30}$$

SVD is applied to each local snapshot matrix to compute the associated ROB $\mathbf{\Phi}_j$ with a reduced dimension k_j.

Cluster Identification

During the online phase of the simulation, it is necessary to quickly determine the distance of the solution to the centroids of the clusters. The shortest distance defines the current cluster and the chosen ROB. Without further modification, the distance calculation in state space scales with the dimension of the HDM. Therefore, in analogy to [140], we avoid the computation of the distance directly and compare the distances among each other. We define the sign function:

$$z_{i-1}^{m,p} = S(u_{i-1}, m_m) - S(u_{i-1}, m_p) = \|u_{i-1} - m_m\|^2 - \|u_{i-1} - m_p\|^2 . \tag{3.31}$$

By a change in sign, the need to change cluster is indicated. It is to note that the function can be similarly defined using the coordinates x by translating the centroids with the initial configuration x_0:

$$S(u_{i-1}, m_p) = \|u_{i-1} - m_p\|^2 = \|x_{i-1} - x_0 - m_p\|^2 = \|x_{i-1} - (m_p + x_0)\|^2 = S(x_{i-1}, m_p + x_0), \tag{3.32}$$

and therefore expressed in terms of coordinates:

$$z_{i-1}^{m,p} = S(x_{i-1}, m_m + x_0) - S(x_{i-1}, m_p + x_0). \tag{3.33}$$

Next, by repeatedly using the IROB definition eq. (3.8), the state x_{i-1} can be expressed solely in terms of reduced increments:

$$x_{i-1} = x_0 + \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r, \tag{3.34}$$

where

$$q_i^r = \sum_{\substack{1 \le m \le i, \\ m \in \mathcal{I}_r}} \Delta \hat{x}_m^{k_{m-1}} \in \mathbb{R}^{k_r}. \tag{3.35}$$

$\mathcal{I}_r$ is the index set of time steps, for which the solution lies in cluster k associated with basis $\mathbf{\Phi}_k$. If the initial condition x_0 is subtracted from both sides of eq. (3.34), the current displacement can be related to the sum of increments q_i^r:

$$u_{i-1} = \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r. \tag{3.36}$$

We insert eq. (3.36) in eq. (3.31) and obtain:

$$\begin{aligned} z_{i-1}^{m,p} &= \left\| \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r - m_m \right\|^2 - \left\| \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r - m_p \right\|^2 \\ &= \left\| \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r \right\|^2 - 2 \sum_{r=1}^{n_c} m_m^T \mathbf{\Phi}_r q_{i-1}^r + \|m_m\|^2 \\ &- \left\| \sum_{r=1}^{n_c} \mathbf{\Phi}_r q_{i-1}^r \right\|^2 + 2 \sum_{r=1}^{n_c} m_p^T \mathbf{\Phi}_r q_{i-1}^r - \|m_p\|^2 \\ &= \sum_{r=1}^{n_c} \underbrace{2(m_p^T - m_m^T)\mathbf{\Phi}_r}_{w_r^{m,pT}} q_{i-1}^r + \underbrace{\|m_m\|^2 - \|m_p\|^2}_{d^{m,p}} . \end{aligned} \tag{3.37}$$

We define the following quantities:

$$w_r^{m,p} = 2\mathbf{\Phi}_r^T (m_p - m_m) \in \mathbb{R}^{k_r}, \, \forall r \in [1, n_c], \, 1 \le m < p \le n_c, \tag{3.38}$$

and

$$d^{m,p} = \|m_m\|^2 - \|m_p\|^2 \in \mathbb{R}, \, 1 \le m < p \le n_c. \tag{3.39}$$

They are precomputable and can be used to relate the increment in the sign function of two consecutive time steps:

$$\Delta z_{i-1}^{m,p} = z_{i-1}^{m,p} - z_{i-2}^{m,p} = \sum_{r=1}^{n_c} w_r^{m,pT}(q_{i-1}^r - q_{i-2}^r). \tag{3.40}$$

Let r_{i-1} be the index of the local basis chosen at t_{i-1}. Taking a closer look at the difference and using eq. (3.35) yields:

$$q_{i-1}^r - q_{i-2}^r = \begin{cases} 0 & r \neq r_{i-1} \\ \Delta\hat{x}_{i-1}^{k_{i-2}} & r = r_{i-1} \end{cases}. \tag{3.41}$$

Eq. (3.41) states that the reduced state is not incremented for all clusters except for the currently used one. The difference in state variables is, therefore, zero for all clusters that are not currently in use. The sign function update eq. (3.40) therefore simplifies to:

$$\begin{aligned} \Delta z_{i-1}^{m,p} &= w_{r_{i-1}}^{m,pT} \Delta\hat{x}_{i-1}^{k_{i-2}} \\ \implies z_{i-1}^{m,p} &= z_{i-2}^{m,p} + w_{r_{i-1}}^{m,pT} \Delta\hat{x}_{i-1}^{k_{i-2}}. \end{aligned} \tag{3.42}$$

All the necessary building blocks are now available, so the online phase can be carried out. With the appropriate initial and boundary conditions, the forces and, subsequently, the accelerations are calculated and projected to the reduced space. The reduced accelerations are integrated in time and the reduced increment is obtained. The need to change the ROB is determined using the previously derived sign function. If no basis change is necessary, the solution is mapped to high-dimensional state again to compute the next force vector. When changing the cluster, the reduced velocity is also transformed.

3.2.3 Spherical Clustering-based IROB

In this section, we propose IROB based on spherical k-means clustering. We start by comparing the properties of crash systems with those of fluid mechanical problems since IROB is mainly used in the latter. Afterward, we explain spherical k-means clustering and derive a new distance (sign) function to quickly determine the current subregion in the online phase.
Classical IROB is based on k-means clustering and was introduced in the context of a fluid-structure-electric interaction. k-means clustering is based on Euclidean distances and not scale-invariant. Amsallem et al. [158] claimed that k-means clustering is suboptimal and proposed a clustering method based on the true projection error. Systems in crash often exhibit large deformations in one direction. The state space is mainly traversed in one direction, and localization around certain points does not occur. We also claim that, in this case, k-means clustering is suboptimal. However, we propose a different clustering metric more closely related to the Euclidean norm. A similar distance function can be derived, and less severe modifications of the original IROB framework are necessary.
We will revisit the main differences between other systems and crash systems below to support using a different clustering metric. Except for differences such as complex materials involving path-dependencies and contact algorithms, systems in crash and impact differ from systems describing fluid problems and some linear/nonlinear elastodynamics problems in the following further points:

- Fluid problems are usually described by Eulerian coordinates, while a Lagrangian description is chosen for crash and impact problems.
- Velocities describe fluids whereas nodal coordinates x or displacements $u = x - x_0$ describe solids. In solid mechanics, the trajectory of a single material point is tracked, whereas the velocity of the particles passing a fixed point in space is considered.
- In fluid problems, the velocity vector changes rather in direction than in magnitude in the course of the simulation.
- Compared to elastodynamic problems, crash solutions are not oscillatory.

- Solutions of crash problems usually show a nearly monotonic increase in magnitude. The models deform in one direction until the kinetic energy is consumed by deformation work.

It is to be noted that the previous claims are general and need to be checked in each case. In the crash box example, the temporal evolution of the solution corresponds to an increase in magnitude of the solution vector. Assuming the trajectories of different parameter combinations are close, the L^2 distance-based k-means clustering will mainly separate the solution along the temporal axis. Snapshots of the beginning of the solution will never be assigned to the same cluster as snapshots from the end of the simulation, even though their deformation direction could be similar. In reduced-order modeling, the ROB dictates the space of admissible deformations. The information is stored in the individual basis vectors of the ROB, where the length of the vectors is irrelevant regarding the subspace they span. This leads us to the claim that the magnitude of the solution snapshots should have no influence and only the direction of deformation is relevant. A similar problem appears in document processing, where documents of differing lengths are clustered. Dhillon et al. [159] proposed spherical k-means clustering, which uses normalization to mitigate the effect of different document lengths [160]. Normalizing the snapshots is equivalent to spherically projecting them onto a unit sphere. Calculating the cosine between two normalized vectors relates to the computation of the L^2, which is why minor modifications of the k-means algorithm are sufficient, although not the most efficient.

Clustering and Reduced Basis Generation

The key idea is to use a different clustering metric. We adopt the definition of the cosine dissimilarity and use it for clustering. The cosine dissimilarity is defined as:

$$D(x, y) = 1 - \cos(\phi) = 1 - \frac{x \cdot y}{\|x\| \, \|y\|}. \tag{3.43}$$

Eq. (3.43) measures the dissimilarity between two vectors $x, y \in \mathbb{R}^n$ and defines the angle ϕ between the two vectors. For two normalized vectors $a, b \in \mathbb{R}^n$, $\|a\| = 1$, $\|b\| = 1$, the cosine dissimilarity can be related to the Euclidian distance:

$$\|a - b\|^2 = (a - b)^T \cdot (a - b) = \|a\| + \|b\| - 2a^T \cdot b = 2(1 - \cos(\phi)) = 2D(a, b). \tag{3.44}$$

This relation is beneficial when modifying existing algorithms that solve the original k-means problem. For spherical k-means, we seek the solution of the following optimization problem:

$$\{\pi_j^*\}_{j=1}^{n_C} = \arg \min_{\{\pi_j\}_{j=1}^{n_C}} \sum_{j=1}^{n_C} \sum_{u \in \pi_j} D(u, c_j), \tag{3.45}$$

where $c_j \in \mathbb{R}^n$ denotes the normalized centroid of cluster π_j, which is defined as:

$$c_j = \frac{m_j}{\|m_j\|}. \tag{3.46}$$

The centroid is analogously defined as in k-means eq. (3.25).
During the experiments, we found that snapshots with a small magnitude are prone to false identification. These snapshots appear mainly at the beginning of the training simulations. To overcome the false identification, an initial cluster is defined in a preprocessing step. The initial cluster includes snapshots which are collected until a user-defined time t_{init}:

$$\pi_{\text{init}} = \{u_i : i \leq \frac{t_{\text{init}}}{\Delta t}\}. \tag{3.47}$$

The time interval between two consecutive snapshots is denoted by Δt. The snapshots from the initial cluster are removed from the snapshots fed to the clustering algorithm. Let U_{glob} be the set of all snapshots contained in the snapshot matrix $\mathbf{U}_{\text{glob}}$:

$$U_{\text{glob}} = \{u_1^{t_1}, \ldots, u_{N_p}^{t_{N_s}}\}. \tag{3.48}$$

Then, the final set of snapshots U is:

$$U = U_{\text{glob}} \setminus \pi_{\text{init}} \tag{3.49}$$

and the modified snapshot matrix $\mathbf{U}$ is:

$$\mathbf{U} = [u_1, u_i, \dots, u_{n_{\text{mod}}}] \in \mathbb{R}^{n \times n_{\text{mod}}}, \quad u_i \in U, \tag{3.50}$$

with

$$n_{\text{mod}} = N_s * N_p - |\pi^{\text{init}}| = |U|. \tag{3.51}$$

Optimization problem eq. (3.45) is solved using the Python implementation "spherecluster" from Jason Laska [161]. The standard implementation presented in Algorithm 2 is modified, such that the new centroid is normalized using eq. (3.46) after the centroids are updated. When the algorithm has converged or reached the maximum number of iterations, the closest snapshots are added to the clusters as in conventional IROB. However, the initial cluster is excluded. A fixed number of snapshots can be added to each cluster, or all snapshots in a certain distance can be added. In this work, we use a fixed number of snapshots r. In analogy to the k-means-based ROM, as soon as the snapshots are added, the final sets/clusters π_j^{fin} and π^{init} are determined. The final clusters are used to construct the cluster-specific snapshot matrices eq. (3.29). They are input for SVD, which yields the local ROBs.

Cluster Identification

As for the k-means-based ROM, a computationally cheap cluster identification must also be derived for the spherical k-means ROM. We again define a sign function, indicating the need for a cluster change by a changing sign. The sign function is defined as:

$$z_{i-1}^{m,p} = D(u_{i-1}, c_m) - D(u_{i-1}, c_p) = \frac{u_{i-1} \cdot c_p}{\|u_{i-1}\| \, \|c_p\|} - \frac{u_{i-1} \cdot c_m}{\|u_{i-1}\| \, \|c_m\|}. \tag{3.52}$$

Using the definition of the normalized centroids eq. (3.46) leads to:

$$z_{i-1}^{m,p} = \frac{u_{i-1} \cdot (c_p - c_m)}{\|x_{i-1}\|}. \tag{3.53}$$

First, we consider the squared denominator of eq. (3.53). We square the equation for notation purposes, but this does not impact computation time since we are dealing with the square root of a scalar. We express the state u_{i-1} by the previous state u_{i-2} and the increment $\Delta\hat{x}_{i-1}^{k_{i-2}}$:

$$\begin{aligned} \|u_{i-1}\|^2 = u_{i-1}^T u_{i-1} &= (x_{i-2} + \mathbf{\Phi}_{i-2}\Delta\hat{x}_{i-1}^{k_{i-2}})^T (x_{i-2} + \mathbf{\Phi}_{i-2}\Delta\hat{x}_{i-1}^{k_{i-2}}) \\ &= \|u_{i-2}\|^2 + \|\Delta\hat{x}_{i-1}^{k_{i-2}}\|^2 + 2(u_{i-2}^T \mathbf{\Phi}_{i-2}\Delta\hat{x}_{i-1}^{k_{i-2}}). \end{aligned} \tag{3.54}$$

The norm $\|u_{i-2}\|^2$ is known from the previous time step; however, the multiplication of u_{i-2} from left in the last term of eq. (3.54) still involves high dimensions. Therefore, we use the reduced representation eq. (3.34) of u_{i-2} and insert it in eq. (3.54):

$$\|u_{i-1}\|^2 = \|u_{i-2}\|^2 + \|\Delta\hat{x}_{i-1}^{k_{i-2}}\|^2 + 2\sum_{r=1}^{n_c} q_{i-2}^r \mathbf{\Phi}_r^T \mathbf{\Phi}_{i-2}\Delta\hat{x}_{i-1}^{k_{i-2}}. \tag{3.55}$$

The terms $\mathbf{\Phi}_r^T \mathbf{\Phi}_{i-2}$ can be precomputed. However, more importantly, the reduced state q_i^r is zero for all clusters not used until the current simulation time in the online phase, and summation can be skipped in these cases. Finally, we replace the numerator in eq. (3.53) by the previous state and the increment:

$$\begin{aligned}
z_{i-1}^{m,p} &= \frac{u_{i-2} \cdot (c_p - c_m)}{\|u_{i-1}\|} + \frac{(\mathbf{\Phi}_{i-2} \Delta \hat{x}_{i-1}^{k_{i-2}}) \cdot (c_p - c_m)}{\|u_{i-1}\|} \\
&= \frac{\|u_{i-2}\|}{\|u_{i-1}\|} z_{i-2}^{m,p} + \frac{(\mathbf{\Phi}_{i-2} \Delta \hat{x}_{i-1}^{k_{i-2}}) \cdot (c_p - c_m)}{\|u_{i-1}\|} \\
&= \frac{\|u_{i-2}\|}{\|u_{i-1}\|} z_{i-2}^{m,p} + \frac{\Delta \hat{x}_{i-1}^{k_{i-2}} \cdot \mathbf{\Phi}_{i-2}^T (c_p - c_m)}{\|u_{i-1}\|} \\
&= \frac{\|u_{i-2}\|}{\|u_{i-1}\|} z_{i-2}^{m,p} + \frac{\Delta \hat{x}_{i-1}^{k_{i-2}} \cdot w_{i-2}^{m,p}}{\|u_{i-1}\|},
\end{aligned} \tag{3.56}$$

with

$$w_i^{m,p} = \mathbf{\Phi}_i^T (c_p - c_m) \in \mathbb{R}^{k_i},\ \forall i \in [1, n_c],\ 1 \leq m < p \leq n_c + 1, \tag{3.57}$$

where the initial cluster is included, therefore $n_c + 1$. An exemplary algorithm to update the sign function is presented in the following:

Algorithm 3 Sign function update for the spherical k-means based ROM.

Input: Sign function at the last time step $z_{i-2}^{m,p}$, precomputed quantities $w_i^{m,p}$, $\mathbf{\Phi}_r^T \mathbf{\Phi}_k$, $1 \leq r \leq k \leq n_c + 1$, norm at the last time step $\|u_{i-2}\|$, current reduced increment $\Delta \hat{x}_{i-1}^{k_{i-2}}$, reduced state at last time step q_{i-2}^r and reduced bases Φ_r, $r \in [1, n_c + 1]$.

Output: Norm of state vector at new time step $\|u_{i-1}\|$ and sign function at new time step $z_{i-1}^{m,p}$.

```
1: Update norm using eq. (3.55)
   $\|u_{i-1}\|^2 \leftarrow \|u_{i-2}\|^2 + \|\Delta \hat{x}_{i-1}^{k_{i-2}}\|^2 + 2 \sum_{r=1}^{n_c} q_{i-2}^r \mathbf{\Phi}_r^T \mathbf{\Phi}_{i-2} \Delta \hat{x}_{i-1}^{k_{i-2}}$
2: Update sign function using eq. (3.56)
   $z_{i-1}^{m,p} \leftarrow \frac{\|u_{i-2}\|}{\|u_{i-1}\|} z_{i-2}^{m,p} + \frac{\Delta \hat{x}_{i-1}^{k_{i-2}} \cdot w_{i-2}^{m,p}}{\|u_{i-1}\|}$
3: return $z_{i-1}^{m,p}$, $\|u_{i-1}\|$
```

3.2.4 Hyper-reduction

As introduced in Section 2.2, hyper-reduction is the only way to achieve computational speedup in highly nonlinear systems besides the larger admissible time step in explicit simulations. We apply ECSW hyper-reduction [130] as it was shown it is a suitable method for highly nonlinear systems [162]. The stability properties are due to the preservation of the Lagrangian structure of the problem [131]. In addition, ECSW is easily compatible with IROB method [92]. In the following, we quickly illustrate the main idea behind ECSW.

To understand ECSW, we recall the assembly of the nonlinear internal force term:

$$f_{int} = \mathop{\mathsf{A}}_{e=1}^{n_e} f_{int}^{(e)} = \sum_{e=1}^{n_e} \mathbf{L}_e f_{int}^{(e)}, \tag{3.58}$$

where the global internal force f_{int} is assembled using the assembly operator A for all elemental force contributions $f_{int}^{(e)}$ of the n_e elements. The assembly operator can be rewritten using assembly matrices $\mathbf{L}_e \in \mathbb{R}^{n \times m_e}$. The DoF for the element are denoted by m_e. Next, we project the internal force onto the reduced basis and obtain the reduced force as appearing in ROM1 eq. (2.25b):

$$\mathbf{\Phi}^T f_{int} = \sum_{e=1}^{n_e} \mathbf{\Phi}^T \mathbf{L}_e f_{int}^{(e)} = \sum_{e=1}^{n_e} \mathbf{\Phi}_e^T f_{int}^{(e)}. \tag{3.59}$$

The assembly matrix selects the rows of $\mathbf{\Phi}$ associated with the DoF of the current element yielding an element-specific reduced basis $\mathbf{\Phi}_e \in^{k\times m_e}$. The reduced force can also be considered a virtual work, with $\mathbf{\Phi}$ being the virtual displacement. The idea of ECSW is to find a weighted combination of a few elements, which yields the same total virtual work as the whole element set:

$$\mathbf{\Phi}^T f_{int} \approx \sum_{e\in\tilde{\mathsf{E}}} \xi_e \mathbf{\Phi}_e^T f_{int}^{(e)}, \quad \xi_e > 0. \tag{3.60}$$

The reduced element set is $\tilde{\mathsf{E}}$ and the non-negative weighting factors are ξ_e. Finding $\tilde{\mathsf{E}}$ and ξ_e requires the solution of an optimization problem, which is defined as [130]:

$$\begin{aligned} \xi^* &= \arg\min_{\xi\in\Lambda} \|\xi\|_0 \\ \Lambda &= \{\xi \in \mathbb{R}^{n_e} : \|G\xi - b\|_2 \le \tau\|b\|_2, \xi_e \ge 0\}, \end{aligned} \tag{3.61}$$

where $\tau \in (0,1]$ is a predefined accuracy. The reduced set $\tilde{\mathsf{E}}$ comprises the elements with non-zero weights:

$$\tilde{\mathsf{E}} = \{e \in \{1,2,\dots,n_e\} : \xi_e \neq 0\} \tag{3.62}$$

The matrix G is composed of submatrices $G_{j,e}$:

$$G = \begin{bmatrix} G_{1,1} & \dots & G_{1,e} & \dots & G_{1,n_e} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{j,1} & \dots & G_{j,e} & \dots & G_{j,n_e} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ G_{n_b,1} & \dots & G_{n_b,e} & \dots & G_{n_b,n_e} \end{bmatrix} \in \mathbb{R}^{n_b\times n_e}, \tag{3.63}$$

where the first dimension n_b is the sum of the reduced dimension k over all n_f force snapshots:

$$n_b = k * n_f. \tag{3.64}$$

The submatrix $G_{j,e}$ is the product of the reduced basis Φ with the force snapshot at t_j of element e:

$$G_{j,e} = \mathbf{\Phi}_e^T f_j^{(e)} \in \mathbb{R}^k. \tag{3.65}$$

The vector b is the column sum of matrix G:

$$b_j = \sum_{e=1}^{n_e} G_{j,e} \in \mathbb{R}^{n_b}. \tag{3.66}$$

Algorithm 4 presents a basic strategy to solve optimization problem eq. (3.61). The algorithm is closely related to the approaches presented in [129, 135], where the active set non-negative least squares (NNLS) solver presented by Lawson and Hanson [163] is utilized in every iteration. It is noted that an advanced version of this algorithm, as proposed by Bach [1] was used in this thesis. Next, we transfer ECSW to local ROB by allowing the reduced basis to change $\mathbf{\Phi} \to \mathbf{\Phi}_i$. That is, an individual matrix G^i and the corresponding vector b^i can be formulated:

$$G_{j,e}^i = (\mathbf{\Phi}_i)_e^T f_j^{(e)} \in \mathbb{R}^{k_i}, \quad f_j^{(e)} \in \pi_i^{\mathsf{fin}} \tag{3.67}$$

and

$$b_j^i = \sum_{e=1}^{n_e} G_{j,e}^i. \tag{3.68}$$

It is assumed that force snapshots are collected at the same time steps as displacement time snapshots. Therefore, $f_j^{(e)} \in \pi_i^{\mathsf{fin}}$ states that if a displacement snapshot at the current time is in cluster i, then also the force snapshot at this time is assigned to the same cluster. Depending on the formulation of the optimization problem, different strategies to calculate the reduced element set and the weights exist. In the following, we examine all three possibilities.

Algorithm 4 Basic greedy algorithm to solve optimization problem eq. (3.61).

```
Input: $G$, $b$, tolerance $\tau \in (0,1]$
Output: Reduced element set $\tilde{\mathsf{E}}$, element weights $\xi_n,\ n = 1,\dots,|\tilde{\mathsf{E}}|$.
1: Initialize element set, weight vector and residual vector
   $\tilde{\mathsf{E}} \leftarrow \{\}$, $\xi \leftarrow \underline{0}$, $r \leftarrow b$
2: while $\|r\| \,/\, \|b\| \geq \tau$ do
3:     Compute gradient of residual regarding not included elements $\nabla \leftarrow (G_{:,\tilde{\mathsf{E}}^C})^T\, r$,
   where $\tilde{\mathsf{E}}^C$ denotes the complement of the set.
4:     Add element corresponding to the largest element in $\nabla$ to the reduced set $\tilde{\mathsf{E}}$.
5:     Solve NNLS of $\xi \leftarrow \arg\min_\xi \left\| G_{:,\tilde{\mathsf{E}}}\xi - b \right\|^2$
6:     Remove zero weights from $\xi$ and the corresponding elements from the reduced set $\tilde{\mathsf{E}}$.
7:     Update residual $r \leftarrow b - G_{:,\tilde{\mathsf{E}}}\xi$.
8: end while
9: return $\tilde{\mathsf{E}}$, $\xi$
```

Global-global

Here, the reduced element set $\tilde{\mathsf{E}}_{\text{glob}}$ as well as the corresponding weights ξ^*_{glob} are global and do not change during a cluster change during the online phase. The matrix G contains all local matrices G^i as defined in eq. (3.67):

$$G_{\text{glob}} = \begin{bmatrix} G^1 \\ \vdots \\ G^i \\ \vdots \\ G^{n_c} \end{bmatrix} \in \mathbb{R}^{n_b \times n_e},\ i = 1,..,n_c, \tag{3.69}$$

where

$$n_b = \sum_{i=1}^{n_c} |\pi_i^{\text{fin}}| k_i. \tag{3.70}$$

The vector b_{glob} is again defined as:

$$b_{\text{glob}} = \sum_{e=1}^{n_e} G^i_{j,e} \in \mathbb{R}^{n_b}. \tag{3.71}$$

Finally, $\tilde{\mathsf{E}}_{\text{glob}}$ and ξ^*_{glob} are obtained by solving the given optimization problem:

$$\xi^*_{\text{glob}} = \arg\min_{\xi\in\Lambda} \|\xi\|_0$$
$$\Lambda = \{\xi \in \mathbb{R}^{n_e} : \|G_{\text{glob}}\xi - b_{\text{glob}}\|_2 \leq \tau \|b_{\text{glob}}\|_2, \xi_e \geq 0\}, \tag{3.72}$$

and the reduced element set obtained as:

$$\tilde{\mathsf{E}}_{\text{glob}} = \{e \in \{1,2,\dots,n_e\} : \xi^*_{\text{glob},e} \neq 0\}. \tag{3.73}$$

The global approach is easy to implement due to no necessity to change the reduced mesh and/or the associated weights. However, the dimension n_b of the matrix G_{G} can become large quickly. Due to the large data set, including snapshots from different subregions/clusters, the resulting reduced mesh is larger than experience supports.

Local-local

In contrast to the global option, the reduced element set $\tilde{\mathsf{E}}_i$ and the weights ξ_i^* can be local (cluster-specific). The following optimization problem is solved for each cluster:

$$\xi_i^* = \arg\min_{\xi\in\Lambda} \|\xi\|_0$$
$$\Lambda = \{\xi \in \mathbb{R}^{n_e} : \|G^i\xi - b^i\|_2 \leq \tau\|b^i\|_2, \xi_e \geq 0\}, \tag{3.74}$$

and the reduced element set obtained as:

$$\tilde{\mathsf{E}}_i = \{e \in \{1, 2, \dots, n_e\} : \xi_{i,e}^* \neq 0\}, \tag{3.75}$$

for each cluster

$$i = 1, \dots, n_c. \tag{3.76}$$

The size of each optimization problem is smaller compared to the global approach. In this approach, the reduced mesh and the weights are tailored to each cluster, usually resulting in a smaller reduced mesh and, consequently, a larger computation speedup. However, this approach is not able to handle path-dependent materials. Assuming an element was not considered in the first reduced mesh and is activated during the online phase of the simulation. In the integration of the material routine, the history needs to be included. Using methods for gappy data, such as the gappy POD, could reconstruct the field of internal variables. However, this is beyond the scope of this work and requires currently not accessible parts of the FEM solver.

Global-local

In this thesis, we propose a global element set with local weights. It benefits from higher accuracy due to the local weights and is compatible with path-dependent materials due to the global reduced mesh. Algorithm 5 presents the applied scheme. In the first loop, the reduced mesh and the local weights are

Algorithm 5 Global ECSW hyper-reduction for IROB with local weights.

```
Input: $G^i$ and $b^i$ for $i = 1, \dots, n_c$, tolerance $\tau \in (0, 1]$
Output: Reduced element set $\tilde{\mathsf{E}}_{\mathrm{glob}}$, local element weight vectors $\xi_i^*$, $i = 1, \dots, n_c$ with true accuracy $\tau_i$.
1: for $i \in \{1, \dots, n_c\}$ do
2:     $\tilde{\mathsf{E}}_i, \xi_i^* \leftarrow$ Algorithm 4($G^i, b^i, \tau$)
3: end for
4: $\tilde{\mathsf{E}}_{\mathrm{glob}} \leftarrow \bigcup_{i=1}^{n_c} \tilde{\mathsf{E}}_i$
5: for $i \in \{1, \dots, n_c\}$ do
6:     Solve NNLS of $\xi_i^*, \tau_i \leftarrow \arg\min_\xi \left\| G^i_{:,\tilde{\mathsf{E}}_{\mathrm{glob}}}\xi - b^i \right\|^2$
7: end for
8: return $\tilde{\mathsf{E}}_{\mathrm{glob}}$, $\xi_i^*$, $\tau_i$
```

computed as described in the local-local approach. However, in a subsequent step, a global reduced mesh is formed by the union of all local reduced meshes. The weights are adjusted to the probably larger reduced mesh in a second loop, and the true accuracy τ_i is returned.

3.3 Results

This section applies the proposed method to a typical crash problem. We use the crash box example as it is a well-established benchmark for crash analysis. The model is introduced in Subsection 2.4.3, and the parameters are varied as explained in Section 3.1.1. Also, the critical time step is not kept fixed and

is calculated in each time step with a safety factor of 0.9. In addition, the eight triangular shells existing in the original mesh are replaced by four quadrilateral shell elements, resulting in a total of 1,864 shell elements. During the 35 ms simulation time, every 0.01 ms, a snapshot of the displacements, rotations, forces, and moments is collected. This collection strategy returns 3,501 snapshots for each variable for each simulation.
The results section is structured as follows. First, the offline phase results are presented, consisting of clustering and basis construction. Next, the resulting ROMs are presented, and the approximation quality is assessed without hyper-reduction. Subsequently, hyper-reduction is introduced, the offline phase is summarized in tables, and the resulting online accuracy is evaluated. The evaluations are always conducted on one test case with high deformation and one test case with low deformation.

3.3.1 Clustering

First, clustering is evaluated. To better understand the clustering results later, the snapshots are assessed by investigating their magnitude $\|x(t)\|$. Fig. 3.18 shows the evolution of the magnitude until the termination time of the simulation. The magnitude is shown for the test case with low deformation and the test case with large deformation. A monotonic increase in magnitude is observable for both simulations. Only at the end of the simulation, where the whole kinetic energy of the impacting plate is absorbed, and the rebound occurs, a small decrease in magnitude appears. Systems with oscillating solutions or systems with Eulerian description would have a different magnitude behavior. Exceptions may exist, or systems may be designed to replicate this behavior. Next, we need to select a reasonable number of clusters. The

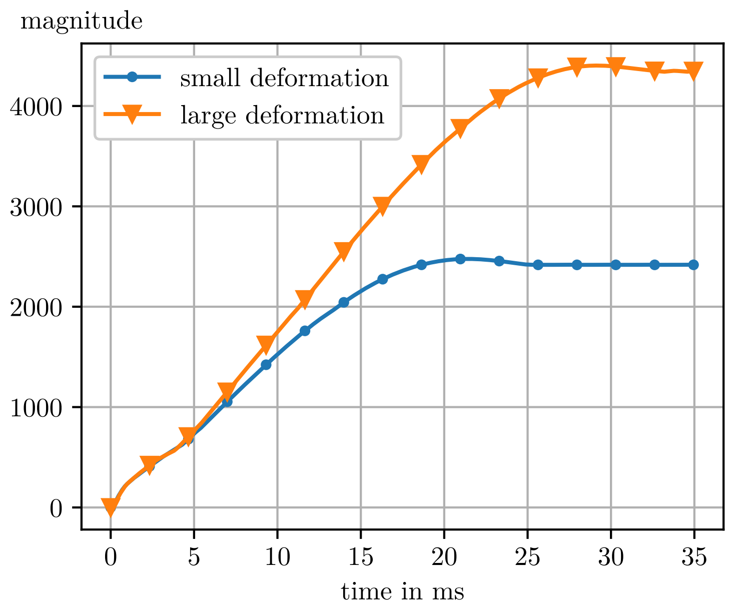


Figure 3.18 Temporal evolution of the magnitude of the state vector $\|x(t)\|$ containing displacement and rotation DoF.

elbow method is a graphical method to select the number of clusters in a data-set. This method observes the total variance of the clustering for different numbers of clusters and stops where there is a kink and selects this number. The method also has this name since the curve looks like an elbow. However, the method is subjective and unreliable if no sharp elbow is visible. Nevertheless, it meets our requirements and is suitable for selecting a reasonable number of clusters. Fig. 3.19 shows the total variance of the k-means clustering for an increasing number of clusters. The curve has the shape of an elbow, and we select three numbers of clusters for all further evaluations. We select $n_c = 3$, $n_c = 6$, and $n_c = 9$. As the local hyper-reduction depends on the number of clusters, we choose a small number, which could be too small, a presumably reasonable number, and finally, a conservative number, which does not reduce the total variance strongly compared to the previous cluster number. Fig. 3.20 evaluates the total variance

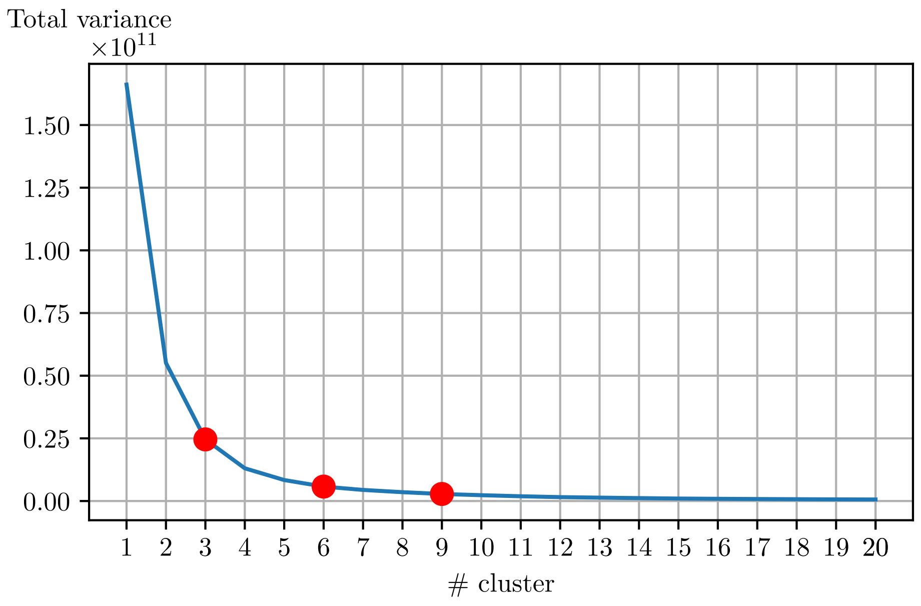


Figure 3.19 Elbow of k-means clustering.

for the spherical k-means clustering. A direct comparison to k-means clustering is impossible due to the different clustering metrics and variance range. However, the same behavior is visible. We choose the same number of clusters for further investigation. Next, we investigate how the clustering assigns the

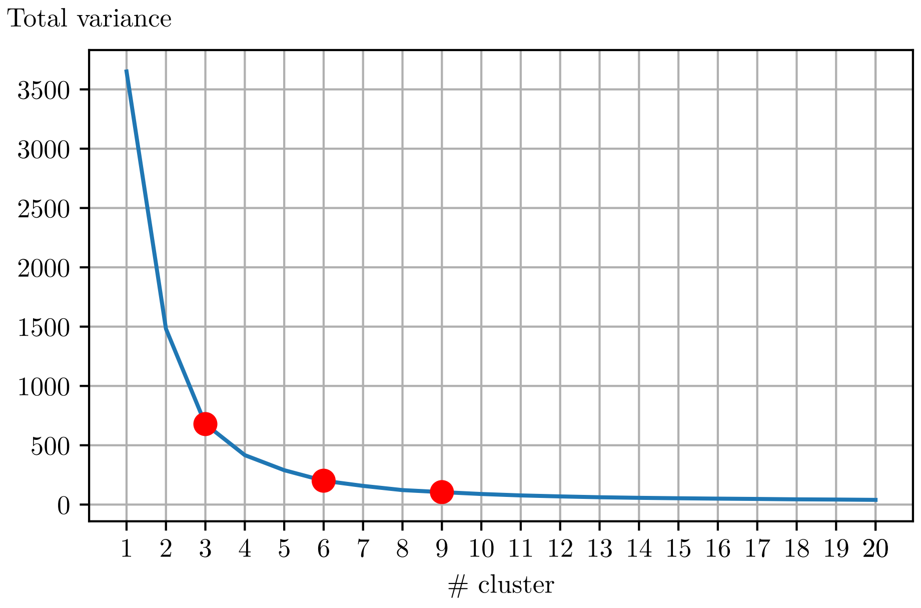


Figure 3.20 Elbow of spherical k-means clustering.

snapshots to the different clusters. Fig. 3.21 and Fig. 3.22 show the results for k-means and spherical k-means clustering, respectively. k-means clustering is based on the L^2 norm. As shown in Fig. 3.18, the norm constantly increases during the simulation. This is mirrored in Fig. 3.21, where the assignment of the individual snapshots to the corresponding cluster is indicated by the color for all 27 training simulations until the final simulation time. The snapshots are divided along the temporal axis. Especially in the first 15 ms, the magnitudes of the different training simulations are similar, and most training simulation snapshots are assigned to the same cluster. Afterwards, the magnitudes differ and snapshots associated with a higher deformation are assigned to a different cluster than snapshots related to a lower deformation. For instance, training simulations 1, 26, and 27 are of lower total deformation and stay in cluster one for the remainder of the simulation. In contrast, other simulations change to cluster 3 in the last milliseconds of the simulation. Fig. 3.22 shows the results for the spherical k-means clustering. The first 4 ms are excluded

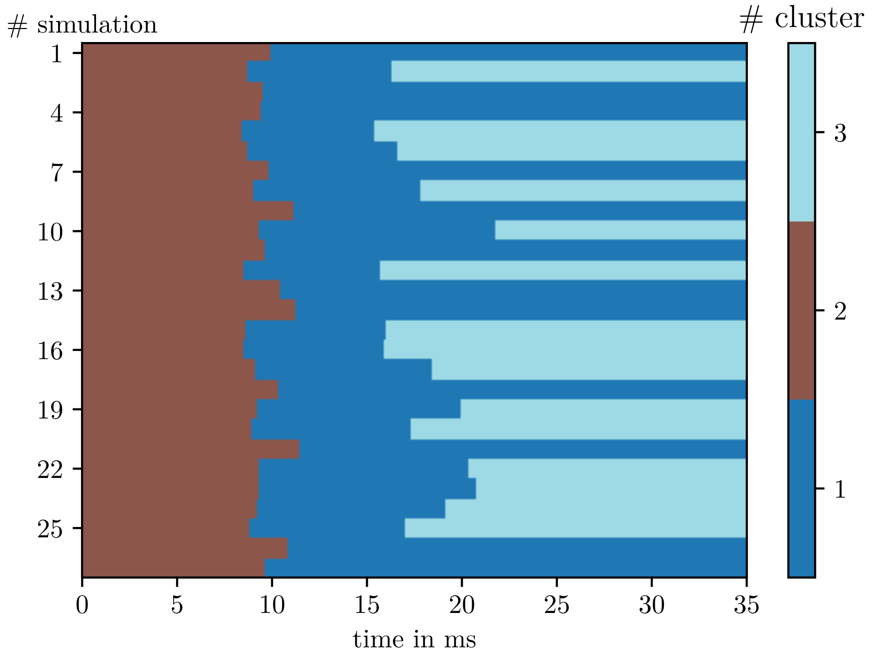


Figure 3.21 Cluster visualization for 3 clusters and k-means clustering.

from clustering, forming the initial cluster. Spherical k-means clustering divides the remaining snapshots. The cluster change to the final cluster happens on average earlier, and fewer training simulations change the cluster only twice. Spherical k-means clustering is more sensitive at later simulation times and, therefore, higher magnitudes. This is probably due to the normalization in spherical k-means. Fig. 3.23 and Fig. 3.24 show the results for k-means and spherical k-means clustering for 6 clusters, respectively. Fig. 3.25 and Fig. 3.26 provide the results for 9 clusters. With the increasing number of clusters, the division of the snapshots becomes finer for both clustering methods. Especially in the spherical k-means clustering, the clusters are thin in the early phase of the simulation, and cluster changes appear more often compared to the k-means clustering. In the later phases, clusters are larger compared to k-means.

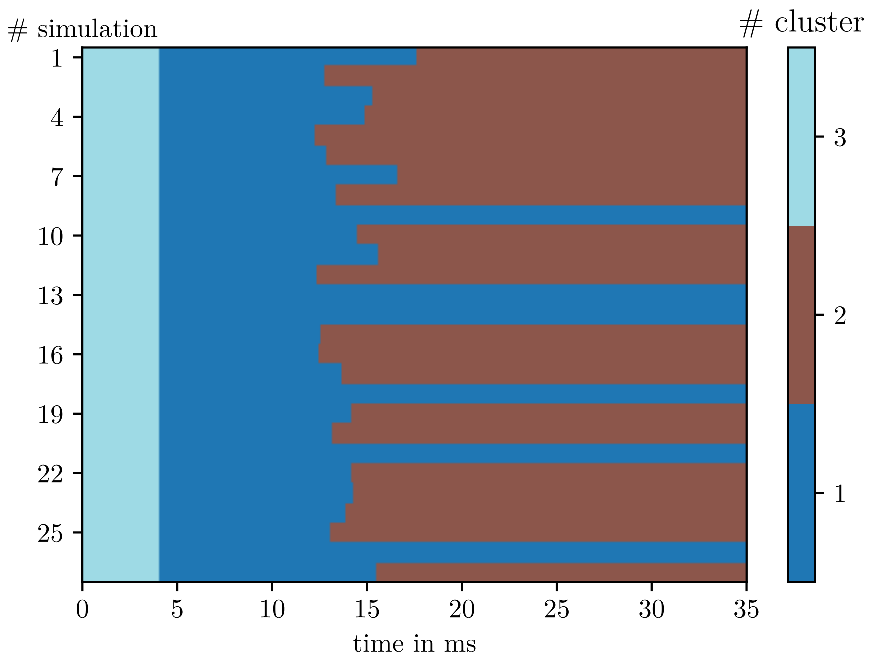


Figure 3.22 Cluster visualization for 3 clusters and spherical k-means clustering. The initial cluster was manually chosen to contain snapshots from the first 4 ms.

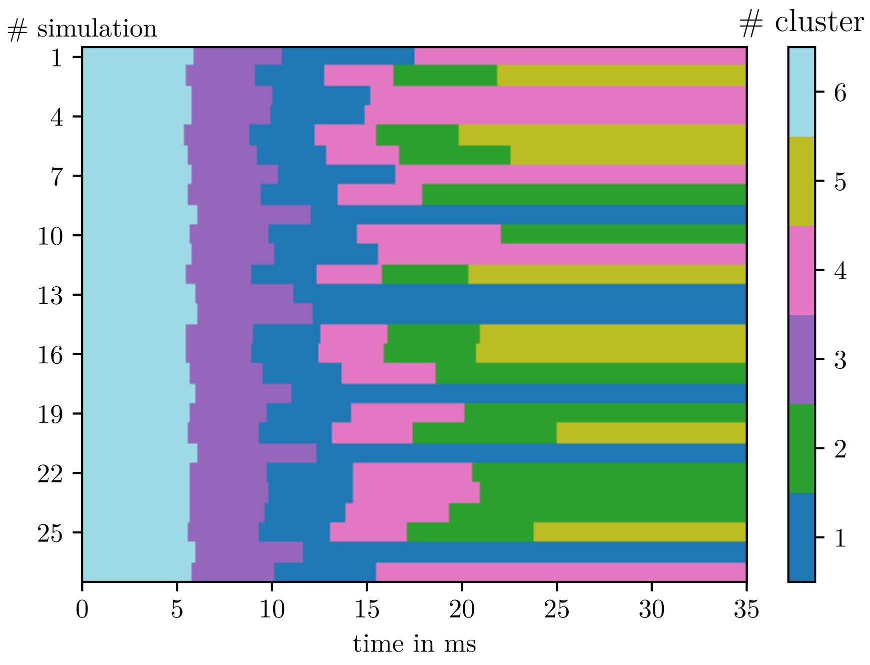


Figure 3.23 Cluster visualization for 6 clusters and k-means clustering.

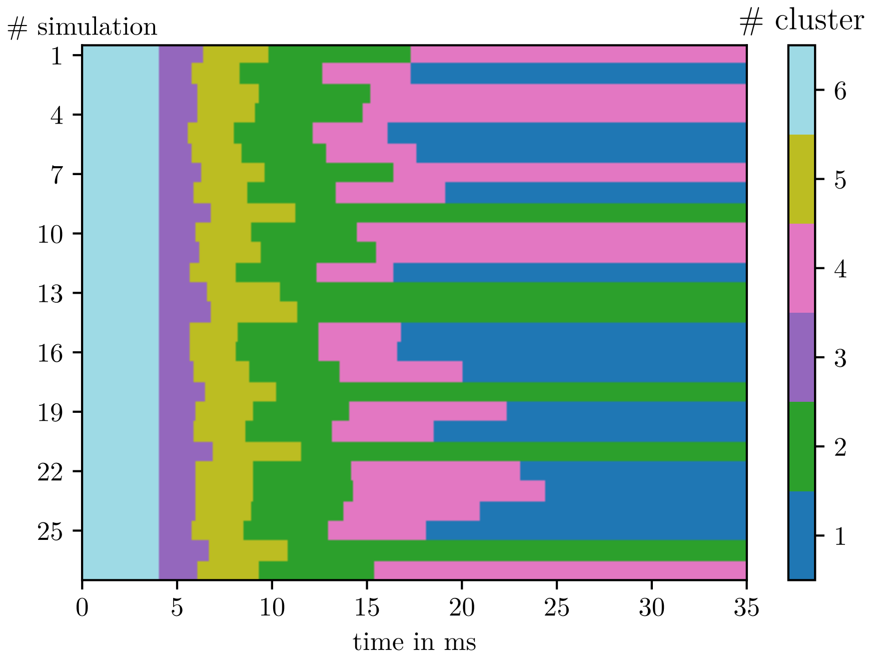


Figure 3.24 Cluster visualization for 6 clusters and spherical k-means clustering. The initial cluster was manually chosen to contain snapshots from the first 4 ms.

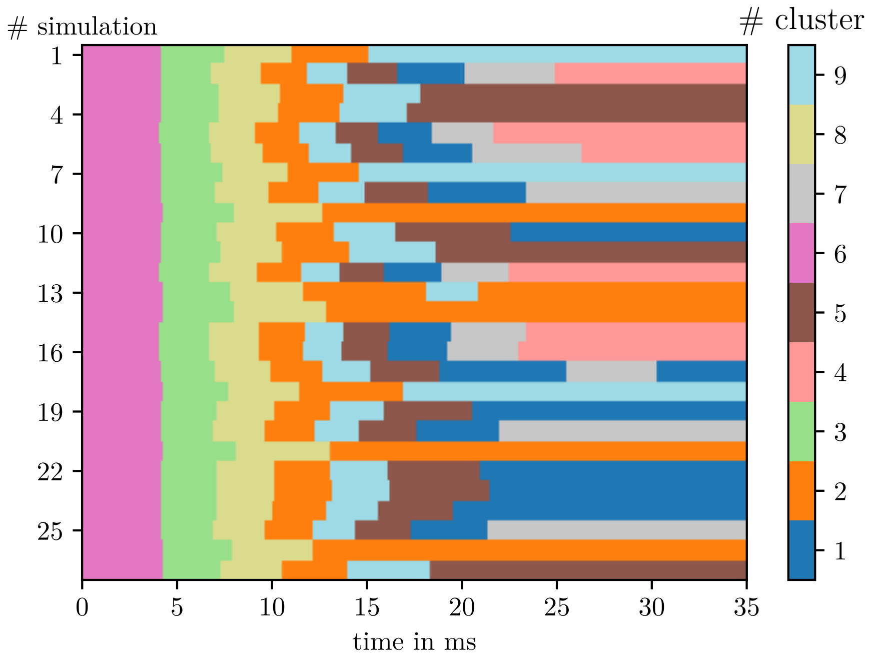


Figure 3.25 Cluster visualization for 9 clusters and k-means clustering.

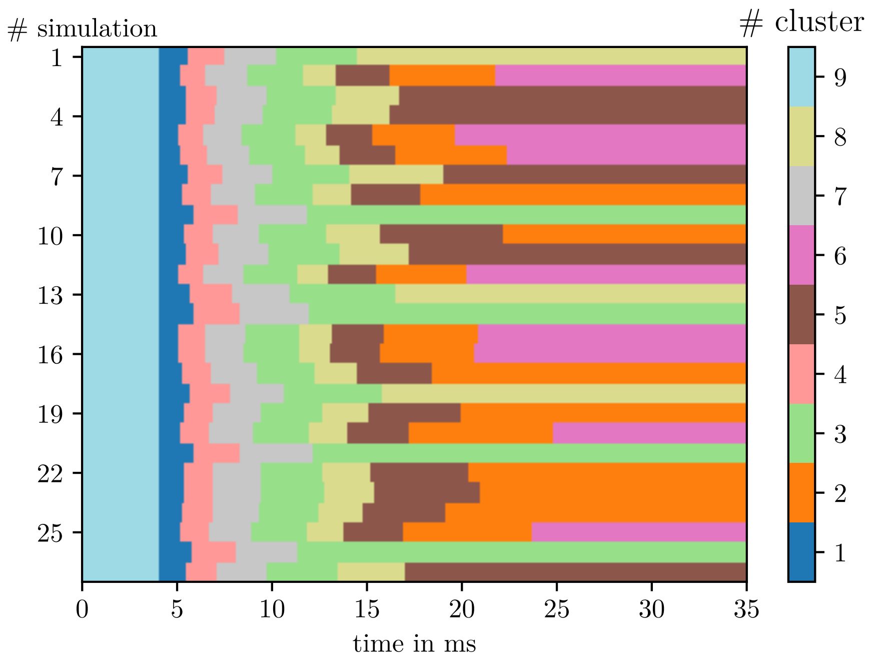


Figure 3.26 Cluster visualization for 9 clusters and spherical k-means clustering. The initial cluster was manually chosen to contain snapshots from the first 4 ms.

Before the ROBs are created, near snapshots are added to ensure an overlap of the subspaces and a smooth transition between them. We visualize where snapshots are added for the ROM with 6 clusters. For each cluster, the nearest 2,000 snapshots are added. Fig. 3.27 visualizes the added snapshots for each snapshot. In each subfigure, the snapshots belonging to the cluster are colored in turquoise, and the added snapshots are colored in yellow. Except for clusters 2 and 5, where snapshots from other training simulations are identified as closest and added to the cluster, the nearest added snapshots lie directly at the border of the cluster itself and ensure a smooth transition of the solution from one cluster to the other during temporal evolution.

Fig. 3.28 shows the results for spherical k-means clustering. The initial cluster (cluster 6) is predefined and considered as "added", which is why it is colored yellow. Like k-means clustering, the clusters containing the snapshots appearing later in time identify snapshots from other training simulations as closest. For the other clusters, snapshots are added at the borders of the clusters.

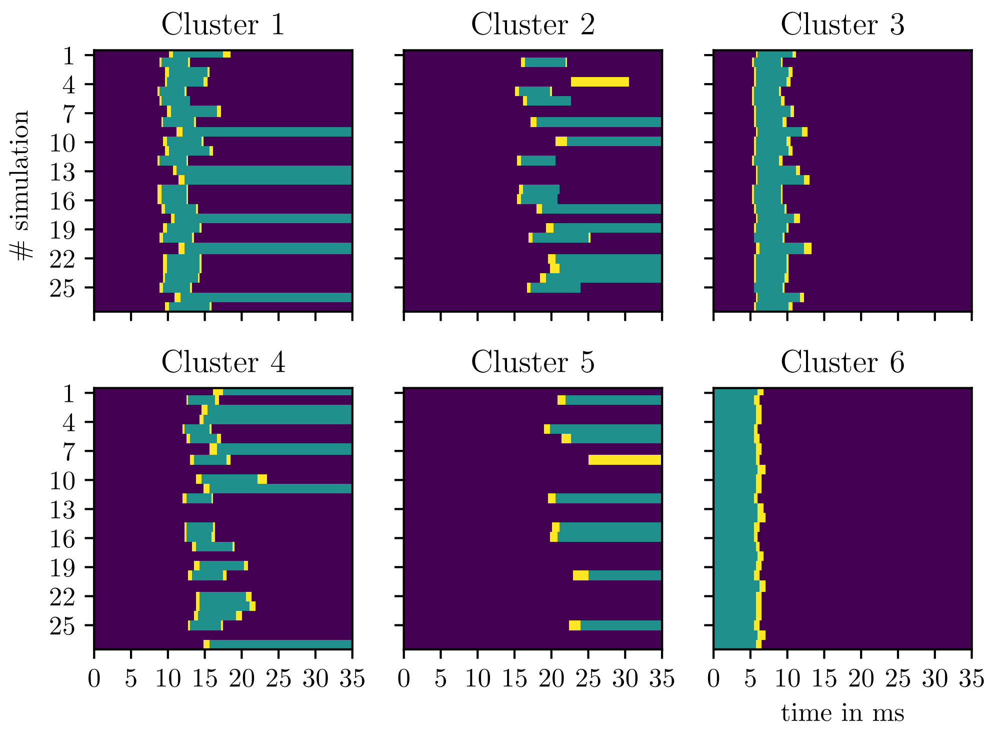


Figure 3.27 Visualization of the closest 2,000 snapshots, which are added for k-means clustering using 6 clusters.

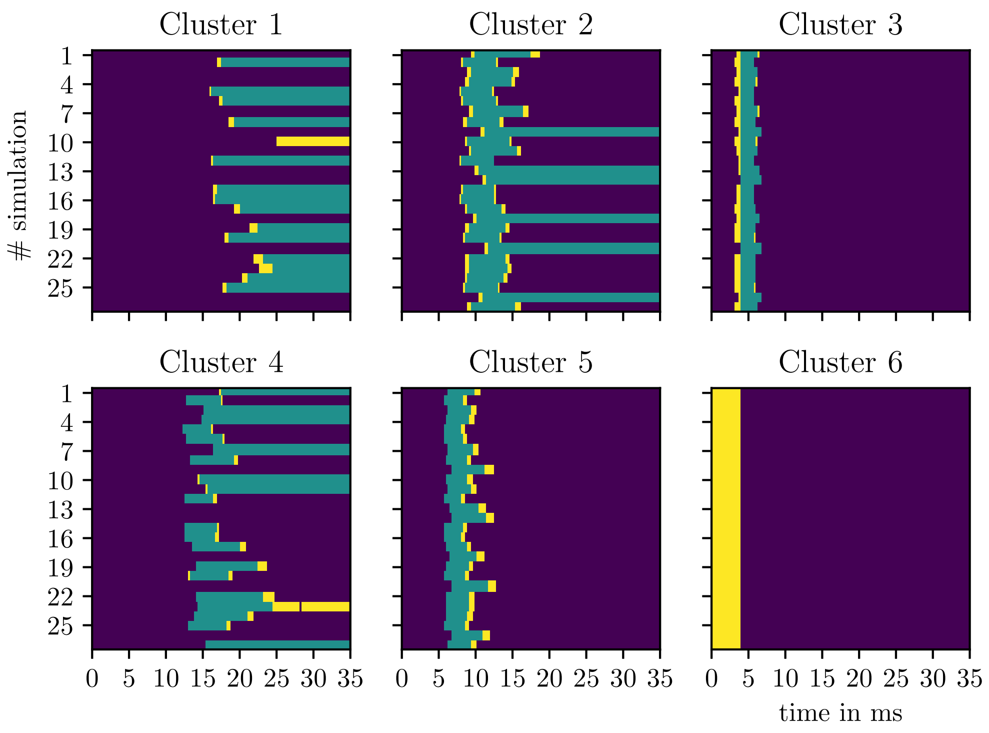


Figure 3.28 Visualization of the closest 2,000 snapshots, which are added for spherical k-means clustering using 6 clusters. No snapshots are added to the initial cluster.

3.3.2 Offline Accuracy

Next, we investigate the approximation quality of the ROB and compare it to the offline error of the global basis. The global basis is the analogy to a local ROM with one cluster containing the snapshots of all training simulations. Fig. 3.29, Fig. 3.30, and Fig. 3.31 show the offline/approximation error $\epsilon^2(k)$ for an increasing number of dimensions k for the clusters obtained by k-means clustering for 3, 6, and 9 clusters, respectively. For comparison, the offline error for the global basis is given for reference. For the clustering with 3 and 6 clusters, all IROBs show a faster decay in error except the basis containing the initial snapshots, which is basis 2 for 3 clusters and basis 6 for 6 clusters. For the clustering with 9 clusters, basis 6, and basis 3 show a slower error decay than the global basis. Both IROBs are related to the initial snapshots. Basis 6 contains the first snapshots, and basis 3 the subsequent ones.

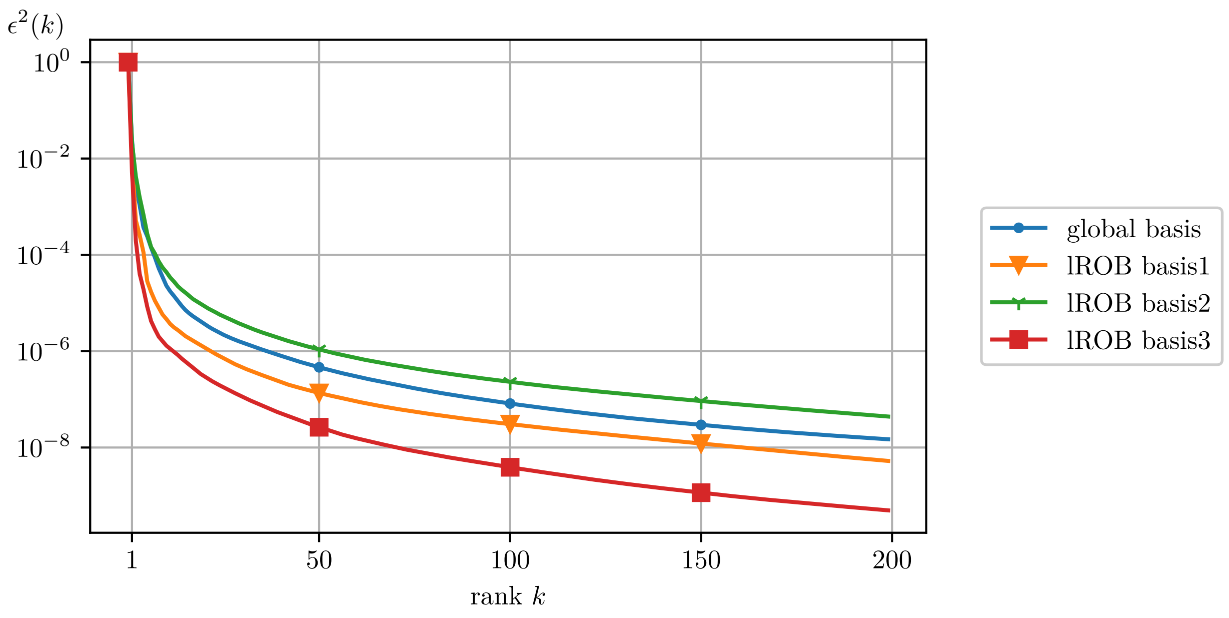


Figure 3.29 Offline accuracy of the ROB for k-means clustering using 3 clusters.

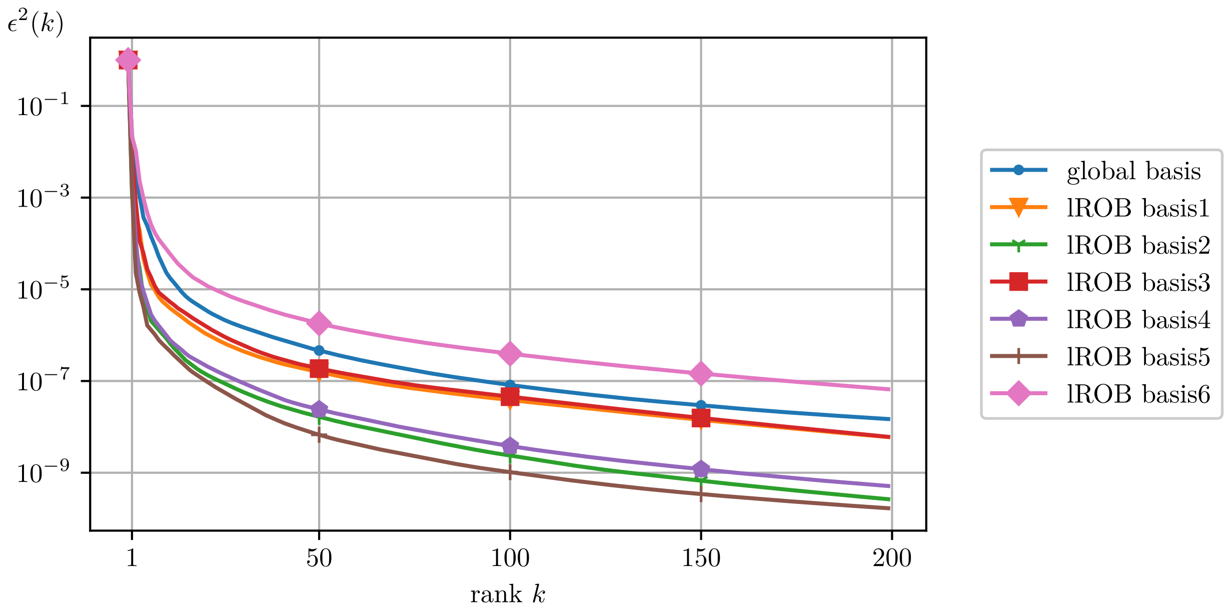


Figure 3.30 Offline accuracy of the ROB for k-means clustering using 6 clusters.

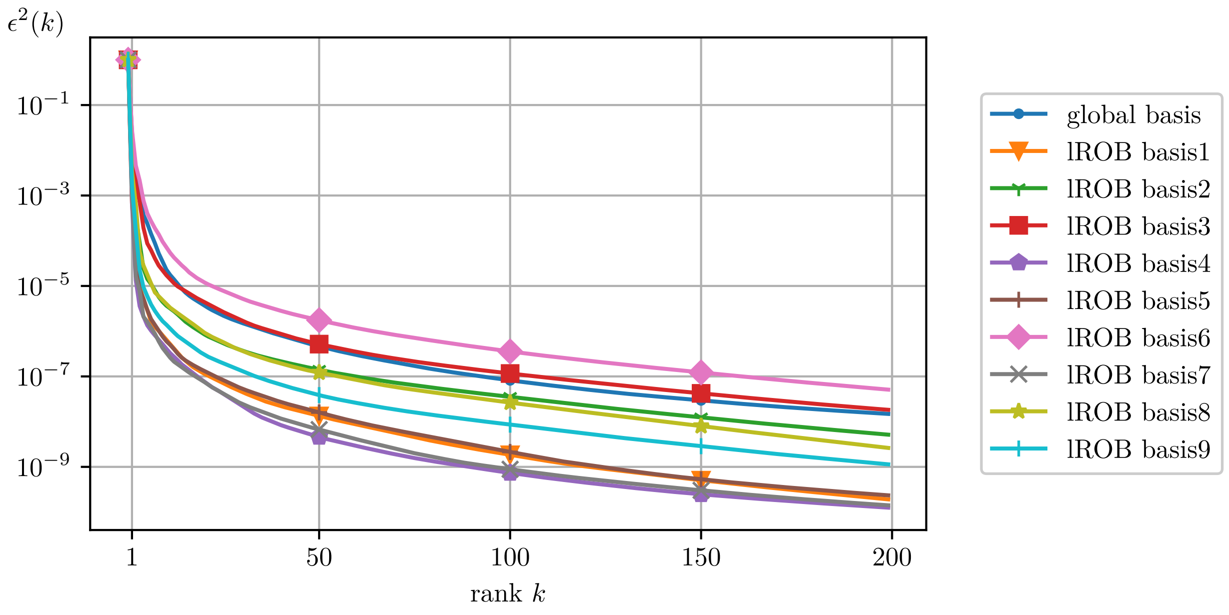


Figure 3.31 Offline accuracy of the ROB for k-means clustering using 9 clusters.

Fig. 3.32, Fig. 3.33, and Fig. 3.34 show the offline/approximation error $\epsilon^2(k)$ of the lROBs associated with the clusters obtained by spherical k-means clustering for 3, 6, and 9 clusters, respectively. The error is plotted for an increasing number of approximation dimensions k. In analogy to the k-means case, the lROBs containing the initial snapshots show the slowest decay in error. In the case of 3 clusters, the second cluster yields a similar error to the global basis. Also, in the 6 and 9 clusters cases, the lROBs related to the early snapshots in the simulation show a slow error decay. The next section assesses whether the slowly decaying approximation error correlates with the online error of the resulting ROM.

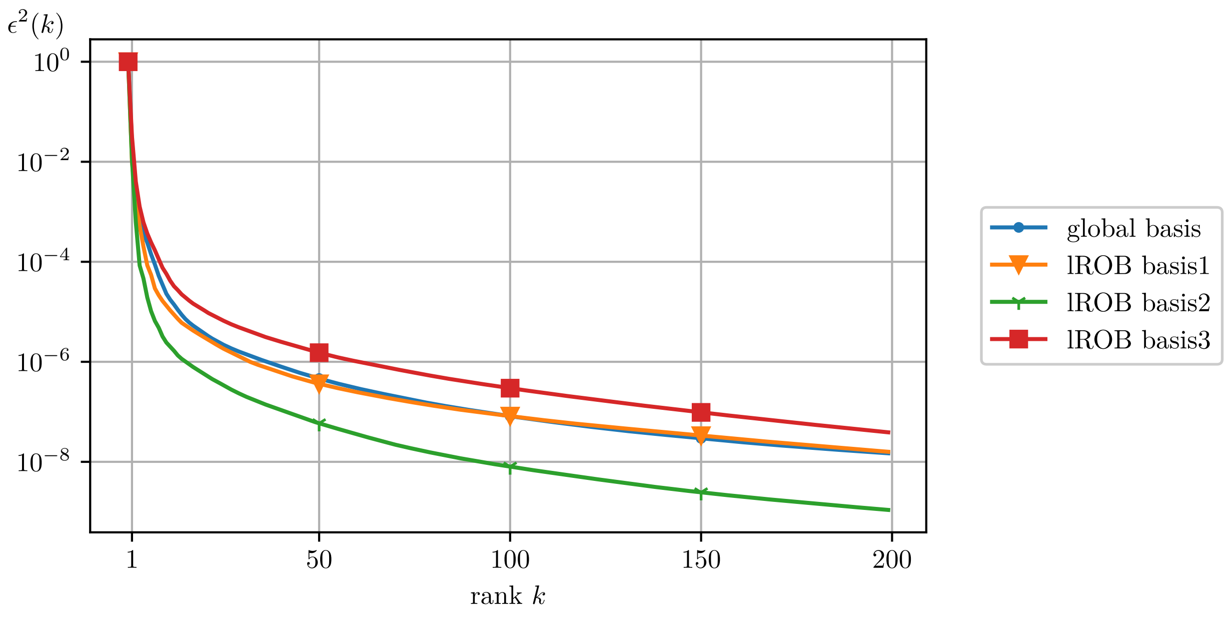


Figure 3.32 Offline accuracy of the ROB for spherical k-means clustering using 3 clusters.

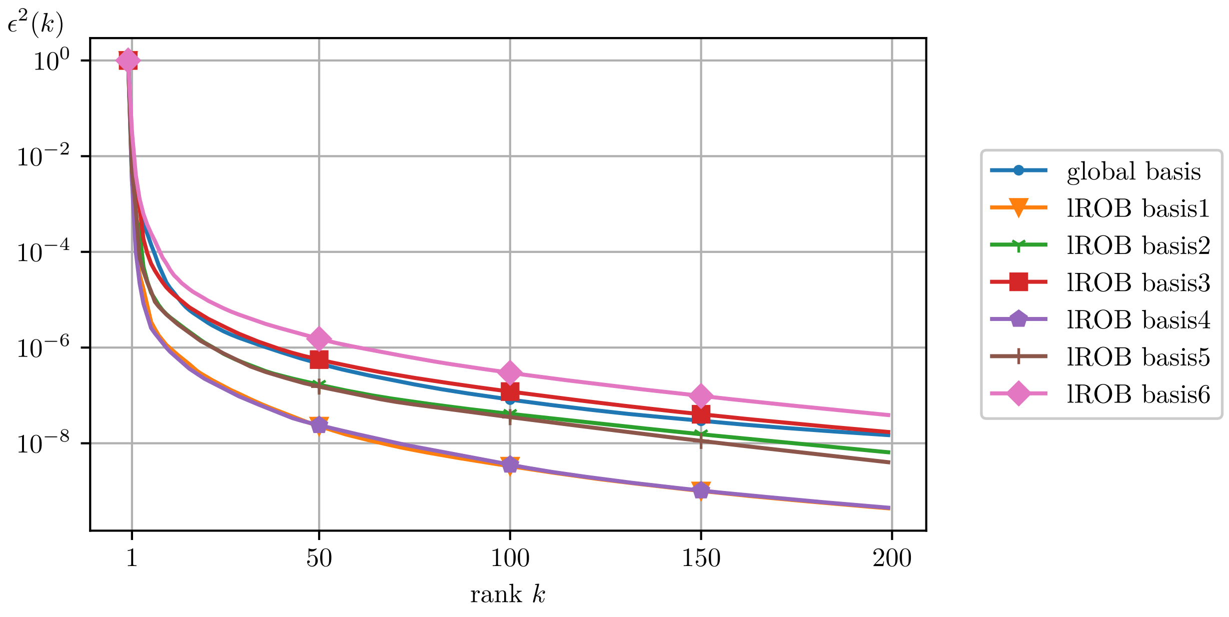


Figure 3.33 Offline accuracy of the ROB for spherical k-means clustering using 6 clusters.

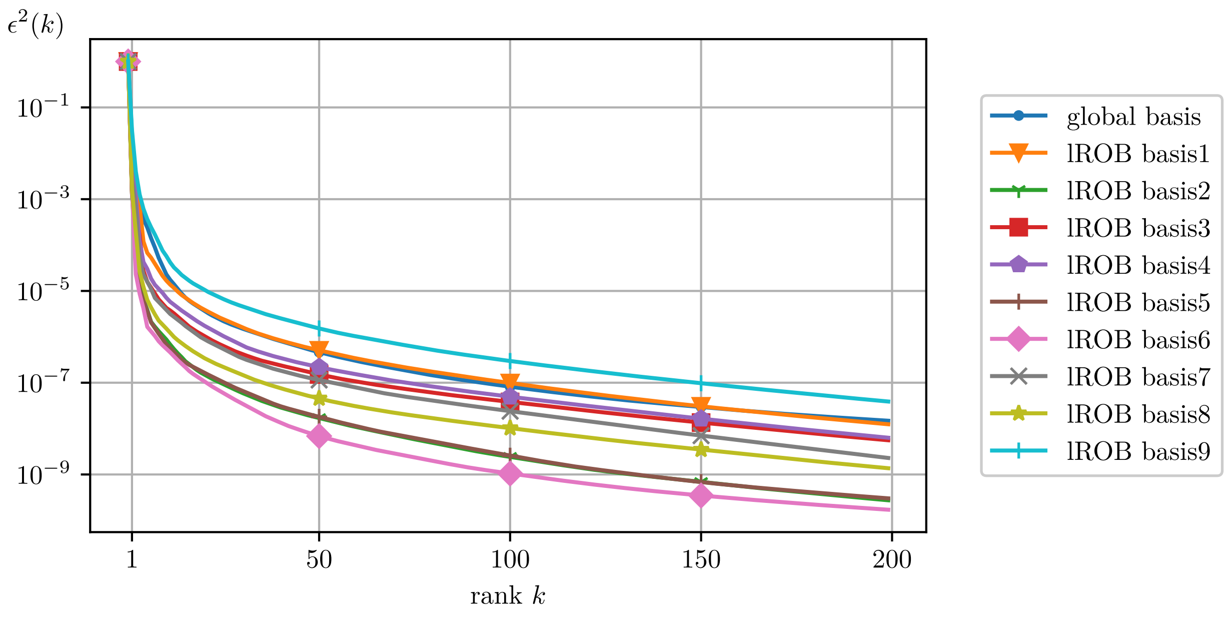


Figure 3.34 Offline accuracy of the ROB for spherical k-means clustering using 9 clusters.

The results of the offline phase are summarized in Tab. 3.2. For each ROM, denoted by the number of clusters in the first column, the number of snapshots in the cluster $|\pi_i|$ and the offline accuracy for an approximation rank of $k = 15, 30, 40$ is tabulated. No correlation between the cluster size and the achievable accuracy can be found. This mirrors the observation of the clusters containing the initial snapshots.

Table 3.2 Summary of clustering results and accuracy of linear dimensional reduction.

# cluster	cluster	k-means				spherical k-means			
	i	$\|\pi_i\|$	$\epsilon_i^2\|_{k_i=15}$	$\epsilon_i^2\|_{k_i=30}$	$\epsilon_i^2\|_{k_i=40}$	$\|\pi_i\|$	$\epsilon_i^2\|_{k_i=15}$	$\epsilon_i^2\|_{k_i=30}$	$\epsilon_i^2\|_{k_i=40}$
3	1	45164	0.205%	0.046%	0.023%	41676	0.534%	0.118%	0.060%
	2	27668	1.640%	0.350%	0.181%	46051	0.101%	0.021%	0.011%
	3	27695	0.061%	0.011%	0.005%	10800	1.921%	0.458%	0.252%
6	1	25308	0.215%	0.045%	0.024%	24281	0.053%	0.010%	0.005%
	2	16991	0.029%	0.006%	0.003%	27251	0.244%	0.049%	0.026%
	3	13714	0.303%	0.061%	0.030%	7631	0.798%	0.180%	0.091%
	4	20586	0.040%	0.009%	0.004%	23949	0.042%	0.009%	0.004%
	5	12495	0.021%	0.003%	0.001%	10615	0.232%	0.047%	0.025%
	6	17433	2.546%	0.555%	0.295%	10800	1.921%	0.458%	0.252%
9	1	13436	0.022%	0.004%	0.002%	5803	0.711%	0.160%	0.083%
	2	19653	0.162%	0.037%	0.021%	17107	0.029%	0.006%	0.003%
	3	10218	0.755%	0.168%	0.087%	19232	0.193%	0.042%	0.024%
	4	8853	0.016%	0.002%	0.001%	6562	0.327%	0.064%	0.034%
	5	14083	0.023%	0.005%	0.003%	16528	0.027%	0.007%	0.003%
	6	13264	2.417%	0.519%	0.282%	12617	0.022%	0.003%	0.001%
	7	8813	0.014%	0.002%	0.001%	9062	0.172%	0.034%	0.018%
	8	10782	0.186%	0.036%	0.019%	12816	0.063%	0.015%	0.008%
	9	13425	0.059%	0.013%	0.007%	10800	1.921%	0.458%	0.252%

3.3.3 Online Phase Results

The last section investigated the approximation error of the IROBs. It is found that the snapshots in the early phase of the simulations are harder to approximate than the ones towards the termination time of the simulations. However, the online error of the ROM must show whether a correlation exists to the approximation error of the IROB. Therefore, the temporal average error ϵ of the ROMs is shown in Fig. 3.35 for all ROMs, including the global ROM tested on the high deformation case. All local ROMs are more accurate than the global ROM using fewer dimensions, which is why Fig. 3.36 zooms in and enables a better comparison of the individual ROMs. The error decreases for all ROMs with increasing rank k. The k-means-based ROM with three clusters has the largest error with the slowest error decay. The k-means-based ROM with 6 clusters and the spherical k-means-based ROM with 3 clusters have comparable accuracy. The error decay for the k-means-based ROMs is not as smooth as for the spherical k-means-based ROMs. That is why the k-means ROM with 9 clusters is more accurate for small ranks, but for higher ranks, the spherical k-means ROM with 9 clusters is more accurate and converges slightly faster to a small error. In general, the error decay of the spherical k-means ROMs is more consistent, and the range of achieved errors is smaller than for the k-means ROMs.

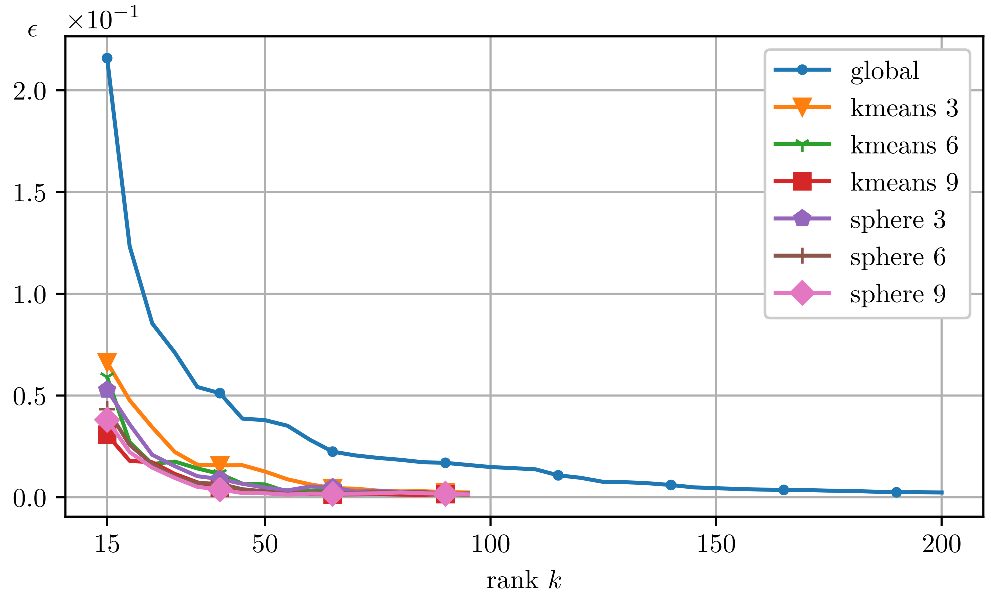


Figure 3.35 Online accuracy for all local ROMs and the global ROM. All ROMs are tested on the high deformation case.

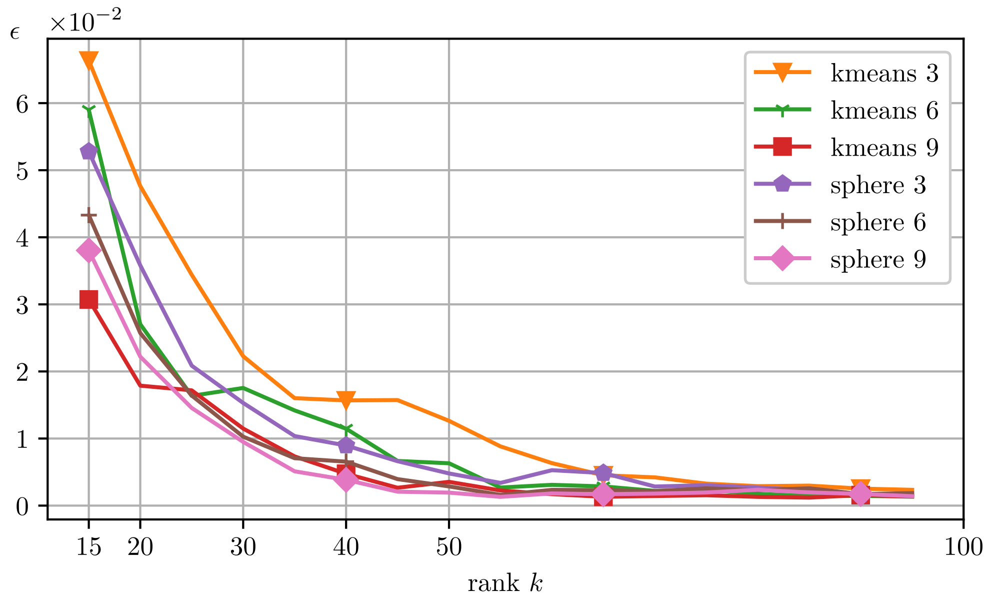


Figure 3.36 Detailed comparison of the online accuracy for all local ROMs, tested on the high deformation case.

The same results are also shown for the low deformation test case. First, in Fig. 3.37, the results of all local ROMs are shown in contrast to the global basis. Also, in the low deformation test case, the local approach proves its success, and the ROMs achieve a higher accuracy with fewer dimensions than the global ROM. Also, here, we zoom in and investigate the difference between the individual ROMs, as can be seen in Fig. 3.38. Compared to the high deformation test case, the ROMs converge faster to low errors. k-means ROMs and spherical k-means ROMs have comparable errors for the same number of clusters. However, the k-means ROM with 3 clusters barely reaches an error below 0.08, and the k-means ROM with 6 dimensions also requires many dimensions to reach the same error as the other ROMs. In contrast, at $k = 40$, all spherical k-means ROMs have a similar error, regardless of the number of clusters.

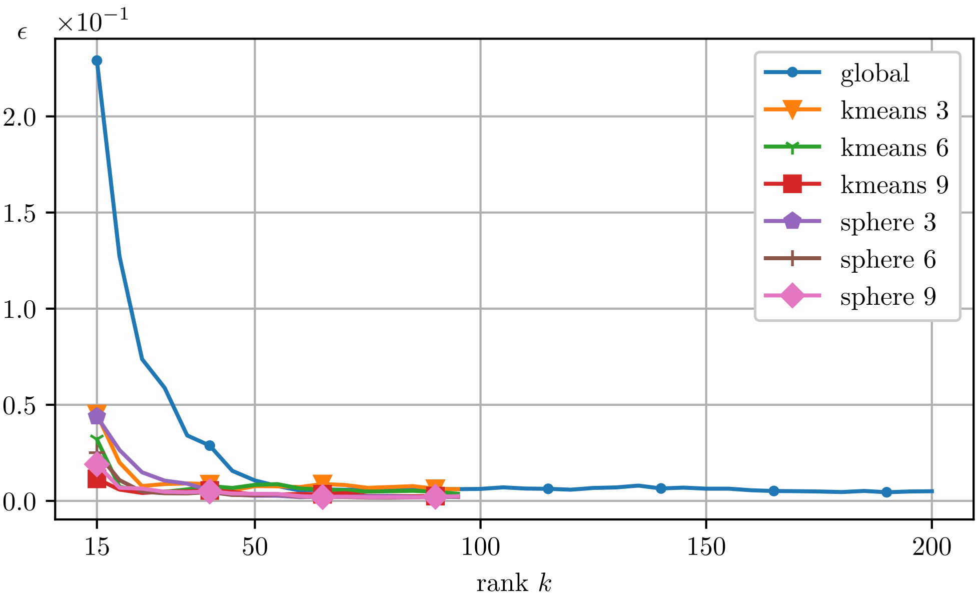


Figure 3.37 Online accuracy for all local ROMs and the global ROM. They are tested on the low deformation case.

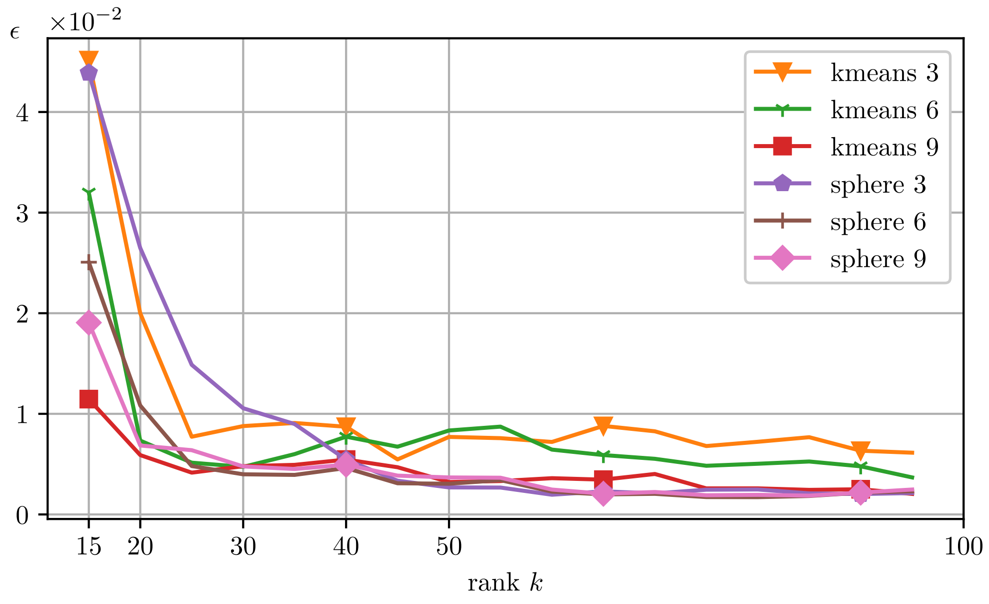


Figure 3.38 Detailed comparison of the online accuracy for all local ROMs tested on the low deformation case.

3.3.4 Hyper-reduction Offline Results

In this section, the results of ECSW hyper-reduction are presented. Force snapshots are collected every 0.01 ms and are assigned to the same cluster as the displacement snapshot at that time. Tab. 3.3, Tab. 3.4, and Tab. 3.5 summarize the offline results of global hyper-reduction for ROMs with reduced dimension $k = 15$, $k = 30$, and $k = 40$, respectively. The tables further show the results for the k-means ROM and the spherical k-means ROM with 3, 6, and 9 clusters and accuracy $\tau = 0.01$, $\tau = 0.02$, and $\tau = 0.03$. The number of clusters has little effect on the reduced mesh size. Although small differences exist, no consistent behavior is observable. Instead, the accuracy τ directly influences the reduced mesh size.

More accuracy requires more elements in the reduced mesh. Also, the higher the ROM dimension k, the larger the selected element set. As the optimization problem scales with the ROM dimension, it becomes more complex with larger dimensions, thus requiring more elements. A general conclusion cannot be made whether k-means or spherical k-means yields a smaller reduced mesh. In all cases, the reduced mesh is for both ROMs in the same size range.

Table 3.3 Summary of global hyper-reduction for an approximation rank of $k = 15$.

# cluster	τ	k-means	spherical k-means
		$\|\tilde{E}\|$	$\|\tilde{E}\|$
3	0.01	485	478
	0.02	365	350
	0.03	306	295
6	0.01	482	479
	0.02	365	360
	0.03	305	299
9	0.01	482	486
	0.02	352	374
	0.03	302	299

Table 3.4 Summary of global hyper-reduction for an approximation rank of $k = 30$.

# cluster	τ	k-means	spherical k-means
		$\|\tilde{E}\|$	$\|\tilde{E}\|$
3	0.01	572	574
	0.02	449	455
	0.03	379	386
6	0.01	574	593
	0.02	456	457
	0.03	385	389
9	0.01	553	574
	0.02	439	456
	0.03	371	395

Table 3.5 Summary of global hyper-reduction for an approximation rank of $k = 40$.

# cluster	τ	k-means	spherical k-means
		$\|\tilde{E}\|$	$\|\tilde{E}\|$
3	0.01	615	601
	0.02	494	484
	0.03	423	412
6	0.01	612	614
	0.02	487	497
	0.03	429	424
9	0.01	607	616
	0.02	482	500
	0.03	417	432

Next, the results of the global-local hyper-reduction approach are tabulated in Tab. 3.6 and Tab. 3.7 for the ROM with $k = 15$. The global-local hyper-reduction results for the ROMs with $k = 30$ and $k = 40$ are tabulated in Appendix A.2 in tables A.1 - A.4.
As the local-local approach is not possible for path-dependent materials, this work does not consider it. Compared to the global-global approach, the global-local approach scales with the ROM dimension k, with the accuracy τ and the number of clusters. The reduced mesh for the k-means ROM with 3 clusters, $k = 15$, and $\tau = 0.01$ is 755 compared to 485. However, the real accuracy after adjusting the weights in each cluster is around 0.5% in each cluster. The clusters with a high offline approximation error, e.g. cluster 2 in the k-means ROM with 3 clusters or the last cluster in spherical k-means clustering, which is always the initial cluster, show the lowest ratio of zero elements α_i. Generally, the ratio of zero elements compared to the total number of elements in the reduced mesh is above 0.5. However, due to plasticity, these elements need to be evaluated to keep the history information in case an element is reactivated, as can be seen later for some elements in Fig. 3.41 and Fig. 3.42.

Table 3.6 Summary of global-local hyper-reduction for ROMs with 3 and 6 clusters and an approximation rank of $k = 15$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\lvert\tilde{E}\rvert$	α_i	τ_i	$\lvert\tilde{E}\rvert$	α_i	τ_i
3	0.01	1	755	0.88	0.509%	716	0.79	0.581%
	0.01	2	755	0.65	0.454%	716	0.92	0.603%
	0.01	3	755	0.84	0.550%	716	0.42	0.237%
	0.02	1	541	0.81	1.052%	530	0.77	1.233%
	0.02	2	541	0.63	1.027%	530	0.84	1.245%
	0.02	3	541	0.74	1.189%	530	0.43	0.588%
	0.03	1	447	0.78	1.715%	420	0.76	1.922%
	0.03	2	447	0.64	1.697%	420	0.81	1.945%
	0.03	3	447	0.71	1.706%	420	0.46	0.883%
6	0.01	1	923	0.67	0.422%	918	0.72	0.494%
	0.01	2	923	0.64	0.355%	918	0.71	0.400%
	0.01	3	923	0.50	0.431%	918	0.39	0.365%
	0.01	4	923	0.62	0.482%	918	0.67	0.491%
	0.01	5	923	0.60	0.449%	918	0.45	0.396%
	0.01	6	923	0.49	0.334%	918	0.33	0.228%
	0.02	1	681	0.64	0.832%	659	0.66	1.009%
	0.02	2	681	0.64	0.574%	659	0.68	0.818%
	0.02	3	681	0.50	0.732%	659	0.44	0.670%
	0.02	4	681	0.61	0.748%	659	0.66	0.779%
	0.02	5	681	0.56	0.839%	659	0.48	0.657%
	0.02	6	681	0.51	0.715%	659	0.35	0.541%
	0.03	1	573	0.63	1.186%	552	0.61	1.525%
	0.03	2	573	0.59	0.947%	552	0.65	1.325%
	0.03	3	573	0.50	1.080%	552	0.46	1.120%
	0.03	4	573	0.61	1.034%	552	0.67	1.027%
	0.03	5	573	0.50	1.228%	552	0.48	0.917%
	0.03	6	573	0.49	1.229%	552	0.35	0.936%

Table 3.7 Summary of global-local hyper-reduction for ROMs with 9 clusters and an approximation rank of $k = 15$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\lvert\tilde{E}\rvert$	α_i	τ_i	$\lvert\tilde{E}\rvert$	α_i	τ_i
9	0.01	1	1017	0.50	0.354%	1026	0.30	0.447%
	0.01	2	1017	0.61	0.383%	1026	0.59	0.346%
	0.01	3	1017	0.44	0.335%	1026	0.60	0.373%
	0.01	4	1017	0.50	0.388%	1026	0.37	0.442%
	0.01	5	1017	0.48	0.514%	1026	0.51	0.512%
	0.01	6	1017	0.34	0.205%	1026	0.56	0.413%
	0.01	7	1017	0.44	0.333%	1026	0.35	0.428%
	0.01	8	1017	0.39	0.403%	1026	0.48	0.397%
	0.01	9	1017	0.50	0.403%	1026	0.30	0.197%
	0.02	1	769	0.49	0.532%	772	0.32	0.687%
	0.02	2	769	0.58	0.759%	772	0.59	0.561%
	0.02	3	769	0.44	0.746%	772	0.58	0.749%
	0.02	4	769	0.43	0.740%	772	0.39	0.705%
	0.02	5	769	0.47	0.731%	772	0.49	0.753%
	0.02	6	769	0.35	0.483%	772	0.51	0.732%
	0.02	7	769	0.43	0.538%	772	0.38	0.613%
	0.02	8	769	0.39	0.630%	772	0.50	0.613%
	0.02	9	769	0.51	0.610%	772	0.31	0.506%
	0.03	1	654	0.48	0.690%	655	0.34	1.002%
	0.03	2	654	0.56	1.063%	655	0.56	0.786%
	0.03	3	654	0.44	1.017%	655	0.57	1.039%
	0.03	4	654	0.39	0.913%	655	0.41	0.889%
	0.03	5	654	0.46	0.923%	655	0.50	0.928%
	0.03	6	654	0.34	0.748%	655	0.46	1.212%
	0.03	7	654	0.41	0.802%	655	0.39	0.824%
	0.03	8	654	0.41	0.844%	655	0.49	0.767%
	0.03	9	654	0.48	0.800%	655	0.31	0.771%

3.3.5 Hyper-reduction Online Results

Given the offline results before, we pick parameters with good accuracy and a reasonable reduced mesh size and investigate the behavior of the ROM for the high deformation and low deformation test cases in detail. The parameters we choose are $n_c = 6$, with a reduced dimension $k_i = 15$, for $i = 1, .., n_c$ and a tolerance $\tau = 0.02$ for hyper-reduction. Fig. 3.39 shows the temporal error for the hyper-reduced k-means ROM and the spherical k-means ROM. All ROMs show a cumulative error behavior. That is, the error accumulates and increases along the simulation time until the rebound phase. The rebound phase is reached earlier in the low deformation test case than in the high deformation test case. For the high deformation test case, the spherical k-means ROM and the k-means ROM have a similar error evolution during the first $15\,\mathrm{ms}$. Afterward, the spherical k-means yields a smaller error. This is underlined by the offline results, where differences in the assignment of snapshots to different clusters later in the simulation were pointed out. In addition, spherical k-means clustering was motivated by the nearly monotonically growing magnitude of the state vector. In the low deformation test case, the k-means ROM has a lower increase in error after $15\,\mathrm{ms}$. Fig. 3.40 investigates the error evolution of the k-means ROM Fig. 3.40a and the spherical k-means ROM Fig. 3.40b in detail by relating the error to the chosen clusters. ROM denotes the cluster chosen by the ROM during the online phase, and it is compared to the cluster chosen by the clustering algorithm of the snapshots from a FOM simulation of the test case, which is denoted by

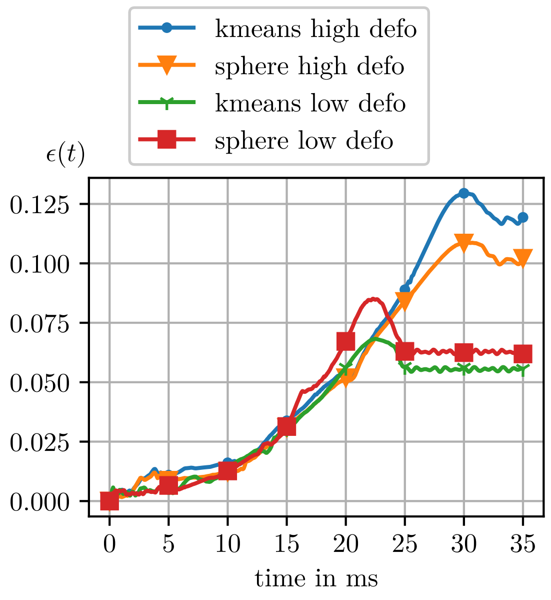
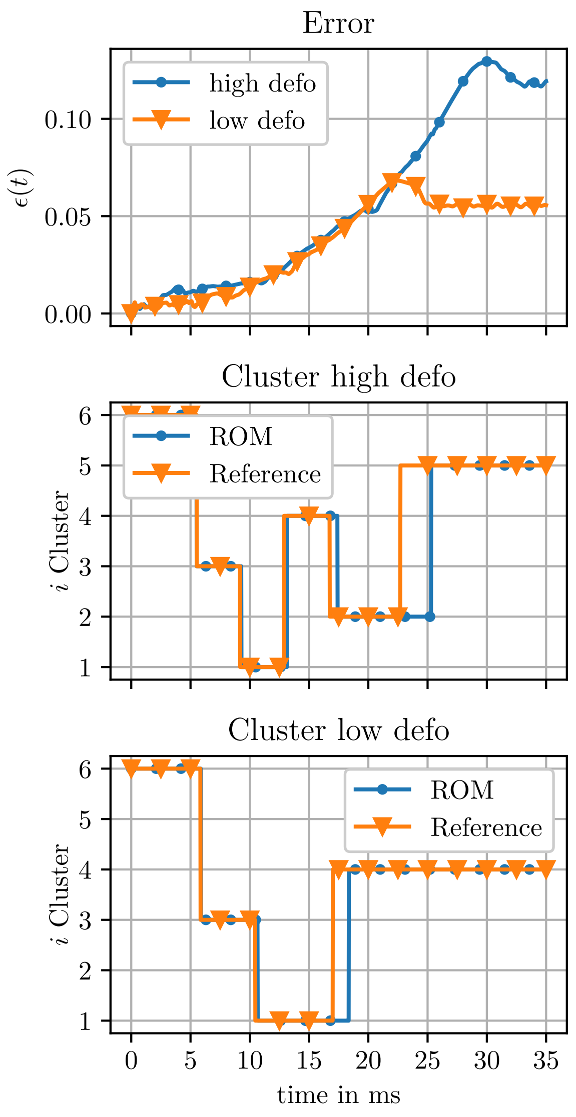
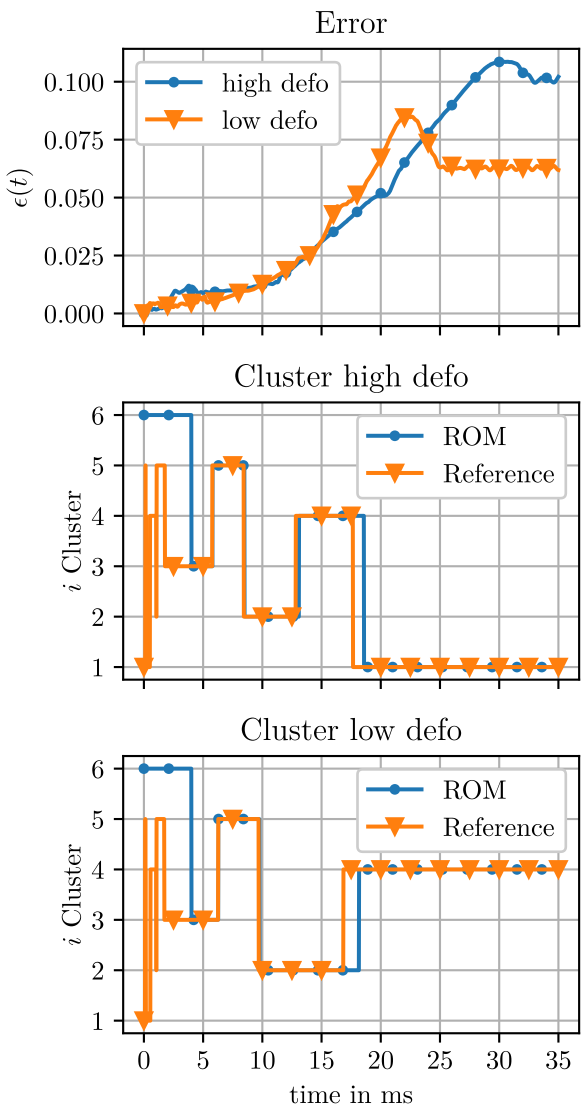


Figure 3.39 Comparison of temporal error $\epsilon(t)$ for the hyper-reduced local ROMs, for the high deformation and low deformation test case.

Reference. The error of the low deformation test case is smaller than that of the highly deforming test case. As the total deformation is lower and the error is cumulative, less total error has evolved. In general, the error always results in a stiffening of the considered example, as can be seen in Fig. 3.43 and Fig. 3.44 for the crash box. As the reduced representation involves an error, the required next deformation state cannot be described precisely, resulting in a stiffening of the solution. As the clusters are partly related to the temporal axis, due to the distance-based clustering and the nature of the solution, the error is also manifested by a later cluster change. Similarly, the low deformation test case changes the clusters later in time than the high deformation test case. After a certain time span, when the solutions reach different points in state space, the cluster choice of the ROM is different. In general, it is difficult to establish a direct connection between cluster choice and error growth. However, a relation between the change to cluster 4 and the increased error growth can be observed in the spherical k-means ROM applied to the low deformation test case. Also, it is difficult to determine the exact cause of the error in a hyper-reduced ROM, as the reduced mesh or the basis approximation could be responsible for it.



(a) k-means ROM



(b) Spherical k-means ROM

Figure 3.40 Accuracy of the k-means ROM and spherical k-means ROM with $n_c = 6$, $k = 15$, and $\tau = 0.02$. The error evolution is related to the chosen clusters and compared to the ideal clusters, obtained by the clustering of the unreduced simulation.

Next, the reduced mesh for the global-local approach is visualized in Fig. 3.41 for the k-means ROM and in Fig. 3.42 for the spherical k-means ROM, applied to the high deformation test case. As the weights of the reduced mesh change during the online phase, the results are shown for t =0 ms, 15 ms, and 35 ms with the current deformation state. The reduced mesh is global, however, it contains zero elements. The zero elements are colored grey, while non-zero elements are colored according to the colorbar. For the k-means ROM, a shift of non-zero elements can be observed. At the t =0 ms, many zero elements are located at the bottom. As ECSW selects elements based on their work and the nodal displacements in the bottom region are zero at the beginning of the simulation, they contribute no work and are therefore not selected. As time progresses, elements closer to the bottom deform and are increasingly assigned non-zero weights. The same behavior can be observed in the spherical k-means ROM, however, in the beginning of the simulation, elements are more evenly distributed across the height of the crash box.

To illustrate the accuracy values shown in the graphs before, Fig. 3.43 and Fig. 3.44 show the results for the test case with high deformation and low deformation, respectively. Both figures show the solutions

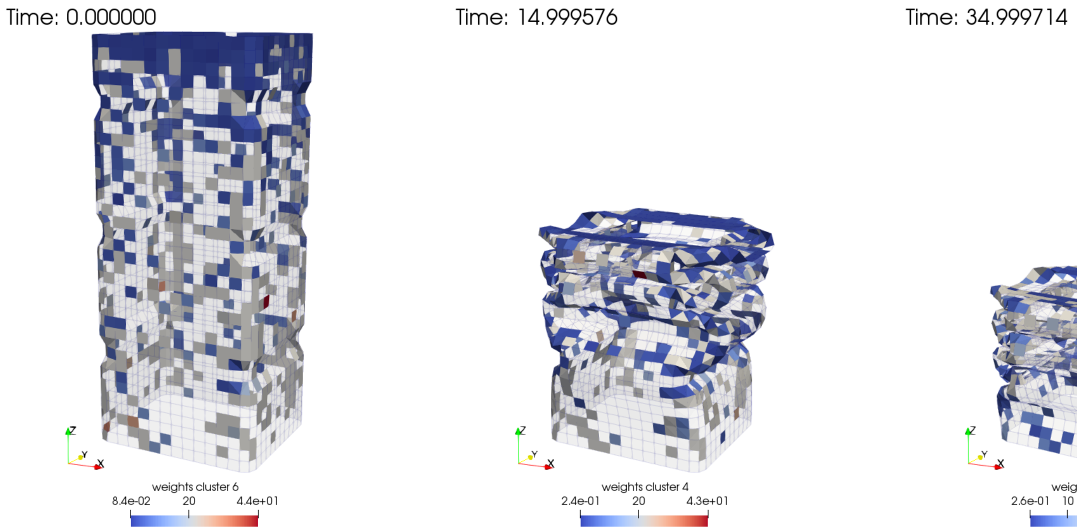


Figure 3.41 Simulation results of the k-means ROM with 6 clusters, $k = 15$ and $\tau = 0.02$ for the crash box example in the high deformation test case with reduced mesh at the times 0 ms, 15 ms, and 35 ms.

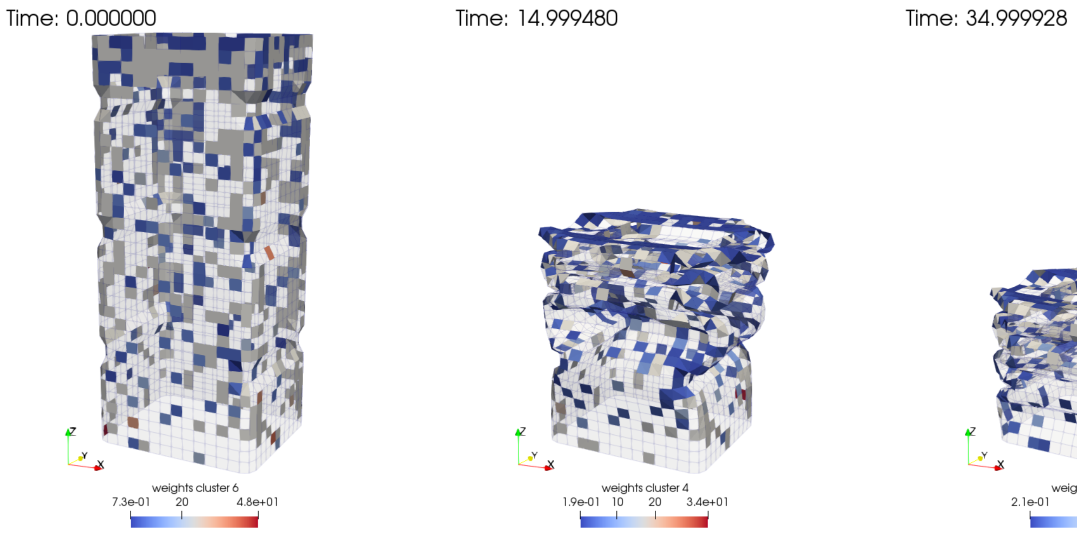


Figure 3.42 Simulation results of the spherical k-means ROM with 6 clusters, $k = 15$ and $\tau = 0.02$ for the crash box example in the high deformation test case with reduced mesh at the times 0 ms, 15 ms, and 35 ms.

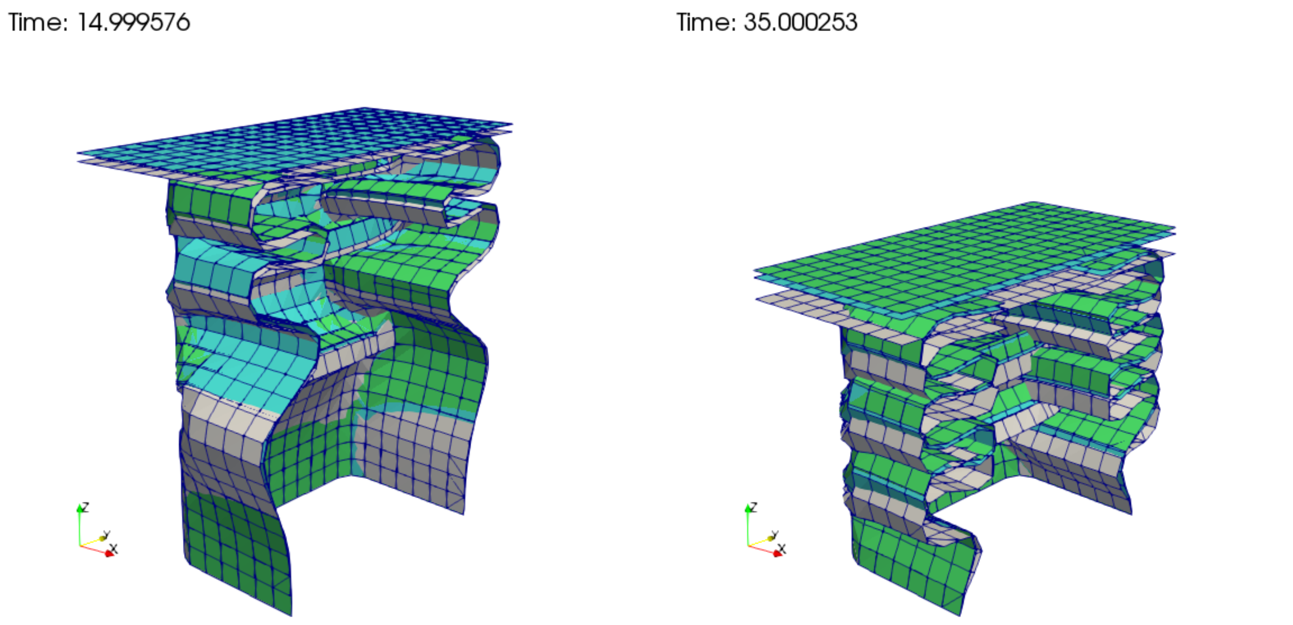


Figure 3.43 Simulation results for the high deformation test case. The original model is colored grey, the k-means ROM is green, and the spherical k-means ROM is blue.

of the reference solution in grey, the k-means ROM in green, and the spherical k-means ROM in blue. The solution is plotted at t =15 ms and at the termination time t =35 ms. The error manifests itself as a stiffening of the model. The deformation approximation by the ROB is not exact and involves small errors (see offline phase). During the online phase, the ROB has to provide the necessary deformation required to proceed to the next time step. If the next required deformation state cannot be achieved exactly, this is reflected in the stiffening mentioned above. Therefore, in Fig. 3.43, the final deformation at t =35 ms of the spherical k-means ROM in blue is larger than the final deformation of the k-means ROM in green. This can be seen by the green plate being above the blue one. Both models deformed less than the reference solution due to the introduced error. The opposite can be observed in the low deformation test case in Fig. 3.44 at the final time. There, the blue plate is above the green plate.

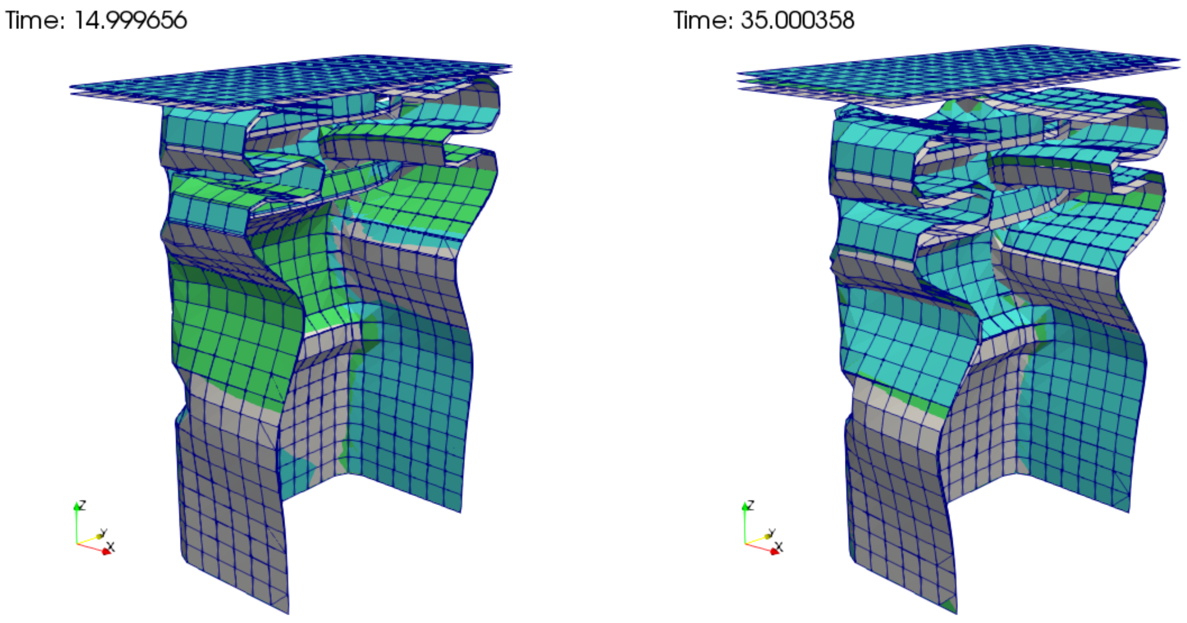


Figure 3.44 Simulation results for the low deformation test case. The original model is colored grey, the k-means ROM is green, and the spherical k-means ROM is blue.

3.3.6 Hyper-reduction Online Results - Parameter Study

In a parameter study, we investigate how the two ROMs behave for different approximation ranks k, numbers of clusters n_c, hyper-reduction approaches, and hyper-reduction accuracy τ, tested on both test cases. The approximation rank is chosen as $k = 15, 30, 40$, the number of clusters is $n_c = 3, 6, 9$, the global-global and global-local strategies are applied, and the accuracy is $\tau = 0.01, 0.02, 0.03$. The average accuracy ϵ is calculated for each combination of the parameters and evaluated in the following figures. Fig. 3.45 compares the accuracy of the k-means ROM and spherical k-means ROM for both test cases in each subplot. The individual subplots vary the number of clusters, the hyper-reduction accuracy and strategy. Except in some cases, the error decreases with increasing dimensions k. For the spherical k-means ROM with $n_c = 3$, $\tau = 0.01$, and the global-local approach, the error increases again between $k = 30$ and $k = 40$. However, the increase in error is small, as can be seen in the scale of the plot. In general, the global-local approach is more accurate than the global-global one due to the higher real accuracy caused by the weight adjustment. This can also be seen in Fig. 3.46 where in each subplot, both hyper-reduction approaches are analyzed for each accuracy. For the ROMs with $n_c = 6, 9$, the global-global approach is strictly less accurate than the global approach. Also, in Fig. 3.46, it can be seen that for some ROMs, the error does not decrease or even increase from $k = 30$ to $k = 40$. A possible explanation is the accuracy of the underlying non-hyper-reduced ROM combined with the larger optimization problem for hyper-reduction due to the higher dimension. It is not always guaranteed that the error decreases for an increasing number of dimensions, as can be seen by the results of the ROMs without hyper-reduction (Fig. 3.36 and Fig. 3.38). Assuming the error does not decrease significantly, the larger optimization problem will yield a different solution regarding the error norm, which is also influenced by the problem's size. It is not guaranteed that the online phase yields more accurate results in a parametric example.
Finally, Fig. 3.47 evaluates the error of one ROM regarding one test case (e.g., spherical k-means ROM with global-global approach) for all possible parameters (number of clusters, tolerance τ and dimension k). The ROMs with a low number of clusters $n_c = 3$ and a low hyper-reduction tolerance $\tau = 0.02, 0.03$ yield high error, especially for the low deforming test case. A comparison of the ROMs error using the same y-axis scale can be found in Appendix A.3.

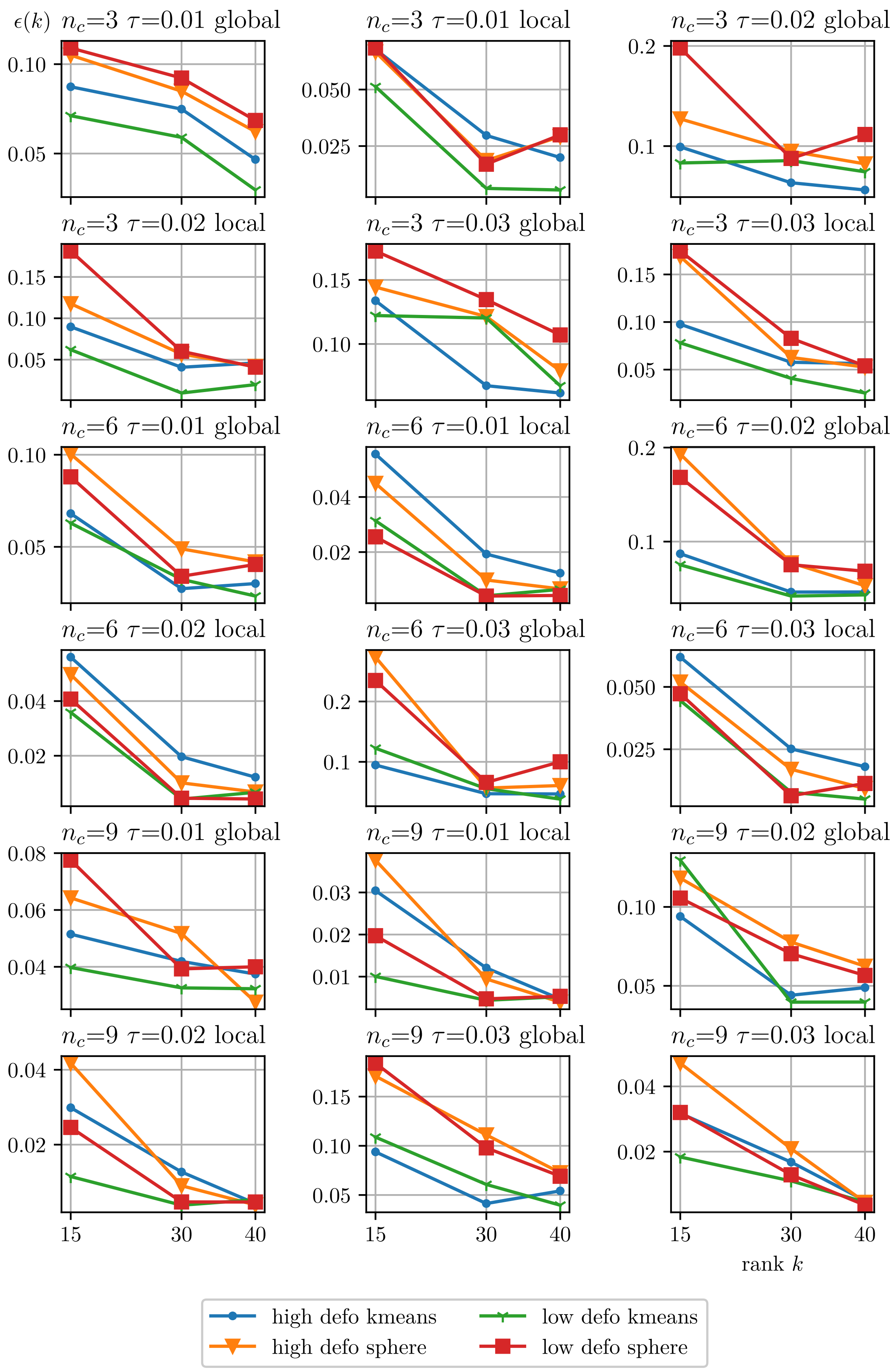


Figure 3.45 Hyper-reduction comparison between the k-means ROM and spherical k-means ROM for both test cases.

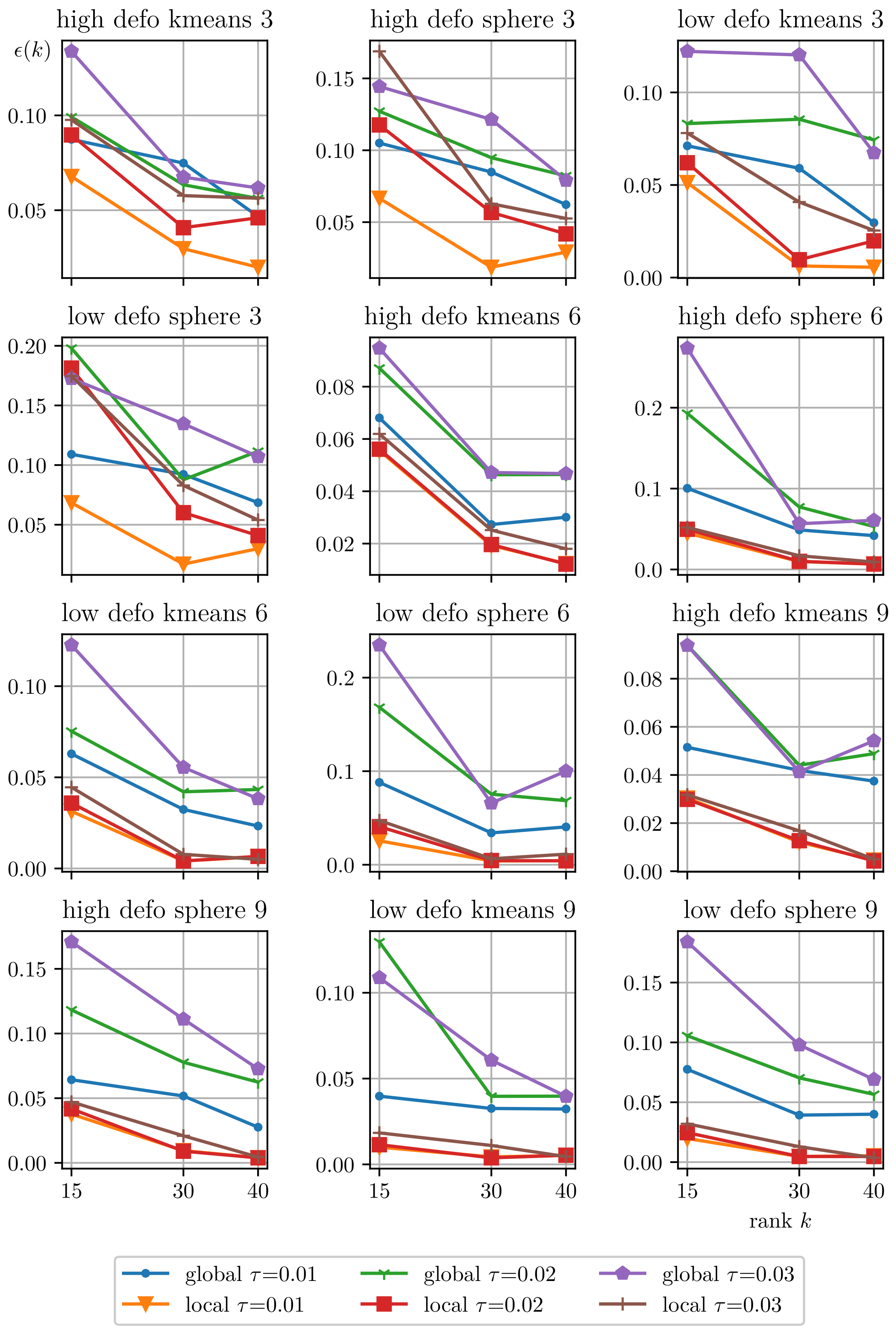


Figure 3.46 Hyper-reduction comparison of the global-global approach versus the global-local approach.

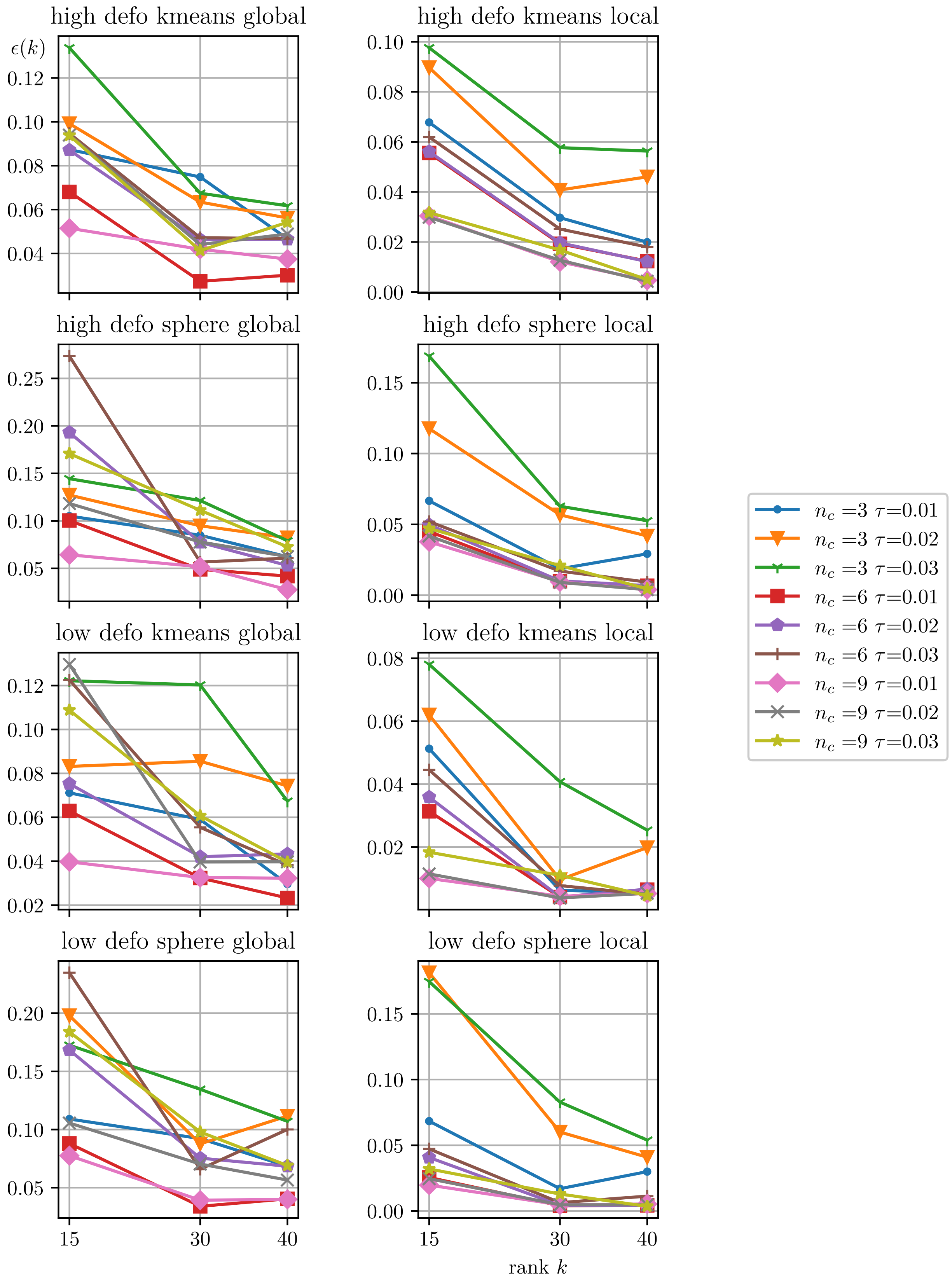


Figure 3.47 Comparison of all cluster numbers and tolerances τ for one hyper-reduced ROM for one test case in each subplot.

3.3.7 Computational Speedup

This section discusses the computational speedup for the chosen parameter configuration of the previous section. Although smaller reduced meshes were shown in the parameter study, an estimate of the simulation time for other reduced mesh sizes is easily possible. Another restriction is the limited source code access in this study. A more efficient implementation would have been possible with full source code access. Nevertheless, we provide simulation times achieved using 4 threads on an Intel(R) Xeon(R) Gold 6254 CPU with 32GB of RAM. The results are obtained by taking the mean of 15 identical simulations of the high deformation test case. The results are tabulated in Table 3.8. The FOM/reference simulation

Table 3.8 Mean simulation time of 15 runs for each model.

Model	Simulation time s
Reference	132.93
K-means ROM	93.27
Spherical k-means ROM	90.6

requires $132.93\,\mathrm{s}$. In comparison, the hyper-reduced k-means ROM runs $93.27\,\mathrm{s}$, corresponding to a reduction in simulation time by 30%. The spherical k-means ROM is slightly faster with a simulation time of $90.6\,\mathrm{s}$, corresponding to a reduction of 32%. The decrease in simulation time is mainly based on the reduced number of evaluated elements. The full model has a total number of 1,864 elements. The reduced mesh size of the k-means ROM is 681 elements, and the size of the spherical k-means ROM is 659 elements. This equals a mesh size reduction of 63% and 65%, respectively. The discrepancy between the reduction in mesh size and computation can be explained by considering the solver-generated statistics. First, the rigid plate is not considered for reduction, and second, shell element processing causes only 52% of the total simulation time. The remainder is occupied by contact evaluation, the additional effort of the projection steps in each time step, I/O operations, and miscellaneous other functions. A further influence on the discrepancy of simulation time between the two ROMs is the evaluation of a different distance function.
For completeness, we want to mention the increased critical time step. Although this work does not consider the probably larger critical time step of the ROMs, it motivates further research in this direction. A larger time step affects all parts of the solver and results in a nearly 1:1 relationship between the increase in critical time step and reduction in computation time. The evaluation of the critical time step would require additional source code access and is therefore left for future research. Further research could also investigate the reduction of zero elements in the global-local approach. Zero elements currently comprise approximately half of all elements in the reduced mesh. Interpolating the missing history variables using a gappy-POD approach could enable further speedups.

3.4 Summary and Discussion

This work concerns extending pROMs for nonlinear solid dynamics from reproductive to parametric applications. We emphasize the necessity of different ROM approaches by experimentally demonstrating the inability to construct an efficient ROM using a global approach. In addition, a comparison between ROMs that combine displacements and rotations and ROMs that treat them separately indicates that a combined treatment is preferable in the offline and online phases of parametric problems. Once the ROM architecture is fixed, the transition from reproductive to parametric examples begins by comparing the offline accuracy with the error decay for the ROBs. In the offline phase, the global basis requires more dimensions to reconstruct the data. In contrast, a ROB, which is trained on snapshots from one parameter point and approximates the same snapshots afterward, is more tailored to the specific simulation than a global basis. This can be seen in the much faster error decay and lower approximation error generally. Also, differences

in the approximation error exist for displacements and rotations. Rotations exhibit a higher approximation error and a slower error decay than displacements. This observation is amplified in the parametric example. However, a combined treatment already in the offline phase can reduce the approximation error (regarding the defined error norm) and yield a reasonable approximation. The error of the combined ROB is higher than the displacement-only basis but smaller than the rotation-only basis. The first basis vectors of the ROBs are visualized, whereby the main deformation modes can be seen. A higher mode is also shown, in which significantly higher frequencies become visible in space. These higher frequencies are required to accurately reconstruct the deformation state as the superposition of modes. In a parametric example, more modes are required to cover the range of possible deformation states, which results in higher-dimensional ROMs.
Next, the results of the offline phase must be confirmed in the online phase of the high deformation test case. The temporal average of the time-dependent error is considered in the online phase. As in the offline phase, the error decreases rapidly for an increasing number of dimensions in a reproductive example. Also, the combined treatment of displacements and rotations is confirmed, as a combined ROM with $k = 30$ yields the same error as the displacement-only ROM with 30 dimensions. For a full ROM, rotations must be considered in addition. Next, the global ROM using the global ROB is applied to the high deformation test case. Similarly, the combined ROM yields for $k = 60$ the same accuracy as the separated ROM with more dimensions. However, a slower error decay is observed. The slower error decay requires retaining more modes in the ROM to achieve high accuracy. However, a high-dimensional ROB increases the complexity of hyper-reduction and potentially alleviates the increase in the critical time step due to modes with higher frequencies. Therefore, new approaches must be utilized to provide accurate but low-dimensional ROMs.
We utilize the lROB approach and extend it to spherical k-means clustering. lROB divides the solution space first into distinct subregions, also called clusters. The standard method uses k-means clustering based on the L^2 norm. In crash and impact models, the state vector usually increases nearly monotonically in time, leading to a state space division along the time dimension. We argue that this is suboptimal and introduce direction-based clustering, which is spherical k-means clustering. First, the offline phase of both clustering methods is investigated. It consists of the clustering itself and the accuracy of the created lROBs. k-means and spherical k-means mainly differ because clusters appearing early in the simulation are smaller, and later clusters are larger. That is, the k-means-based ROM changes clusters towards the end of the simulation less often than the spherical k-means-based ROM. Before the lROBs are created, the nearest snapshots are added to ensure a smooth transition between the clusters. Snapshots are added at the borders of the clusters in both methods. However, both methods are sensible towards the end of the simulation and add snapshots from other parameter configurations instead of snapshots at the border of the cluster.
After lROBs are created for each cluster, the accuracy is compared to the global basis. While local bases associated with clusters containing snapshots from later simulation times yield good approximation error, especially the lROBs associated with the early clusters show larger error than the global basis. As spherical k-means divides the early phase of the simulation into more clusters, especially the ROMs with many clusters have more lROBs with a larger offline error. However, the online phase shows that the offline error is not directly related to the time-dependent error of the ROM, as no excessive error growth can be observed during the early phases of the simulation. Before hyper-reduction, both ROMs are tested on a high deformation test case, and a low deformation test case for different numbers of clusters and approximation ranks k. Compared to a global ROM, both local methods show a significantly higher accuracy with the same number of dimensions. Also, we observe a much faster error decay with increasing dimensions. While the error decay is slower for the high-deformation test case than for the low-deformation test case, especially the k-means ROM has convergence problems to a low error in the low deformation test case. The spherical k-means ROMs show a smoother error decay in general and a faster error decay in the low deformation test case, compared to the k-means ROM.
Finally, hyper-reduction is applied in two variants, the global-global and the global-local approach. The reduced mesh size mainly governs the speedup of the ROM. In the global-global approach, the mesh size is

mainly influenced by the tolerance τ and the dimension of the ROM. In contrast, the introduced global-local approach also scales with the number of clusters n_c, resulting in a larger reduced mesh given a tolerance τ. However, the local approximation's real accuracy τ_{real} is higher. The online phase analysis is detailed for a k-means and spherical k-means ROM with $k = 15$, $\tau = 0.02$, and $n_c = 6$. The spherical k-means ROM yields a lower error for the high deformation test case for which it was designed. The hyper-reduced k-means ROM yields a smaller error for the low deformation test case. The increase in error of the spherical k-means ROM towards the end can be related to a cluster change. However, in a hyper-reduced ROM, the exact error cause is unclear. Either it is due to the IROB, or the reduced weight set is unsuitable for the current simulation state.

A final hyper-reduction study compares the two ROMs for different ranks k, tolerances τ, hyper-reduction strategies, and number of clusters n_c. Although the global-global strategy yields smaller reduced-meshes, the global-local approach yields smaller online error, given that the real tolerance is lower. For the accurate, not hyper-reduced local ROMs, the hyper-reduced ROMs show a consistent error behavior. The error decreases with increasing rank, tolerance, or cluster number. However, there is no convergence guarantee in parametric problems, as some ROMs show a slightly higher error for the ROM with $k = 40$ compared to the ROM with $k = 30$. A possible explanation is that an increase in dimensions of the non-hyper-reduced ROM does not always lead to an increase in accuracy. However, the hyper-reduction optimization problem becomes larger, and the chosen error norm could lead to a worse approximation of the single quantities. Also, overfitting could be an additional reason for the slightly higher error.

Finally, the computational speedup is strongly coupled to the reduced mesh size. This work does not consider the speedup due to the larger stable time step. We demonstrate a reduction in mesh size by 60%, resulting in 30% less computation time due to element processing accounting for only 52% of the total time. There is still potential for further computational speedup, which promotes further research. Overcoming the source code limitation present in this work would enable the computation of the local ROM's stable time step. The adjustment of the stable time step affects the whole simulation directly. Also, the local weight sets contain approximately 50% zero weights. If an element is reactivated again, trying to interpolate the field of internal variables could further reduce the reduced mesh size, and a local-local hyper-reduction approach would be possible for history-dependent materials. Trying to better understand how the offline phase's approximation error correlates to the ROM's accuracy could enable the development of an accuracy-based IROB creation and even lower-dimensional ROMs. Finally, the whole work does not address contact forces. The reduction of contact forces is still an open field of research.

3.5 Conclusion

This chapter introduces parametric pROMs based on IROB. These ROMs enable applications requiring large parameter variations, e.g., optimization. Compared to reproductive examples, global pROMs show a slow error decay. The large variance in the training data makes it difficult for a linear dimensionality reduction method such as POD to find a low-dimensional structure. Minor to no speedups are achieved since hyper-reduction scales with the dimension of the ROM. More sophisticated methods are required to ensure small dimensions. The IROB method is chosen due to its hyper-reduction compatibility, which is essential for explicit nonlinear models. In addition to utilizing the correlation between displacements and rotations, dividing the training data set into subregions ensures again local low-dimensional ROBs. k-means clustering is compared to spherical k-means clustering, a direction-based clustering method. Both clustering procedures yield similar distributions of the snapshots. The resulting approximation error of the ROBs does not necessarily correlate with the accuracy of the associated Galerkin ROM, which is why it was impossible to define an error-based criterion dimension criterion. Nevertheless, all local ROMs show a fast online error decay using fewer dimensions than the global ROM. The requirements for effective hyper-reduction are fulfilled, and a parameter study is conducted. Two hyper-reduction approaches are investigated: the global-global and the global-local approach. Compared to the global-global method, which has a global reduced element set and global weights, the global-local has local weights that are cluster-specific. For a given hyper-reduction tolerance, the global-global approach yields smaller meshes than the global-local approach. Nevertheless, the global-local approach is preferred, as the accuracy of the hyper-reduced ROM is higher compared to the global-global ROMs, and the larger reduced mesh is mainly due to zero elements. Zero elements are elements that are evaluated and afterwards weighted with zero to preserve the history once the local weights change and the elements are activated. Avoiding the evaluation of zero elements using gappy POD or image reconstruction is left for future research. Applying hyper-reduced lROB pROMs yields a reduction of 60% of the elements and a reduction in computation time of 30%. Parametric pROMs are successfully demonstrated for a typical crash example, and the potential for further speedup is still present. Approximately 50% of the reduced mesh are zero elements, the critical time step is usually larger for ROMs and can be increased, and contact forces are not considered in this work. All these open questions provide opportunities for further research to improve the accuracy and computational speedup of parametric ROMs for nonlinear solid dynamics in crash and impact.

4 Partial MOR

The previous chapters derived pMOR and IROB enabled parametric pMOR. This chapter examines a use case of parametric pMOR. After partial MOR is motivated in Section 4.1, Section 4.2 introduces the theory. Section 4.3 introduces the considered model including geometry variations first and highlights the need for parametric pMOR in Subsection 4.3.3 and Subsection 4.3.4 afterwards. In the remainder of the results section, the ROMs are tested in increasing order of complexity. The ROMs are reaching from a global linear basis in Subsection 4.3.5 to different parametric ROMs using IROB with hyper-reduction in Subsection 4.3.6. Finally, the computational speedup is discussed, and the results are concluded.

4.1 Motivation

Often, in modern car development, only a single component of interest is optimized; i.e. only one component in a large model is modified, while the rest remains unchanged. E.g., mechanical properties of the battery pack are optimized without changing the surrounding structure. The adjacent structure influences the modified part and vice versa. To reduce computation time, displacement or force histories are stored and imposed on the optimized part. However, this approach assumes that the model's response changes only slightly and the influence on the surrounding structure is small. The interface displacements or forces do not change due to changes in the modified model. If the previously stated assumption is invalid, significant errors are assumed. Crash problems often include buckling of structures. These instabilities are sensitive to small changes in the boundary conditions. Therefore, a more sophisticated approach is needed that adapts the surrounding structure's response to changes in the response of the modified model.
In this chapter, we choose pMOR to create a physics-based ROM of parts of a structure, while one part of the model is geometrically modified and remains unreduced. A global approach and standard k-means-based local ROB with global hyper-reduction are applied. Different training strategies are assessed to show the method's limitations and highlight the difficulties with sensitive problems. Finally, we assume the theoretically achievable speedup and conclude.

4.2 Theory

Fig. 4.1 shows the interface of two parts, which are directly connected. The blue upper part remains unreduced, while the lower part is considered for POD and ECSW. Multiple parts can be defined as one continuous material. In this figure, one part of the crash box is replaced by different geometries. They are modeled as separate parts in LS-Dyna. However, the parts consist of the same material and elements as the surrounding parts. This is equivalent to a tied-contact condition between the parts. The division of one part into sub-parts is different from real distinct parts. For genuine distinct parts, the nodes at the interface do not belong to two parts simultaneously. The node sets are, therefore, naturally separated and can be assigned to the respective parts. In our case, we decide to consider the nodes at the boundary for POD, as can be seen in Fig. 4.1.
ECSW hyper-reduction works on the unassembled forces and moments. As each element is assigned to one part, we must take no additional care in dividing the elements. The hyper-reduction problem is formulated using solely the elements in the reduced parts. This approach results in a smaller offline optimization problem. However, ECSW removes fewer elements from the model, which will result in a lower computational speedup. Next, we derive the mathematical formulation for the partial reduction. First,

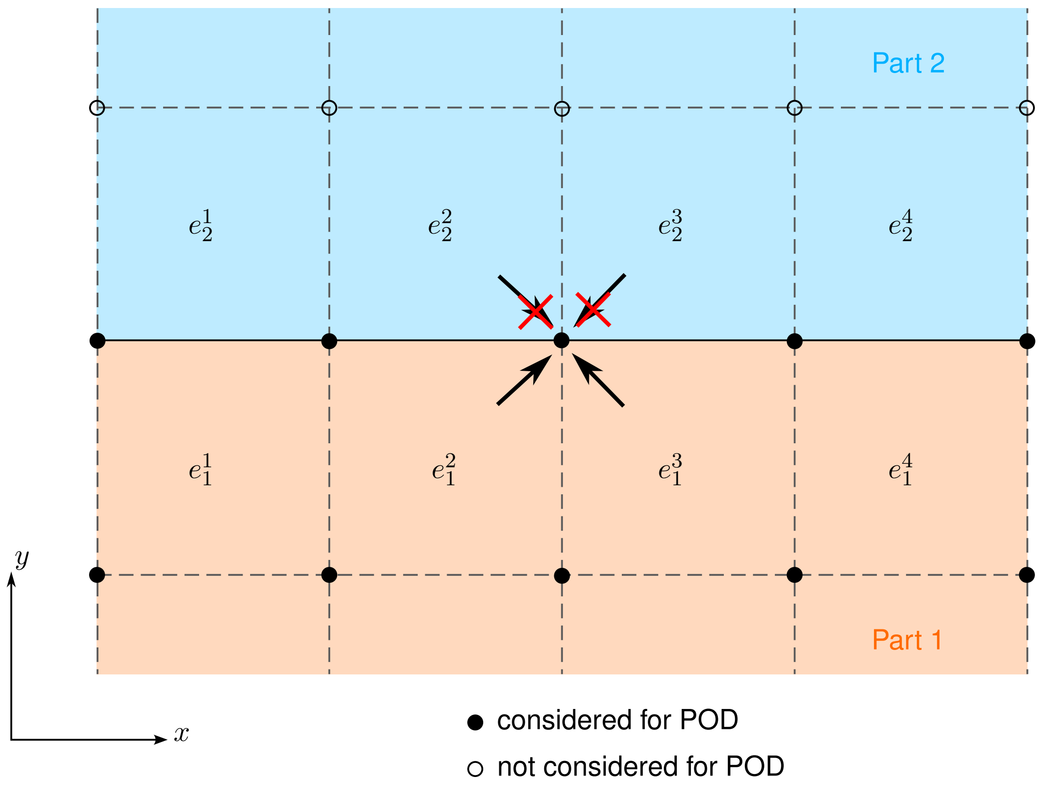


Figure 4.1 Interface nodes of a model consisting of two parts, which are directly connected to each other.

we select the considered nodes by applying the selection matrix $\mathbf{S}_\partial \in \mathbb{R}^{n_\partial \times n}$ to the state vector x. The new state vector $x_\partial \in \mathbb{R}^{n_\partial}$ containing the reduced nodes is:

$$x_\partial = \mathbf{S}_\partial x, \tag{4.1}$$

where the selection matrix assigns one element of the input vector x to a new position in the partial state vector x_∂. The row index of the selection matrix determines the new position of the entry in the state vector x, determined by the column index. That is, if the entry

$$S_{\partial i,j} = 1, \tag{4.2}$$

then

$$x_{\partial i} = x_j. \tag{4.3}$$

Element j of the full state vector x is assigned to element j of the partial state vector j. This is again illustrated here:

$$\begin{bmatrix} x_{\partial 1} \\ x_{\partial 2} \\ x_{\partial 3} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}. \tag{4.4}$$

For ECSW hyper-reduction, we consider the partial internal force vector

$$f_{\partial,int} = \sum_{e \in \mathrm{E}_\partial} \mathbf{L}_{\partial,e} f_{int}^{(e)} \in \mathbb{R}^{n_\partial}, \tag{4.5}$$

where the partial assembly matrix $\mathbf{L}_{\partial,e}$ is obtained as

$$\mathbf{L}_{\partial,e} = \mathbf{S}_\partial \mathbf{L}_e \in \mathbb{R}^{n_\partial \times m_e}. \tag{4.6}$$

The summation over all n_e elements is replaced by the sum over all elements, contained in the reduced parts. The elements of the reduced parts are contained in the set E_∂. Once the correct nodes and elements are assigned to the respective domains (reduced, unreduced), the steps to construct a ROM are equal, as described in Chapter 3. First, the snapshot matrices for each geometry are compiled containing the partial displacement vector $u_{\partial,i}^{t_{N_s}}$ at the defined time t_j instances:

$$\mathbf{U}_{\partial,i} = \left[u_{\partial,i}^{t_1}, \quad u_{\partial,i}^{t_2}, \quad \ldots, \quad u_{\partial,i}^{t_{N_s}} \right] \in \mathbb{R}^{n_\partial \times N_s}, \quad \text{for } i = 1, \ldots, N_G, \tag{4.7}$$

where N_G is the number of different geometries. The local snapshot matrices can be concatenated in one large snapshot matrix

$$\mathbf{U}_{\partial,\text{glob}} = \left[\mathbf{U}_{\partial,1}, \quad \mathbf{U}_{\partial,2}, \quad \ldots, \quad \mathbf{U}_{\partial,N_G} \right] \in \mathbb{R}^{n_\partial \times N_G N_s}. \tag{4.8}$$

All ROM construction steps are equal to the steps in the previous chapters and the initial variables are replaced by the partial ones, denoted by the ∂ subscript.

4.3 Results

4.3.1 Geometry Variations

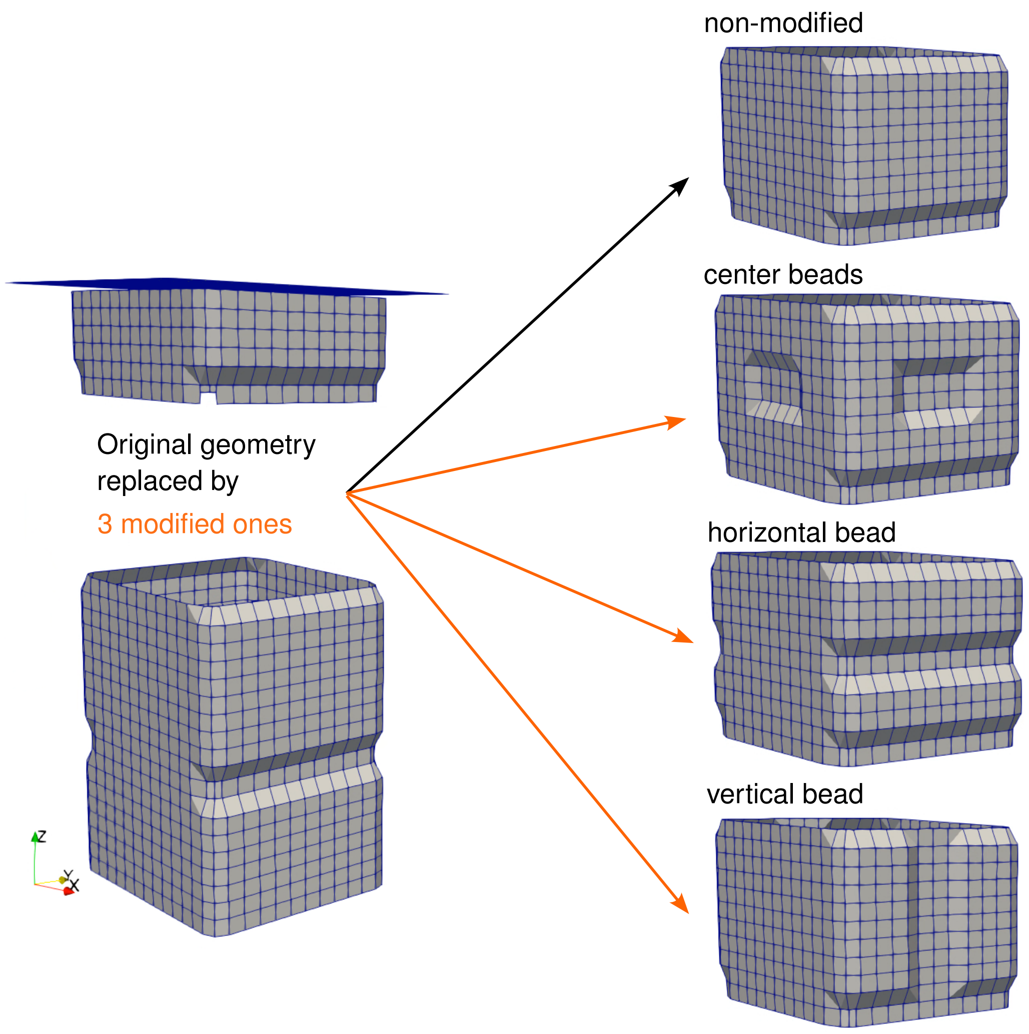


Figure 4.2 Geometry variance.

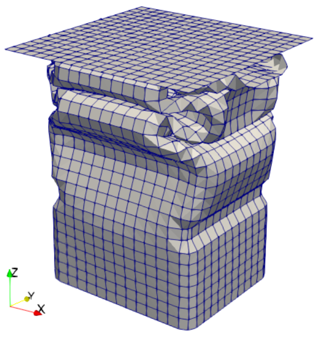
As shown in Fig. 4.2, the crash box geometry is varied in three different kinds. The original geometry is the same as described in Section 3.1.1, with a wall thickness and plate mass equal to the high deformation test case. This test case has a plate mass of 159.833 kg and a wall thickness of 1.717 mm. The plate mass and wall thickness are kept constant for all examples in this chapter. However, different geometry modifications are introduced. The first modification is center beads. The beads with a normal pointing in the x-direction are rectangular. They are centered around the middle of the tube in the y-direction, and the lower edge of the bead has a distance of 30.276 mm to the lower adjacent part. The bead has a depth of 4 mm, a width of 50 mm and a height of 30.276 mm. The bead's depth is reached within one element. The second beads with a normal pointing in the y-direction are centered around the x-axis and are located at the same z-direction position. The dimensions are equal to those of the first beads except for a different width of 48 mm.
The second modification is a horizontal bead. Two beads are placed on the same sides of the tube as the other folding triggers. The lower edge of the bead is placed in a z-distance of 30.276 mm to the lower edge of the adjacent part. The height of the bead is 22.707 mm, and the bead extends across the entire width of the tube. The bead has a depth of 4 mm, which is reached within a distance of one element.
The third modification is a vertical bead. The vertical bead extends across the entire height of the exchanged part. It is placed on the same side as the folding triggers. It is centered around the y-axis and has a width of 32 mm. The bead depth is 4 mm, which is linearly reached within a distance of one element. The geometries are chosen such that the stiffness of the exchanged part is varied, which influences the behavior of the remaining tube. The difference in results is shown in the next section.

4.3.2 Deformed Geometries

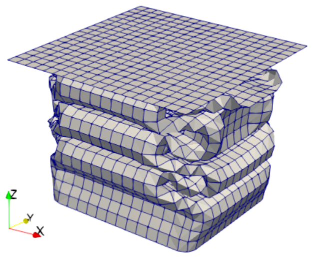
Fig. 4.3 shows the deformation of the three modified geometries at 12 ms and at 35 ms, the termination time of the simulation. Fig. 4.3a and Fig. 4.3b show the deformation of the center beads model. Compared to the non-modified geometry, the buckle directed outwards is affected by the bead. A full folding is prevented due to the increased stiffness. At 12 ms, mainly the area around the bead is affected, and the changes in the solution can be considered local. However, at 35 ms, other folds are also distorted due to the strong folding of the whole model. For instance, the fold with the bead bends the fold below due to the larger height of the bead. The center bead does not affect the buckling order compared to the horizontal bead and the vertical bead. Fig. 4.3c and 4.3d show the deformation of the horizontal bead example at 12 ms and 35 ms, respectively. The final deformation is similar to the non-modified example, and the geometry modifications have a small impact on the final deformation. However, the horizontal bead strongly impacts the deformation history. The first buckle begins to form during the first milliseconds of the simulation. The increased stiffness of the bead prevents the fold from fully developing, and the crash box starts to buckle at the bottom. Once the other folds are fully developed, the remaining energy is sufficient to fully form the fold with the bead. The horizontal bead case highlights the sensitivity of the solution to changes in the geometry of the model. Finally, the vertical bead, as can be seen in Fig. 4.3e and 4.3f, introduces sufficient stiffness to first trigger folding in the bottom region. The vertical bead never bends outwards, as the non-modified example would, and distorts the adjacent parts heavily.
To summarize, the three modifications introduce three levels of impact. The center bead impacts the solution rather locally in the modified area. In contrast, the vertical bead introduces significant stiffness in the modified area, directly affects the reduced parts of the model, and causes a different folding order. The horizontal bead is in between the two other cases. It highlights the sensitivity of the crash box problem by still triggering a different folding order but ending in the same final deformation state as the non-modified model.

4.3.3 Reproductive Example (PoC)

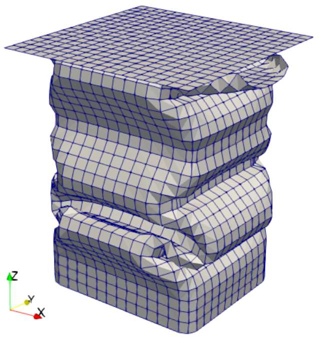
We begin with a PoC of partial MOR. The non-modified geometry is used to generate the training data for displacement and force data. Snapshots are taken every 0.01 ms, resulting in a total of 3,501 snapshots



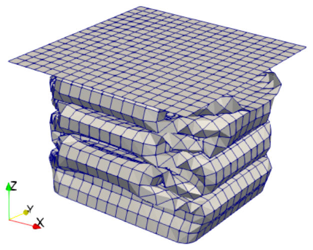
(a) Geometry with center bead at 12 ms.



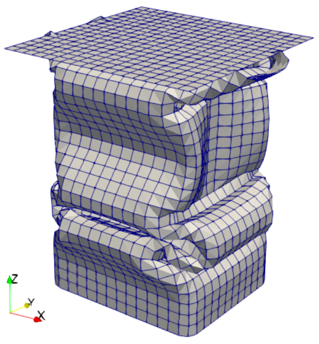
(b) Geometry with center bead at 35 ms.



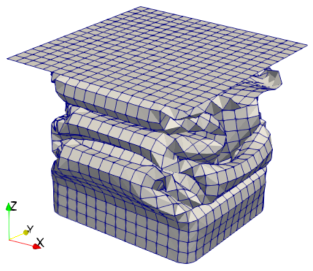
(c) Geometry with horizontal bead at 12 ms.



(d) Geometry with horizontal bead at 35 ms.



(e) Geometry with vertical bead at 12 ms.



(f) Geometry with vertical bead at 35 ms.

Figure 4.3 Deformation of modified geometries at 12 ms and 35 ms.

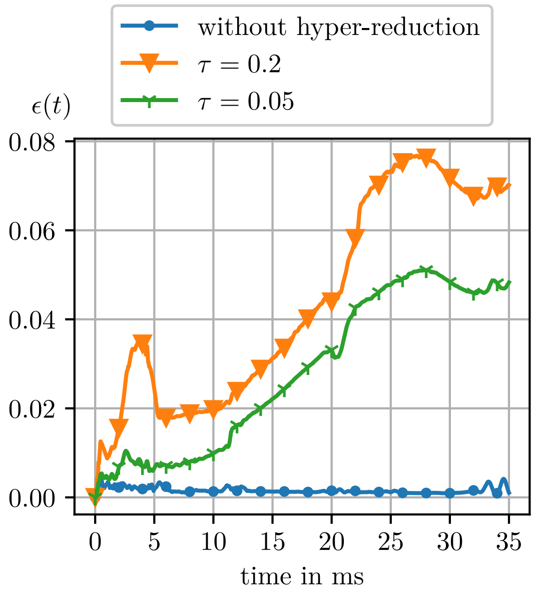
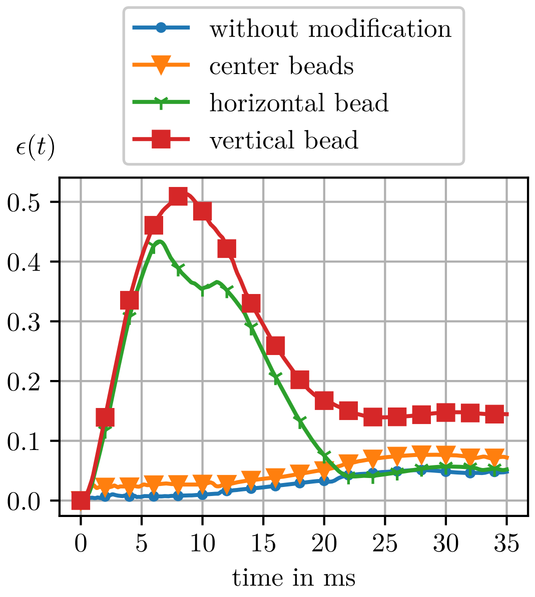


Figure 4.4 PoC of the partial MOR using a reproductive example.

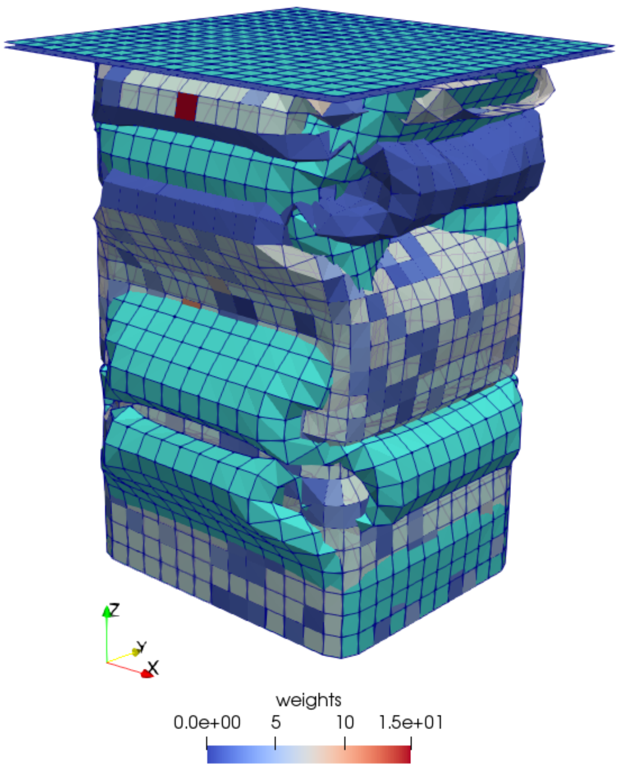
for displacements and forces each. We choose $k = 30$ as the dimension for the ROM. In addition to a non-hyper-reduced ROM, we also construct two hyper-reduced ROMs with tolerances of $\tau = 0.2$ and $\tau = 0.05$. The crash-box without the modified part consists of 1,292 elements, and the modified part consists of 572 elements. The reduced mesh for a tolerance of $\tau = 0.2$ has a size of 246 elements, and the reduced mesh for $\tau = 0.05$ consists of 410 elements. The time-dependent error of the three ROMs for the non-modified example is depicted in Fig. 4.4. The error increases along the simulation time, and the maximum error is small for all ROMs. Hyper-reduction adds additional error, which decreases with decreasing tolerance τ. Summarized, all ROMs show good accuracy with a small dimension of the ROM. A small error was expected as Chapter 3 shows good accuracy for low-dimensional ROMs in reproductive examples. Next, the geometries are modified.

4.3.4 ROM Without Modification

The motivation behind partial MOR is to replace simple boundary conditions as prescribed displacements or forces by ROMs that describe the behavior of the surrounding structure more accurately. In this subsection, we investigate the influence of the geometry modifications on the remaining structure and whether a ROM purely trained on the non-modified geometry is capable of predicting the structure's behavior with these modifications. The hyper-reduced ROM with $k = 30$ and $\tau = 0.05$ is tested on the modified geometries. This ROM is purely trained on the results of the non-modified geometry. The time-dependent error for a simulation time of 35 ms can be seen in Fig. 4.5a. The error for the example without modifications is transferred from Fig. 4.4 and serves as a reference. As observed in Subsection 4.3.2, the changes in the solution of the center beads model are predominant in the vicinity of the geometry variation. Only towards the end of the simulation time the beads influence other parts of the crash tube due to folding. However, the beads cause small changes in the overall solution, which is reflected in the small error of the ROM for the center beads model. Both models that influence the folding order cause large errors during the simulation time. Towards the termination time, the horizontal bead model fully folds, and the horizontal bead does not affect the final deformation. Also, the ROM fully deforms and shows small errors towards the end, although the ROM folds in the wrong order. The ROM fails for the vertical bead modes, as it yields a different folding order and final deformation shape. The wrong folding order is illustrated in Fig. 4.5b. The turquoise crash box is the reference solution, and the other crash box is the ROM solution. The ROM is colored as follows:



(a) Time-dependent error $\epsilon(t)$ of the hyper-reduced ROM with $k = 20$ and $\tau = 0.05$, applied to all geometries.



(b) Hyper-reduced ROM applied to the vertical bead example at 10 ms. Reference solution in turquoise and ROM in grey with elements colored according to their weights.

Figure 4.5

missing elements are colored in grey, and the remaining elements are colored according to their weights. The rigid plate and the non-reduced parts are colored as elements with a weighting factor of 1.0. The reference solution starts to buckle at the bottom while the ROM buckles first at the non-reduced part and in the wrong direction. The ROB of the ROM is not possible to provide the correct boundary conditions for the vertical bead as the impact of the geometrical modification onto the ROM is severe. Therefore, more training data must be included to provide a richer ROB for a more accurate ROM.

4.3.5 Global Approach

After the ROM, which was purely trained on the non-modified geometries and tested on modified ones, the next level of complexity is to include the results of the geometry modifications as training data in a global sense. The resulting training data consists of 3,501 snapshots for four simulations, including six DoF for 1,400 considered nodes in the model. A global ROB is constructed by concatenating all snapshots in one snapshot matrix. Fig. 4.6 shows the time-dependent error for the ROM with $k = 30$ and $k = 60$. Inserting more variance in the snapshot matrix by including snapshots from multiple different solutions requires the resulting ROM to use more dimensions. This was also observed in Chapter 3. While the error for a ROM with $k = 60$ is large and the ROM behaves too stiff for all geometries, the ROM with $k = 60$ already accurately predicts the buckling order for the vertical bead example. The horizontal bead example is the most sensitive one regarding the buckling order. The sensitivity is further underlined by Fig. 4.8a, where the error for a ROM with $k = 140$ still indicates a wrong buckling order. The solution of the ROM and the reference solution is also depicted in Fig. 4.8b. The reference solution is colored turquoise, and the ROM solution is colored grey. At 5 ms, the ROM deformed close to the rigid plate in the top region, while the reference solution mainly deforms in the bottom region of the tube. Even with $k = 140$, the global ROM is not able to accurately reproduce the correct buckling order of the sensitive horizontal buckling example. Nevertheless, a hyper-reduced ROM is created with $k = 60$ and $\tau = 0.05$, resulting in a reduced mesh of 525 elements instead of 1,292 elements. The time-dependent error for the hyper-reduced ROM is shown in Fig. 4.7. Hyper-reduction adds a small amount of error. However, the ROM only fails to predict the buckling order for the horizontal bead, equal to the non-hyper-reduced ROM.

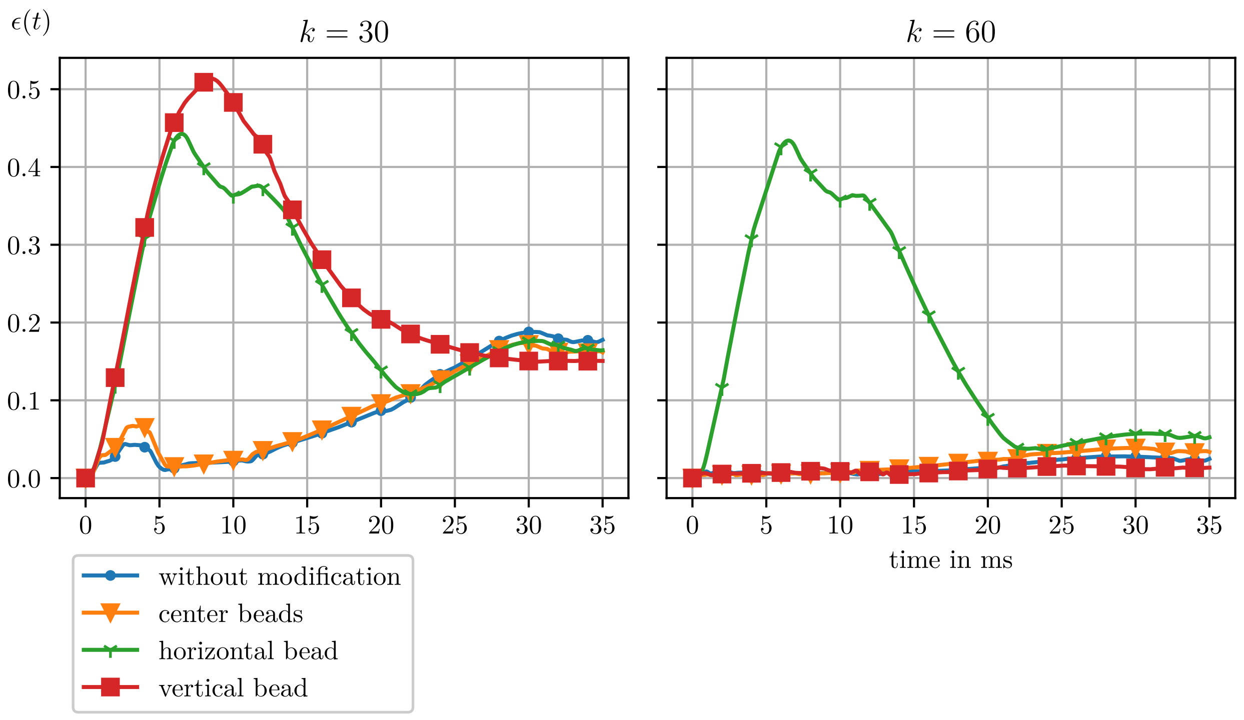


Figure 4.6 Time-dependent error for a non-hyper-reduced global ROM with $k = 30$ and $k = 60$ applied to all geometries.

In conclusion, a more sophisticated method must be utilized to accurately predict the buckling behavior of the sensitive horizontal bead example. Therefore, we apply the IROB approach in the next subsection.

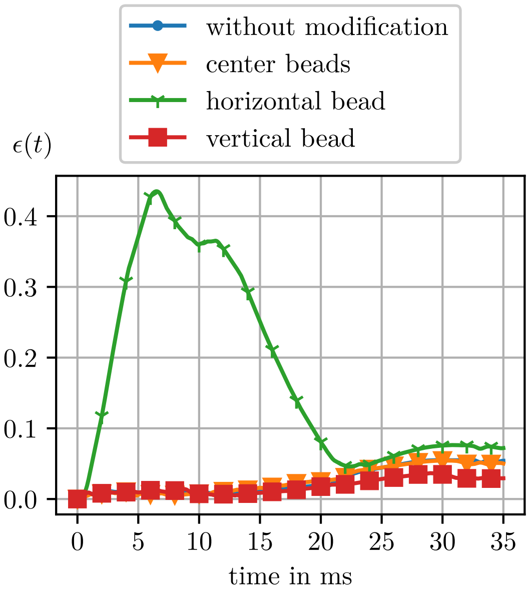
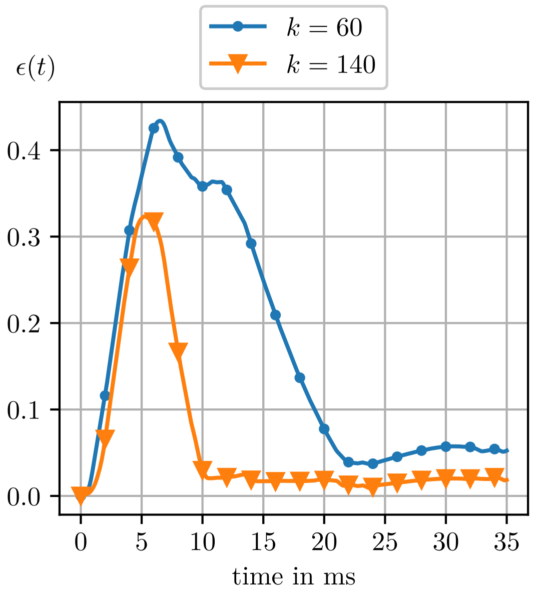
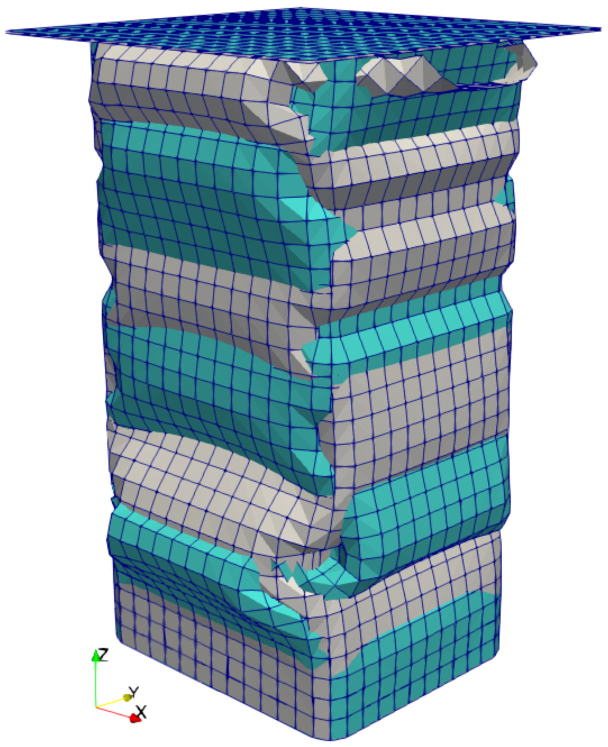


Figure 4.7 Time-dependent error for a hyper-reduced global ROM with $k = 60$ and $\tau = 0.05$ applied to all geometries.



(a) Time-dependent error for a non-hyper-reduced global ROM with $k = 60$ and $k = 140$ applied to the horizontal bead example.



(b) Solution of the ROM with $k = 140$ at 5 ms. The ROM is colored grey and the reference solution is colored turquoise.

Figure 4.8

4.3.6 Local ROB

Similar to the parametric case discussed in Chapter 3, the variance in the data set is too large to be captured by one low-dimensional ROB. In addition, the geometric variations in this chapter lead to sensitive models, which depend even stronger on the approximation quality of the ROB. The sensitivity is illustrated in Fig. 4.8a, where even a ROM with $k = 140$ is not able to reproduce the correct buckling behavior, without mentioning that no efficient hyper-reduction could be carried out with this large dimension. To construct an accurate and low-dimensional ROM even for the sensitive model, we utilize standard k-means-based lROB. The following sections are structured as follows: first, we decide on a suitable number of clusters and evaluate the resulting division of snapshots. Second, we investigate the accuracy of the non-hyper-reduced local ROMs. Finally, global-global hyper-reduction is applied, and the reduced mesh with simulation results is shown as the most accurate local ROM.

Clustering

First, we estimate a suitable number of clusters by evaluating the total variance of the k-means clustering for an increasing number of clusters. The result is shown in Fig. 4.9. As typical for a total variance plot, the curve drops rapidly for small cluster numbers and converges for larger cluster numbers. For further investigation, we select cluster numbers of $n_c = 6, 8, 10$. The assignment of the snapshots to the different

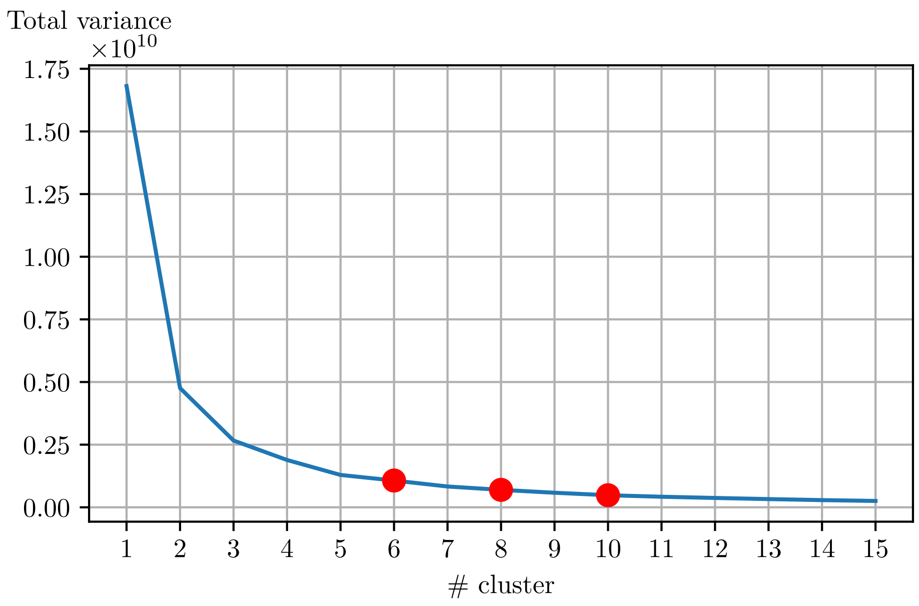
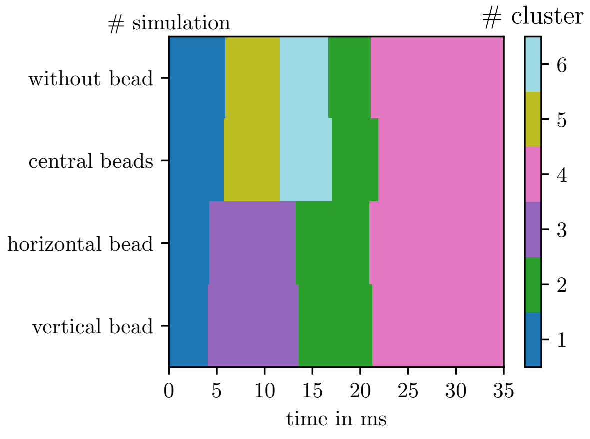
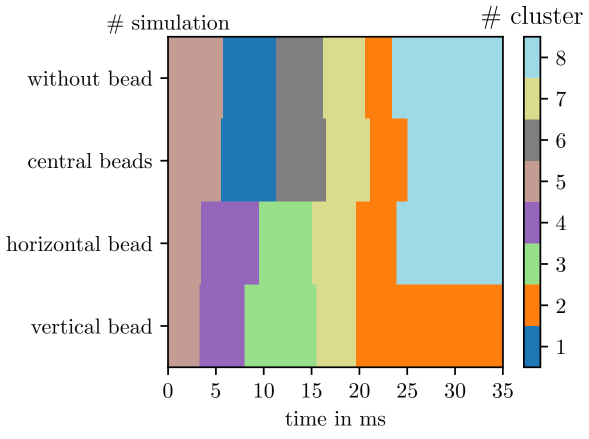


Figure 4.9 Elbow partial MOR

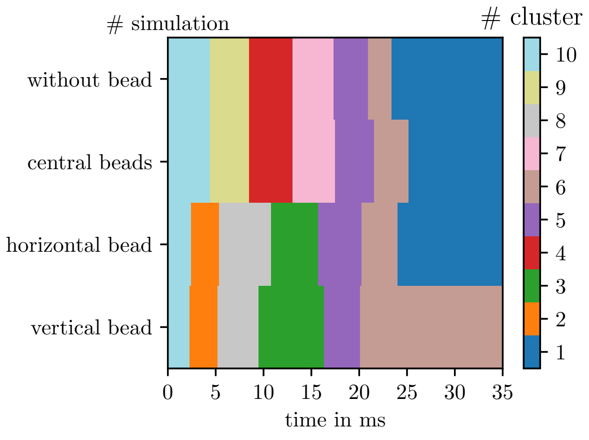
clusters is visualized in Fig. 4.10. The results are shown for $n_c = 6$ in Fig. 4.10a, $n_c = 8$ in Fig. 4.10b, and $n_c = 10$ in Fig. 4.10c. The horizontal and the vertical beads, which are geometry variations with a different buckling order, are assigned to different clusters in the period of the buckling. This can be seen in all subfigures of Fig. 4.10 between 4 ms and 15 ms. The more clusters are used, the thinner the clusters, especially in the early phase of the simulation, where the velocity is high. That is, the magnitude of the state vector $x(t)$ increases fast. Hence, the L^2-based k-means clustering divides the region into more clusters. The final deformation of the vertical bead differs from the remaining geometries. This is observable in Fig. 4.10b and 4.10c, where the final cluster for the vertical bead example differs from the other models. Although the horizontal bead has a different deformation history, the final deformation is similar to the non-modified and center beads example.



(a) $n_c = 6$.



(b) $n_c = 8$.



(c) $n_c = 10$.

Figure 4.10 Assignment of snapshots to clusters, for different cluster numbers. The cluster is indicated by the color. The results are shown for all 35 ms and for all geometry variations.

To ensure a smooth transition of the solution between the clusters, the 300 nearest snapshots are added to each cluster. The number of added snapshots is significantly smaller compared to the previous chapter, where 2,000 snapshots were added. This is because the clusters, in this case, contain fewer snapshots, and the 300 added snapshots are divided among just four different variants. The added snapshots are exemplarily visualized in Fig. 4.10b for the ROM with 8 clusters. Each subplot visualizes one cluster, and in each subplot, the snapshots of the cluster are colored turquoise, and the added ones are colored yellow. Depending on the velocity of the solution, the distance between two consecutive snapshots is larger or smaller. The snapshot distance explains the one-sided addition to the right-hand side for cluster 2 or cluster 7. However, the adjacent clusters add snapshots at the border, which is why a smooth transition is still ensured. Additionally, differences in the velocity of the different models can lead to an unequal addition of snapshots among them. This is seen in cluster 5, which is the initial cluster, where snapshots are mainly added for the non-modified geometry and the central beads model.
Once the snapshots are added, the final clusters are used to compute a ROB for each cluster. As the offline accuracy does not directly correlate with the online accuracy of the ROM, we refrain from examining the offline accuracy and now investigate the time-dependent error of local ROMs with different dimensions in the following subsection.

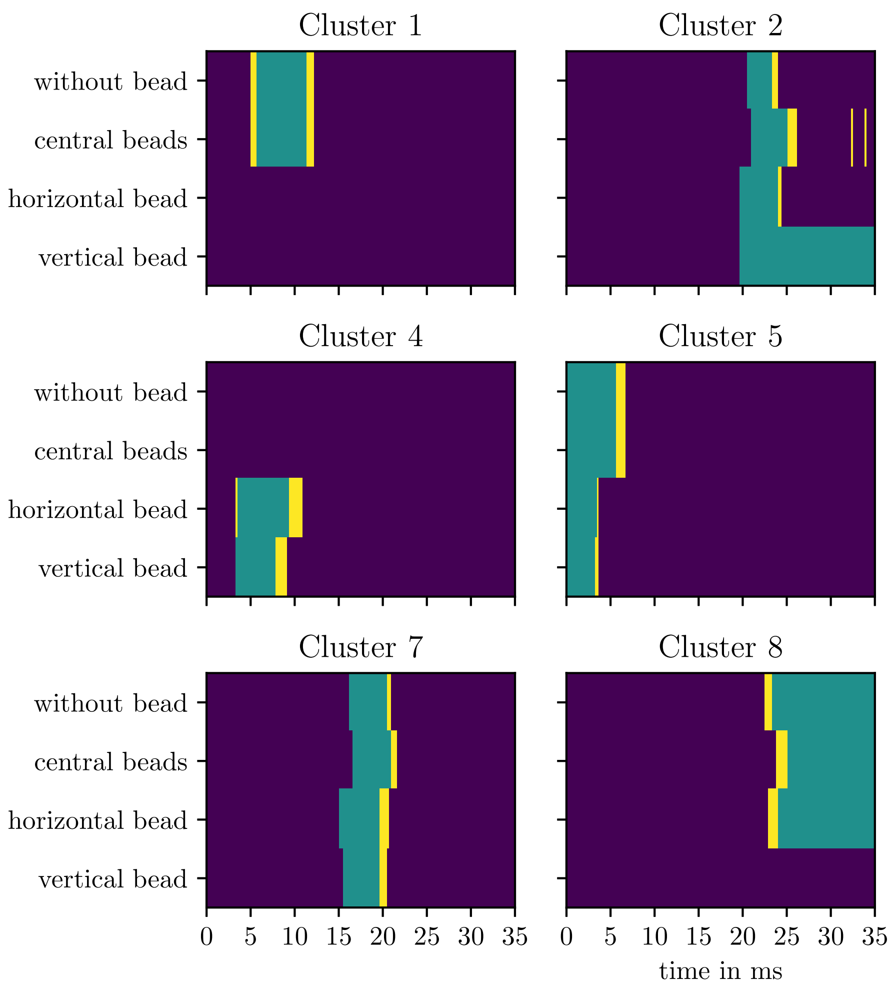


Figure 4.11 Visualization of the added snapshots for the ROM with 8 clusters. The snapshots in the cluster are colored turquoise and the added snapshots are colored yellow.

Online Accuracy

Based on the results of the previous subsection, we construct local k-means-based ROMs with $n_c = 6, 8, 10$ clusters and $k = 20, 60$ dimensions. Fig. 4.12 shows the time-dependent error $\epsilon(t)$ during the simulation time of 35 ms for all four geometry variants. The row in Fig. 4.12 corresponds to the number of clusters n_c and the column to the dimension of the ROM. The local ROM yields accurate results for all geometries, except for the horizontal bead, even for the smallest number of clusters and dimensions. Also, here, the sensitivity of the horizontal bead example is highlighted. Only the ROM with the highest number of clusters and dimensions can accurately predict the horizontal bead example. Also, increasing the dimension of the ROM does not guarantee a more accurate ROM, as can be seen for the ROM with 6 and 8 clusters at the horizontal bead example. Fig. 4.13 shows the chosen cluster of the ROMs with $n_c = 6$ (Fig. 4.13a) and $n_c = 8$ (Fig. 4.13b). For each number of clusters, a ROM with $k = 20$ and $k = 60$ is compared to the reference solution, which is the solution as shown in Fig. 4.10. The high errors of the higher-dimensional ROMs can be related to a wrong choice of the cluster during the simulation time.

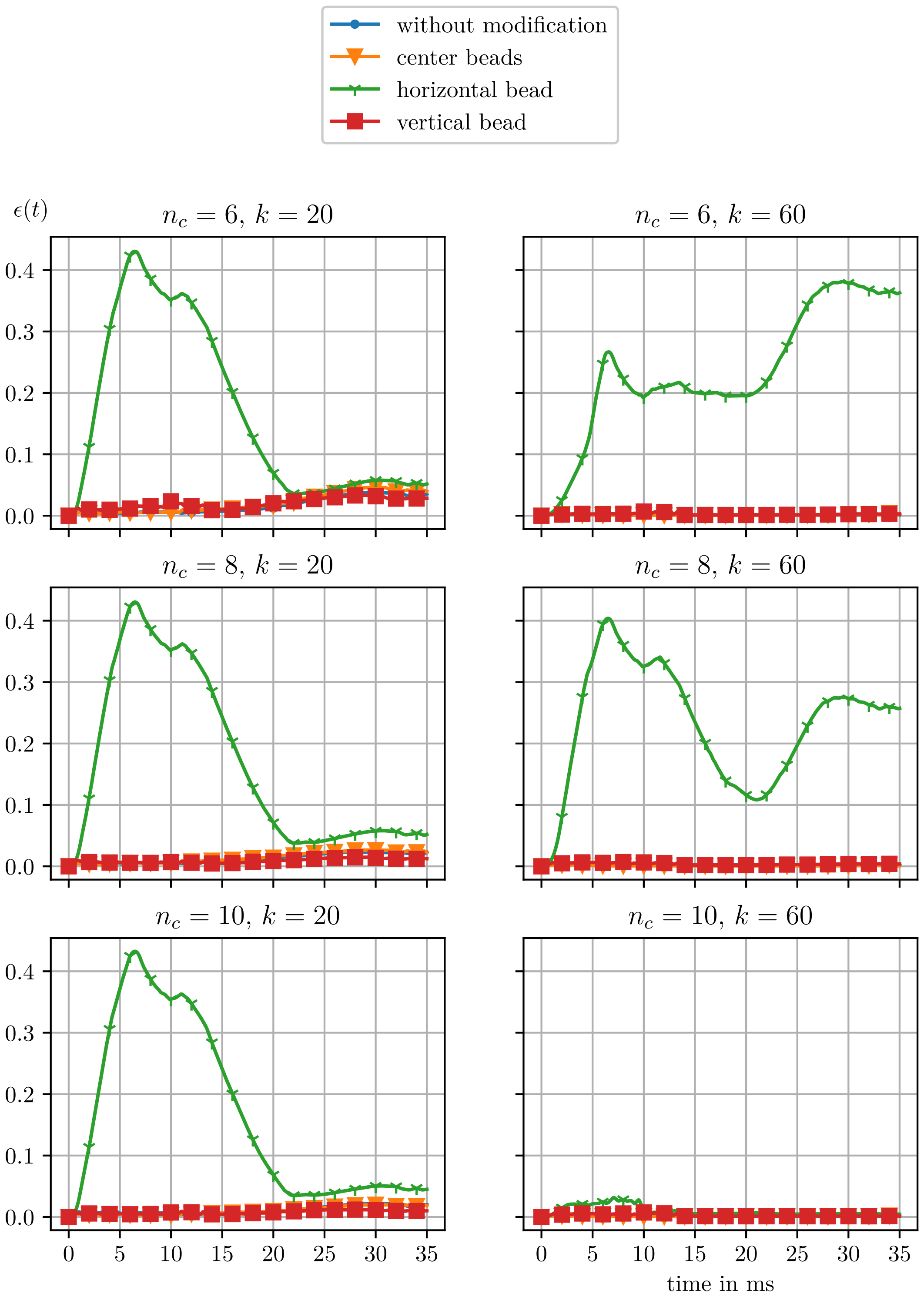
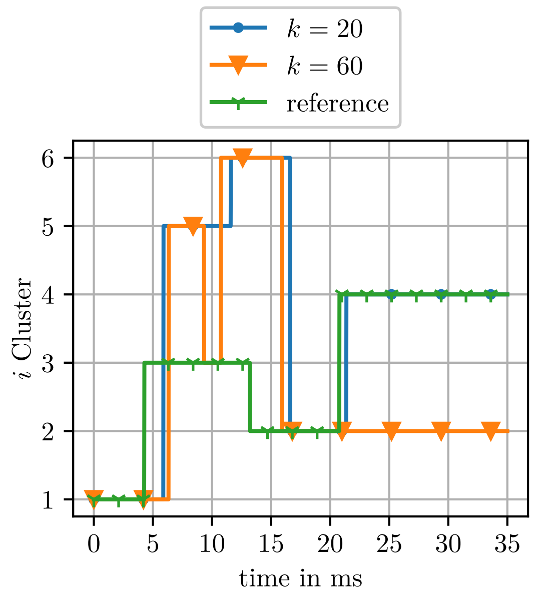
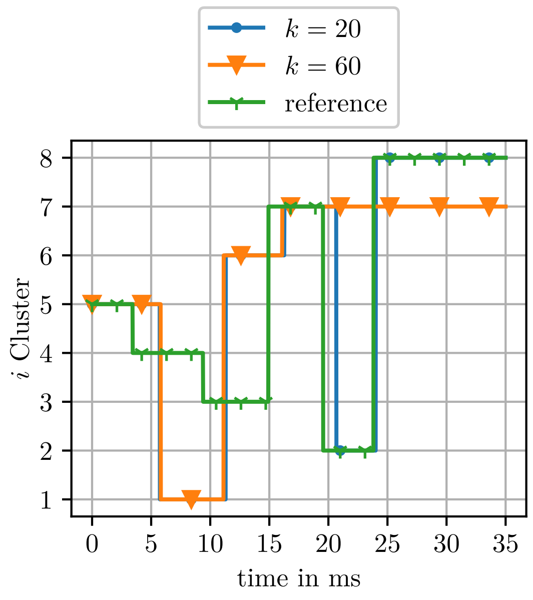


Figure 4.12 Non-hyper-reduced local ROMs for a different number of clusters and dimensions.

Finally, hyper-reduction in a global-global approach is applied to the non-hyper-reduced local ROMs introduced before. The global-global approach was described in Subsection 3.2.4 and yields the reduced mesh sizes, as tabulated in Tab. 4.1. While the number of clusters has a small impact on the reduced mesh size, the dimension k of the ROM and the tolerance τ strongly influence it. Finally, the time-dependent error of the hyper-reduced ROMs as summarized by Tab. 4.1 is assessed in Fig. 4.14. Each row is



(a) Chosen clusters during the simulation for the local ROM with 6 clusters.



(b) Chosen clusters during the simulation for the local ROM with 8 clusters.

Figure 4.13

associated with one cluster number n_c, and the columns are one combination of dimension k and hyper-reduction tolerance τ. The first two columns are the hyper-reduced ROMs with a tolerance of $\tau = 0.02$, and the remaining two are the ROMs with $\tau = 0.005$. The results show that hyper-reduction decreases the accuracy of the ROM. Hyper-reduction is a second layer of approximation, and a loss of precision is to be expected. For the lowest number of clusters $n_c = 6$, hyper-reduction triggers a wrong buckling order for the vertical bead example. An unstable behavior is also observed for $k = 60$ and $\tau = 0.02$. In general, increasing the dimension of the ROM and lowering the approximation tolerance improves the accuracy. However, no convergence proofs are available for local ROMs and parametric problems, which can be seen for the ROM with $n_c = 10$. There, the accuracy is worse compared to the problems with a lower number of clusters. The sensitive horizontal bead example remains challenging, and solely the most accurate ROM with $n_c = 10$, $k = 60$, and $\tau = 0.005$ can predict the solution accurately for all geometry variants. The hyper-reduced ROM results highlight the importance of an accurate ROB as the foundation for hyper-reduction, as only this ROM is also able to accurately predict the solution in the non-hyper-reduced case.
The reduced mesh for the most accurate hyper-reduced local ROM is shown in Fig. 4.15. We assign a weight of 1.0 to the non-reduced parts, which is equal to a non-modified evaluation of the material routine. The reduced mesh is 623 elements large, which is a reduction of 52% related to the reduced elements (1,292) or a reduction of 36% associated with the number of elements in the deformable crash tube. The final deformation at 35 ms of the most accurate local ROM compared to the reference solution, colored in turquoise, is shown in Fig. 4.16. In addition to the non-reduced part, the rigid plate is colored according to a weighting factor of 1.0. In all cases, the ROM behaves stiffer than the FOM. The interpretation of the error as a stiffening has already been discussed. In general, the ROM and the reference agree well, and the error is small.

4.4 Speedup

We estimate the achievable speedup by considering the effort of the projection steps in each time update and the reduced mesh size. A measurement of the computational speedup is not meaningful, as we have

Table 4.1 Summary of hyper-reduction for partial MOR.

# cluster	τ	$k = 20$	$k = 60$
		$\lvert\tilde{\mathsf{E}}\rvert$	$\lvert\tilde{\mathsf{E}}\rvert$
6	0.02	275	403
	0.005	426	580
8	0.02	279	433
	0.005	431	591
10	0.02	290	430
	0.005	439	623

to choose a less efficient implementation due to the limited source code access. The projection of the state vector is:

$$\hat{x}_\partial = \mathbf{\Phi}_\partial^T x_\partial, \tag{4.9}$$

where $\hat{x}_\partial \in \mathbb{R}^{k_\partial}$ is the reduced state vector and $\mathbf{\Phi}_\partial \in \mathbb{R}^{n_\partial \times k_\partial}$ the ROB for the partial reduction. The operation is of asymptotic complexity $\mathcal{O}(k_\partial n_\partial)$. The projection and back-projection are both operations of the same complexity. These additional operations must be performed each time update and slightly increase the computational effort. It is to note that $n_\partial < n$, hence the effort is smaller for the partial ROM than for a full ROM. Also, efficient parallelized algorithms exist to perform linear operations.
The smallest and stiffest elements in the model dominate the critical time step. The crash box consists of elements of similar size. The critical time step is, therefore, similar to that of the whole model, as in the unreduced part. Increasing the time step is not possible and cannot be used to achieve speedup. Increasing the time step would be possible if the dominating elements are in the reduced section and allow for a larger time step after the reduction procedure.
Hyper-reduction effectively reduces the computation time by removing elements from the mesh. The crash box consists of 1,864 shell elements. The non-reduced part has 572 elements. Hence, 1,292 elements are considered for hyper-reduction. For the estimation, we consider the most accurate ROM with a reduced mesh size of 623 and an accurate ROM that fails in the horizontal bead example but has a smaller reduced mesh size of 431. This equals a reduction of 52% elements with respect to the elements considered for hyper-reduction and a reduction of 36% with respect to the total number of shell elements. For the less accurate ROM with a reduced mesh size of 431, the reduction is 67% with respect to the reduced parts and 46% with respect to the full model. According to the solver statistics, shell element processing accounts for approximately 55% of the CPU time in the application. To finally estimate the computational speedup, we multiply the mesh-size-reduction by the share of element processing and obtain an estimated speedup for the most accurate ROM of 20% with respect to the full model. Equally, we obtain a computational speedup of 25% for the second ROM. If we assume a proportional scaling of all processes in the solver, we can estimate the speedup associated with the reduced-only parts of the crash box. In this case, the speedup for the most accurate ROM is 28.6% and 37% for the less accurate ROM.

4.5 Conclusion

In this chapter, we introduced partial MOR. It is motivated by the need to optimize single components embedded in a large model. Since large models require high computational times, we seek to replace the non-modified parts with a ROM. The variation of the component can have a significant impact on the surrounding model, which is why classical methods that enforce a displacement or force fail. Partial MOR replaces the model around the modified component with a projection-based ROM. In this chapter, we begin with a PoC of the proposed ROM formulation. Afterward, we increase the complexity of the ROM method until a suitable ROM is found.
The PoC is a reproductive example of the non-modified crash tube, where one section is left unreduced.

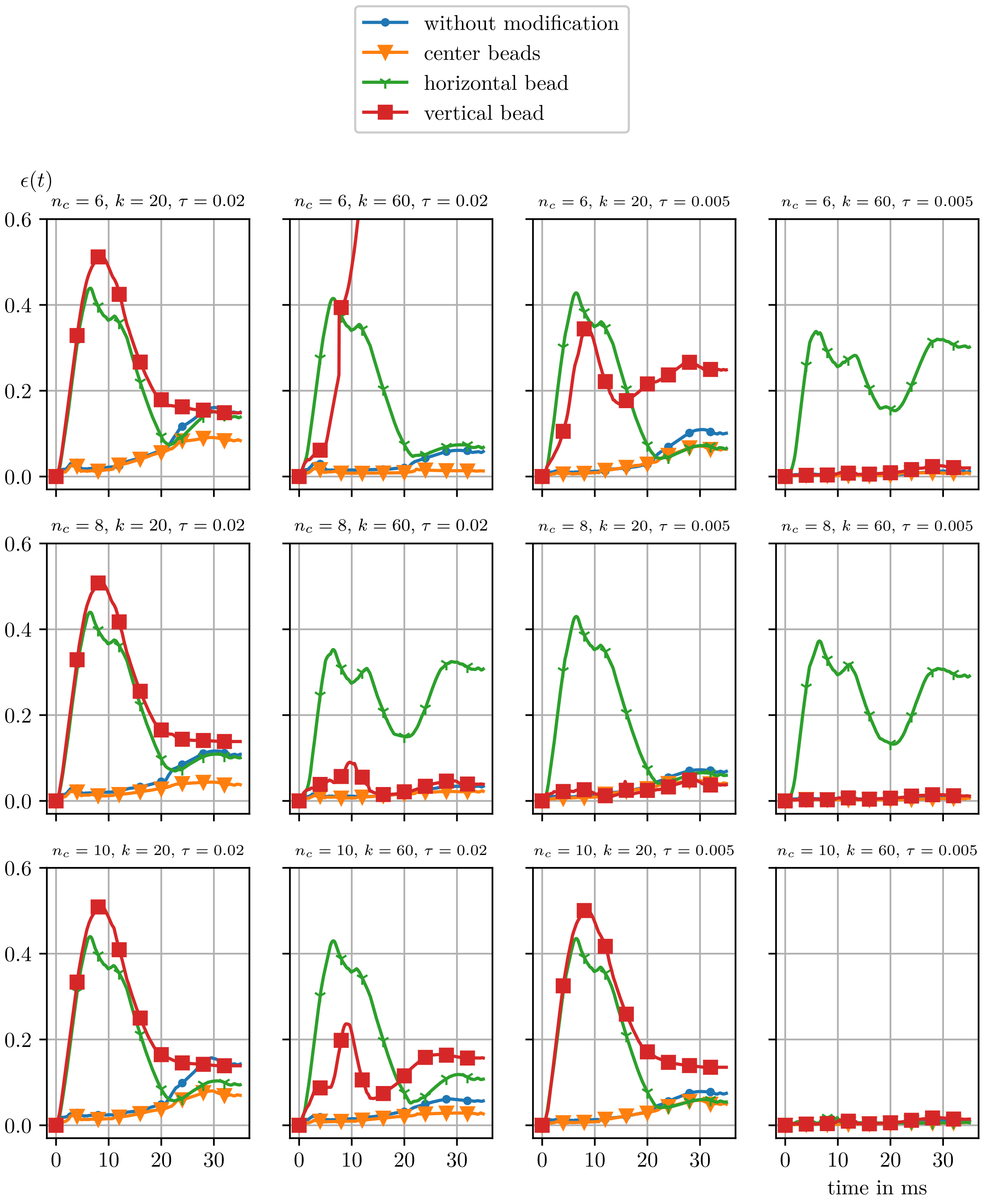


Figure 4.14 Time-dependent error of the hyper-reduced local ROMs applied to all geometry variants.

The proposed method divides the model into reduced and unreduced sections. Next, by modifying the geometry of the unreduced part, the hypothesis is confirmed that small variations can have a large impact on the surrounding structure. The modified geometries partially lead to a different buckling order and shape. The ROM purely trained on the non-modified geometry has not seen the new deformation shapes and fails to accurately predict their behavior. Only the center beads modification is predicted accurately. It

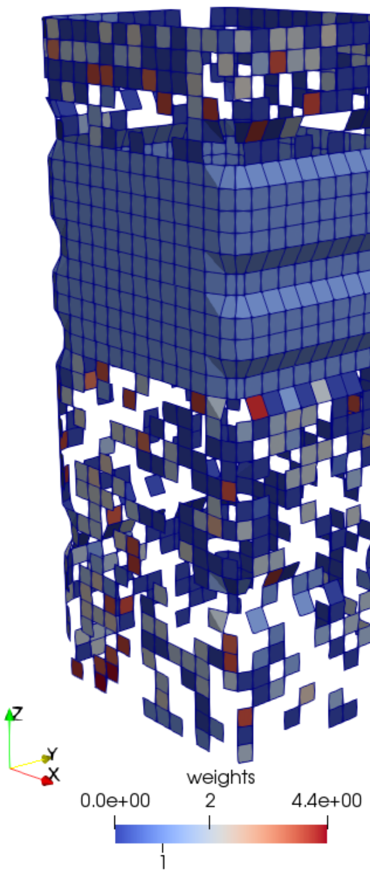


Figure 4.15 Reduced mesh for $n_c = 10$, $k = 60$, and $\tau = 0.005$.

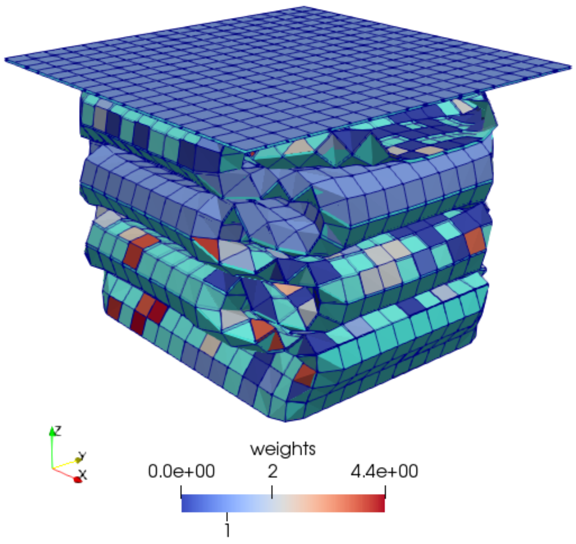
is the modification with the least effect on the surrounding structure. Additional training data of the modified examples is included in the ROM in a global approach to increase the ROM's accuracy for the other geometry variants. However, we observe the same behavior as in Chapter 3. The error slowly decreases for increasing dimensions of the ROM, which leads to high-dimensional ROMs. These ROMs will yield a large reduced mesh, which results in a small computational speedup. The horizontal bead modification poses the most challenging example due to the sensitivity of the buckling order. Even 140 dimensions are not sufficient to predict the correct buckling order in a global approach.

IROB based on k-means clustering is introduced to further increase the complexity of the ROM approach. This approach yields more accurate ROMs while using fewer dimensions than the global approach. However, correctly predicting the buckling order of the horizontal bead example remains challenging, and solely a ROM with a sufficient number of clusters and dimensions is capable of that. In the hyper-reduced case, the hyper-reduction tolerance must be chosen small. We decided on the global-global hyper-reduction approach as it yields smaller reduced mesh sizes. The small reduced mesh size is especially important here to achieve speedup since the critical time step cannot be modified, and a smaller number of elements is available for hyper-reduction.

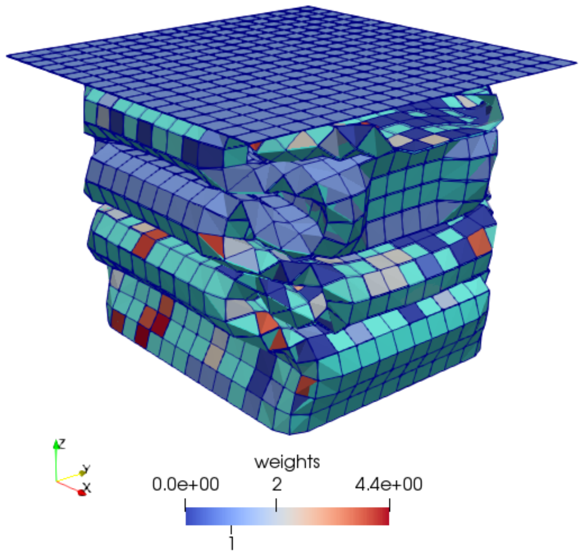
To further improve the ROM's speedup, more accurate ROBs or hyper-reduction must be found. This decreases the dimension of the ROM, which is, therefore, more efficient. One idea to reach this objective is to divide the solution not just along its trajectory, as k-means-based IROB does. Instead, the state vector can also be divided in space. Currently, the solution is reconstructed as follows:

$$x_\partial = \mathbf{\Phi}_\partial \hat{x}_\partial = \sum_{i=1}^{k_\partial} \phi_{\partial,i} \hat{x}_{\partial,i}, \tag{4.10}$$

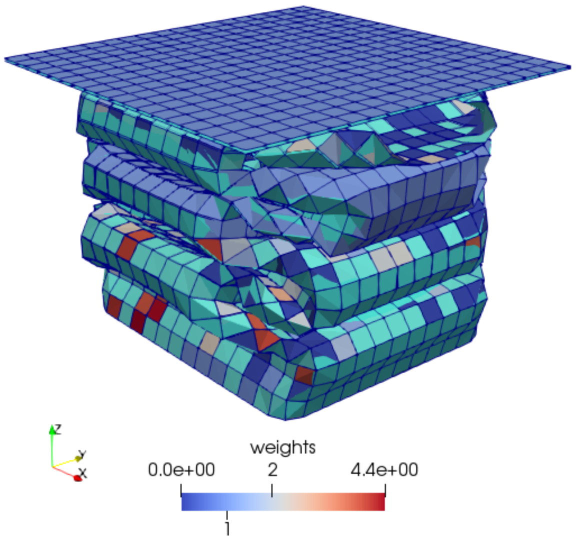
where $\phi_{\partial,i}$ is the i-th column vector of the ROB and $\hat{x}_{\partial,i}$ is the i-th component of the reduced state. The state vector x_∂ contains the DoF of the reduced parts above and below the unreduced part of the crash



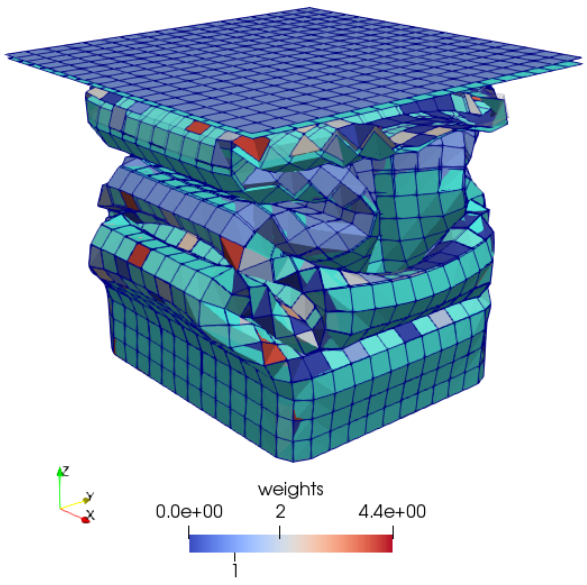
(a) Without modification.



(b) Center beads.



(c) Horizontal bead.



(d) Vertical bead.

Figure 4.16 Final deformation of all geometries at 35 ms.

box. As can be seen in eq. (4.10), one global basis function $\phi_{\partial,i}$, weighted by one scalar coefficient $\hat{x}_{\partial,i}$ describes the deformation of the reduced parts. This means that the reduced part above and below the unreduced part are coupled. One way to decouple them is to treat them separately and make the reduced bases space-local. Each space-local basis is weighted with a separate scalar factor. The state can be reconstructed as follows:

$$x_{\partial} = \sum_{i=1}^{k_{\partial}} = \begin{bmatrix} \phi_{\partial 1,i} & \mathbf{0} \\ \mathbf{0} & \phi_{\partial 2,i} \end{bmatrix} \begin{bmatrix} \hat{x}_{\partial 1,i} \\ \hat{x}_{\partial 2,i} \end{bmatrix}, \tag{4.11}$$

where the subscript $\partial 1$ denotes all DoF associated with the region above the unreduced part and $\partial 2$ with the DoF below.
In conclusion, we successfully replaced the adjacent parts of one modified part in a model with a ROM. This method allows simple boundary conditions to be replaced by complex ROMs. However, sensitive models require complex ROMs, which provide a lower speedup. Further promising research directions exist to design more accurate and faster ROMs.

5 MOR using AE

Chapter 5 is the last technical chapter and investigates pMOR on nonlinear manifolds. So far, all ROMs have been constructed using a global linear subspace or local linear subspaces (lROB). Being aware of linear dimensionality reduction limits and inspired by successful applications in computer graphics [108] and fluid mechanics [79], we decided to investigate pMOR on nonlinear manifolds for crash. In addition, current research strongly focuses on extending linear subspaces for pMOR [110, 112, 114]. During the emergence of this thesis and especially towards the end, newly published research underlines the potential of nonlinear dimensionality reduction for pMOR [164–166]. Nonlinear dimensionality reduction promises smaller reduced dimensions, which positively influence hyper-reduction and promise large speedups despite the additional costs of the nonlinear dimensionality reduction.
This chapter is structured as follows. First, MOR on nonlinear manifolds is introduced in Section 5.1 using a three-dimensional pendulum example to facilitate understanding. Later in this section, the results are transferred to n-dimensional second-order systems to derive the ROM. Once the ROM is presented, hyper-reduction is introduced for nonlinear manifolds in Section 5.2. We use an autoencoder (AE) as a manifold learning method, which is why neural networks and AEs are introduced next. The results section, Section 5.4, discusses and presents results of the AE training and achievable offline approximation errors, the influence of regularization, different AE hyper-parameters, and activation functions. Finally, the most accurate AE is chosen, and hyper-reduction is applied and studied.

5.1 MOR on Nonlinear Manifolds

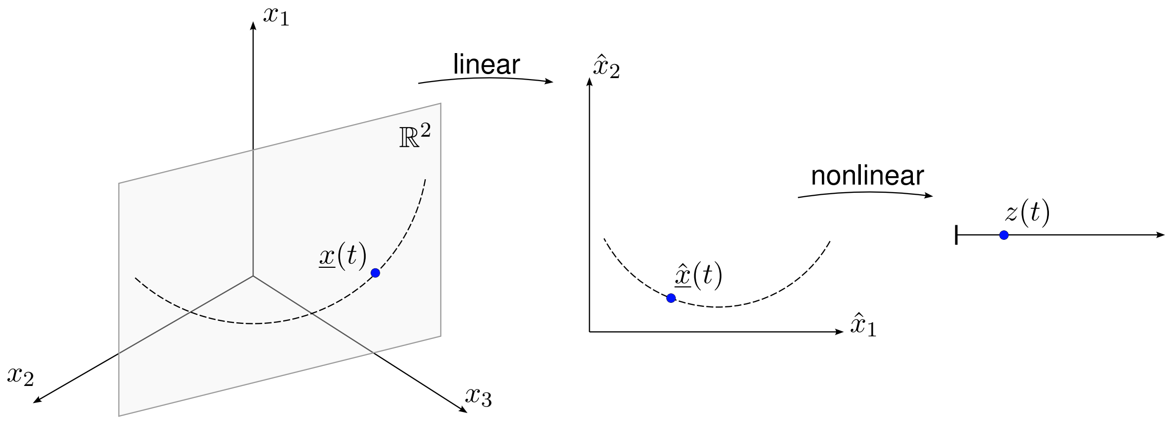


Figure 5.1 One-dimensional manifold embedded in three-dimensional space.

The last chapter of this thesis concerns projection-based MOR on nonlinear manifolds. To illustrate the central concept of nonlinear manifolds, we consider a pendulum example first. With the help of the pendulum example, we introduce all quantities necessary to construct the ROM. Once all required variables are introduced, they are generalized to the high-dimensional case, and the final ROM is presented. Fig. 5.1 shows a one-dimensional manifold embedded in three-dimensional space to emphasize the need for nonlinear dimensional reduction. The left picture in Fig. 5.1 shows the solution $\underline{x}(t)$ of a nonlinear system. This figure assumes the solution of a two-dimensional nonlinear pendulum embedded in three-dimensional space. Using a linear coordinate transformation, the system is transformed into two-dimensional space. We utilize the same notation as in the previous chapters to emphasize the connection. The two-dimensional state $\underline{\hat{x}}(t)$ is obtained as orthogonal projection onto the two-dimensional plane:

$$\underline{\hat{x}}(t) = \mathbf{\Phi}^T \underline{x}(t). \tag{5.1}$$

The plane is spanned by the orthonormal column vectors of $\mathbf{\Phi} \in \mathbb{R}^{3\times 2}$. In the previous chapters, the column vectors were the deformation modes. In addition, it is to be noted that the mapping in eq. (5.1) is exact. The error-free example is for illustrative purposes. However, real examples with approximation error assume a small error. That is, the out-of-plane component during the projection step is small.
The solution is still a line, a one-dimensional manifold that can be described by one parameter. This parameter is the angle in the pendulum model. To formulate the system and the solution in their intrinsic dimension $z \in \mathbb{R}$, a nonlinear transformation $\Gamma:\ \mathbb{R} \to \mathbb{R}^2$ is required. Using Γ, we can express $\underline{\hat{x}}$ in terms of z:

$$\underline{\hat{x}}(t) = \Gamma(z(t)). \tag{5.2}$$

Since the solution of the pendulum example is known, we can further specify Γ. First, we know that the solution manifold $\mathcal{M}$ is the set of points on the circle with a radius equal to the pendulum's length. For simplicity, we assume that the radius is 1. Hence, we define the solution manifold as:

$$\mathcal{M} = \{\hat{x}_1, \hat{x}_2 \in \mathbb{R}\,|\,\hat{x}_1^2 + \hat{x}_2^2 = 1\}. \tag{5.3}$$

The positive quarter $\hat{x}_1, \hat{x}_2 \geq 0$ of $\mathcal{M}$ is shown in Fig. 5.2. In this example, we identify the mapping Γ for the circle as:

$$\underline{\hat{x}} = \Gamma(z(t)) = \begin{bmatrix} \cos(z(t)) \\ \sin(z(t)) \end{bmatrix}. \tag{5.4}$$

The time derivative of $\underline{\hat{x}}$ is obtained by applying the chain rule:

$$\dot{\underline{\hat{x}}} = \underbrace{\frac{\partial \Gamma}{\partial z}}_{\mathbf{J}} \underbrace{\frac{\mathrm{d}z}{\mathrm{d}t}}_{\dot{z}}, \tag{5.5}$$

where $\mathbf{J}$ is the Jacobian matrix of the mapping Γ and $\dot{z}$ is the latent velocity. We insert eq. (5.4) in eq. (5.5) and obtain:

$$\dot{\underline{\hat{x}}} = \begin{bmatrix} -\sin(z(t)) \\ \cos(z(t)) \end{bmatrix} \dot{z} \tag{5.6}$$

for the circle. The column vectors of the Jacobian span the tangent space to the solution manifold. In this simple example, the tangent space is a line spanned by one column of $\mathbf{J}$.

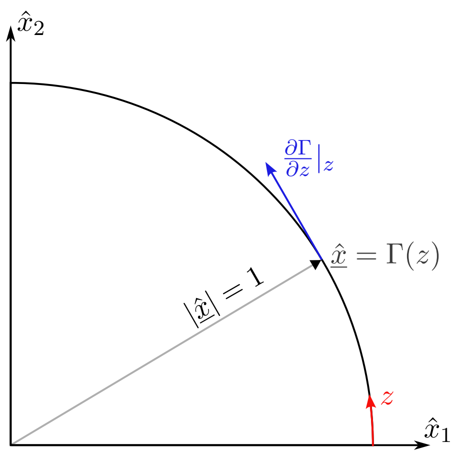

$\hat{x}_2$
$\frac{\partial \Gamma}{\partial z}|_z$
$\underline{\hat{x}} = \Gamma(z)$
$|\underline{\hat{x}}| = 1$
z
$\hat{x}_1$

Figure 5.2 AE based pROM using a linear outer layer and an AE to resolve the nonlinear correlations.

Before we derive the equations for projection-based MOR on nonlinear manifolds in the k-dimensional case, we generalize the concepts introduced before and visualized in Fig. 5.2 with the help of a pendulum example. Fig. 5.3 depicts an r-dimensional manifold $\mathcal{M}$ embedded in the k-dimensional space. The solution $\underline{\hat{x}}(t)$ evolves over time. At each point in time, the columns of the Jacobian matrix $\frac{\partial \underline{\hat{x}}}{\partial z_i} \in \mathbb{R}^k$ for $i = 1, .., r$ span the tangent space $\mathbf{T}_{\underline{\hat{x}}}\mathcal{M}$. Next, we derive projection-based MOR for the r-dimensional

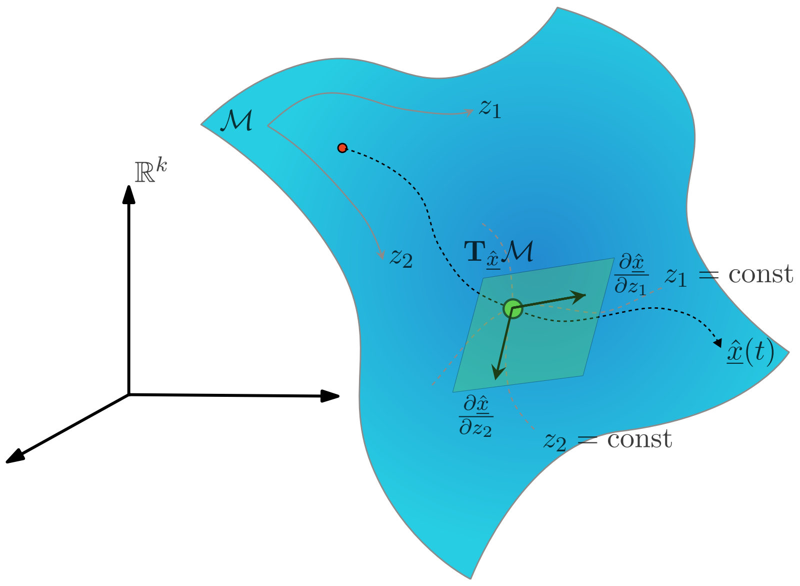

$\mathcal{M}$
z_1
$\mathbb{R}^k$
z_2
$\mathbf{T}_{\underline{\hat{x}}}\mathcal{M}$
$\frac{\partial \underline{\hat{x}}}{\partial z_1}$
$z_1 = \text{const}$
$\underline{\hat{x}}(t)$
$\frac{\partial \underline{\hat{x}}}{\partial z_2}$
$z_2 = \text{const}$

Figure 5.3 AE approximating the solution manifold $\mathcal{M}$.

case based on nonlinear mappings. The nonlinear mapping is defined as follows:

$$\hat{x}(t) \approx \tilde{\hat{x}}(t) = \Gamma(z(t)), \tag{5.7}$$

where $\hat{x}(t) \in \mathbb{R}^k$ is the linear reduced state with dimension k, $z(t) \in \mathbb{R}^r$ is the nonlinear reduced state with dimension $r < k$, Γ, $\mathbb{R}^r \to \mathbb{R}^k$ is the nonlinear mapping that connects the two states, and $\tilde{\hat{x}}(t) \in \mathbb{R}^k$ is the approximation of the state. Since Γ is not exact, the equations hold approximately. However, we forego the approximate sign in the following and write an equal sign for clarity.
Next, we recall the global ROM eq. (2.25b):

$$\underbrace{\mathbf{\Phi}^T \mathbf{M}(\mu) \mathbf{\Phi}}_{\tilde{\mathbf{M}}} \ddot{\hat{x}} + \mathbf{\Phi}^T f(t, \mathbf{\Phi}\hat{x}, \mathbf{\Phi}\dot{\hat{x}}, \mu) = 0.$$

To insert eq. (5.7) into eq. (2.25b), we differentiate the nonlinear mapping twice with respect to time, which yields:

$$\dot{\hat{x}} = \frac{\partial \Gamma}{\partial z} \dot{z}, \tag{5.8}$$

$$\ddot{\hat{x}} = \frac{\partial \Gamma}{\partial z} \ddot{z} + \frac{\partial^2 \Gamma}{\partial z^2} \dot{z}\dot{z}, \tag{5.9}$$

We identify the Jacobian matrix as:

$$\mathbf{J} = \frac{\partial \Gamma}{\partial z} \in \mathbb{R}^{k \times r}, \tag{5.10}$$

and the Hessian as:

$$\mathbf{H} = \frac{\partial^2 \Gamma}{\partial z^2} \in \mathbb{R}^{k \times r \times r}. \tag{5.11}$$

Next, we insert the second time derivative and the nonlinear mapping into the linearly reduced ROM. The ROM, including the nonlinear mapping, reads:

$$\tilde{\mathbf{M}}(\mathbf{J}\ddot{z} + \mathbf{H}\dot{z}\dot{z}) + \mathbf{\Phi}^T f(t, \mathbf{\Phi}\Gamma(z), \mathbf{\Phi}\mathbf{J}\dot{z}, \mu) = r, \tag{5.12}$$

where the residual $r \in \mathbb{R}^k$ is formed due to the approximation. To obtain the final ROM, we must still determine which condition to apply to the residual. We look closer at the variational formulation of the governing equations derived in Section 2.1.2 to find this condition. Although we named the test function δv_i virtual velocity, and the derivation consequently principle of virtual power, the choice of the test function is arbitrary. The test function is required to test the whole function space, which is the space of linear shape functions in this thesis. The function space in the principle of virtual power is defined in eq. (2.7). Considering the test function as virtual displacement δd yields the equivalent result if the function space is maintained. Using virtual displacements as a test function is also known as the principle of virtual work in literature. The test function represents all kinematically admissible deformations. The test function reads:

$$\delta \mathbf{d}(\mathbf{X}) = N_I(\mathbf{X})\delta \mathbf{d}_I, \quad \text{for } I \in [1, n_N]. \tag{5.13}$$

Compared to the virtual velocities eq. (2.12), only the variable naming has changed. However, since we perform SVD and the AE training based on displacements, continuing the derivation with virtual displacements is more intuitive. The governing equation including the arbitrary virtual displacements reads:

$$\delta d^T(\mathbf{M}\ddot{x} + f) = 0, \tag{5.14}$$

where all n arbitrary values $\delta \mathbf{d}_I$ are flattened in $\delta d \in \mathbb{R}^n$. The dimension n is defined in eq. (2.15). In the following, we restrict the test function to different subspaces. First, we restrict the solution to lie in the k-dimensional subspace spanned by the columns of the ROB $\mathbf{\Phi}$. Using the linear mapping eq. (2.23), the virtual displacement can be further written as:

$$\delta d = \delta(x(\hat{x}) - x_0) = \frac{\partial(x - x_0)}{\partial \hat{x}}\delta\hat{x} = \mathbf{\Phi}\delta\hat{x}. \tag{5.15}$$

Inserting the linear mapping eq. (2.23) and the variation eq. (5.15) in eq. (5.14) yields the standard linear ROM, which we used throughout this thesis:

$$\delta\hat{x}^T[\mathbf{\Phi}^T\mathbf{M}(\mu)\mathbf{\Phi}\ddot{\hat{x}} + \mathbf{\Phi}^T f(t, \mathbf{\Phi}\hat{x}, \mathbf{\Phi}\dot{\hat{x}}, \mu)] = 0. \tag{5.16}$$

Since the variations $\delta\hat{x} \in \mathbb{R}^k$ are arbitrary, the expression inside the square brackets must vanish, and we obtain the linear ROM (eq. (2.25b)).
Similarly, we derive the AE ROM. We express the variation $\delta\hat{x}$ in terms of the AE's latent coordinates variation $\delta z \in \mathbb{R}^r$. For an infinitesimal displacement, we can express $\delta\hat{x}$ as:

$$\delta\hat{x} = \frac{\partial\Gamma}{\partial z}\delta z = \mathbf{J}\delta z. \tag{5.17}$$

We identify the Jacobian matrix again, which governs the space of kinematically admissible deformations. Inserting the second time derivative eq. (5.9) in eq. (5.16) and expanding the variation according to eq. (5.17) yields the final AE ROM:

$$\mathbf{J}^T\tilde{\mathbf{M}}\mathbf{J}\ddot{z} + \mathbf{J}^T\tilde{\mathbf{M}}\mathbf{H}\dot{z}\dot{z} + \mathbf{J}^T\mathbf{\Phi}^T f(t, \mathbf{\Phi}\Gamma(z), \mathbf{\Phi}\mathbf{J}\dot{z}, \mu) = 0. \tag{5.18}$$

Comparing the ROM derived using the principle of virtual work with eq. (5.12) reveals, that the principle of virtual work is equal to enforcing the residual to be orthogonal to the tangent space spanned by the columns of $\mathbf{J}$:

$$\mathbf{J}^T r \overset{!}{=} 0. \tag{5.19}$$

Finally, we evolve the ODE system in time using the explicit central difference method, as described by Algorithm 6. In the initial step of the time stepping procedure for the AE ROM, the initial conditions must be transformed once to the latent space. The transformation of the initial conditions is the only step in which the Encoder ϕ is used. To still obtain an explicit scheme, the velocity of the last half time step is used to evaluate the additional term that includes the Hessian matrix. This term is due to the curvature of the solution manifold. Once the initial conditions are transformed, the Jacobian and Hessian matrix can be

Algorithm 6 Explicit time integration for the AE ROM.

Input: Initial conditions $x_0, \dot{x}_{\frac{1}{2}}$, boundary conditions, termination time t_{end}, linear mapping $\mathbf{\Phi}$, nonlinear mapping $\Gamma(z)$, and additional parameters μ

Output: Temporal evolution of model $x_{t_n}, \dot{x}_{t_{n+\frac{1}{2}}}, z_{t_n}$, and $\dot{z}_{t_{n+\frac{1}{2}}}$

```
 1: t = 0
 2: $\tilde{\mathbf{M}} = \mathbf{\Phi}^T\mathbf{M}\mathbf{\Phi} \leftarrow$ compute linearly reduced mass matrix
 3: $z_0 = \phi(\mathbf{\Phi}^T x_0)$, $\dot{z}_0 = \dot{z}_{-\frac{1}{2}} = \frac{\partial \phi}{\partial \tilde{x}}|_{x_0}\mathbf{\Phi}^T\dot{x}_0 \leftarrow$ compute initial conditions
 4: $\tilde{x}_0 = \Gamma(z_0)$, $\mathbf{J}_0 = \frac{\partial \Gamma}{\partial z}|_{z_0}$, $\dot{\tilde{x}}_{-\frac{1}{2}} = \mathbf{\Phi}\mathbf{J}_0\dot{z}_{-\frac{1}{2}}$, $\mathbf{H}_0 = \frac{\partial^2\Gamma}{\partial z^2}|_{z_0} \leftarrow$ Forward pass of Decoder
 5: $\mathbf{J}_0^T\tilde{\mathbf{M}}\mathbf{J}_0 \leftarrow$ Compute nonlinearly reduced mass matrix
 6: while $t \leq t_{\text{end}}$ do
 7:     $f_n^{\text{AE}} = \mathbf{J}_n^T\tilde{\mathbf{M}}\mathbf{H}_n\dot{z}_{n-\frac{1}{2}}\dot{z}_{n-\frac{1}{2}} + \mathbf{J}_n^T\mathbf{\Phi}^T f(t_n, \tilde{x}_n, \dot{\tilde{x}}_{n-\frac{1}{2}}, \mu) \leftarrow$ calculate force
 8:     $\ddot{z}_n = (\mathbf{J}_n^T\tilde{\mathbf{M}}\mathbf{J}_n)^{-1} f_n^{\text{AE}} \leftarrow$ Compute new reduced acceleration
 9:     $\dot{z}_{n+\frac{1}{2}} = \dot{z}_{n-\frac{1}{2}} + \Delta t_x \ddot{z}_n$
10:     $z_{n+1} = z_n + \Delta t_2 \dot{z}_{n+\frac{1}{2}}$
11:     $\tilde{x}_{n+1} = \Gamma(z_{n+1})$, $\mathbf{J}_{n+1} = \frac{\partial \Gamma}{\partial z}|_{z_{n+1}}$, $\dot{\tilde{x}}_{n+\frac{1}{2}} = \mathbf{\Phi}\mathbf{J}_{n+1}\dot{z}_{n+\frac{1}{2}}$, $\mathbf{H}_{n+1} = \frac{\partial^2\Gamma}{\partial z^2}|_{z_{n+1}} \leftarrow$ Forward pass of
        Decoder to obtain new high-dimensional state
12:     $t = t + \Delta t_2$
13:     $n = n + 1$
14:     $\mathbf{J}_{n+1}^T\tilde{\mathbf{M}}\mathbf{J}_{n+1} \leftarrow$ Compute next nonlinearly reduced mass matrix
15: end while
```

computed for the first time step. Next, the nonlinear force term is obtained, and the latent acceleration can be determined using the reduced mass matrix. The acceleration is then integrated in time, and the velocity and state are obtained. In the forward pass to obtain the high-dimensional state and velocity, the Jacobian and Hessian matrices are also obtained at the next time step. Finally, the time and index are incremented, and the next iteration can be entered.

As the ROM derivation for a nonlinear dimensionality reduction model is completed, the remaining part is to determine the model Γ and ϕ. As the name of the ROM and the chapter suggests, this thesis concerns AEs as nonlinear dimensinality reduction method. An AE is a Neural Network (NN) type consisting of an Encoder and a Decoder. Before explaining in the next sections NNs in detail and introducing different types of AEs, Fig. 5.4 shows the basic structure of the employed model. The AE is enclosed by two linear layers, which a global SVD determines. These linear layers are intended to resolve the linear correlations, as illustrated in Fig. 5.1. The AE resolves the final nonlinear correlations. The pre-reduction enables the use of an efficient NN, which is significantly reduced in size, requires less training data, and calculates faster. In the next section, we introduce the concept of NNs.

5.2 Hyper-reduction on Manifolds

Also, in the AE ROM, hyper-reduction is the only way to achieve speedup. Hyper-reduction is even more important for the AE ROM because many operations still scale with the higher linear ROM dimension. As shown in Chapter 3 and 4, hyper-reduction strongly scales with the dimension of the ROM. The aim is, therefore, to use the small latent dimension of the AE to create a small reduced mesh. The small reduced mesh size should accelerate the ROM more than the additional computational effort of the Jacobian and Hessian matrix decelerates it.

Without further modification, ECSW is directly compatible with the AE ROM. The nonlinear internal force term reads:

$$\mathbf{J}(z)^T\mathbf{\Phi}^T f_{int} = \sum_{e=1}^{n_e}\mathbf{J}(z)^T\mathbf{\Phi}^T\mathbf{L}_e f_{int}^{(e)} = \sum_{e=1}^{n_e}\mathbf{J}(z)^T\mathbf{\Phi}_e^T f_{int}^{(e)}. \tag{5.20}$$

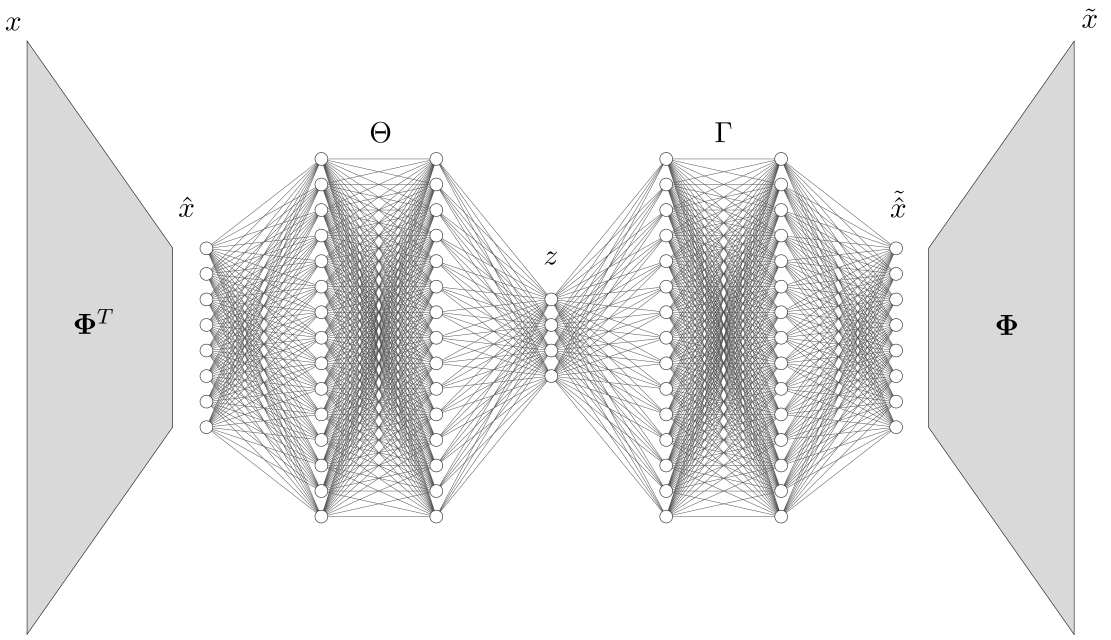


Figure 5.4 AE-based pROM using a linear outer layer and an AE to resolve the nonlinear correlations.

Equally to Section 3.2.4, the assembly of all n_e elements is replaced by a weighted reduced sum of all elements in the reduced mesh $\tilde{\mathsf{E}}$. The approximation reads:

$$\mathbf{J}^T\boldsymbol{\Phi}^T f_{int} \approx \sum_{e\in\tilde{\mathsf{E}}} \xi_e \mathbf{J}(z)^T \boldsymbol{\Phi}_e^T f_{int}^{(e)}, \quad \xi_e > 0. \tag{5.21}$$

To determine the weights ξ_e and the reduced mesh $\tilde{\mathsf{E}}$, optimization problem eq. (3.61) is solved. For simplicity, we restate the optimization problem again:

$$\xi^* = \arg\min_{\xi\in\Lambda} \|\xi\|_0$$
$$\Lambda = \{\xi \in \mathbb{R}^{n_e} : \|G_{\mathrm{AE}}\xi - b_{\mathrm{AE}}\|_2 \leq \tau\|b\|_2, \xi_e \geq 0\},$$

where $\tau \in (0, 1]$ is a predefined accuracy. The reduced set $\tilde{\mathsf{E}}$ comprises the elements with non-zero weights:

$$\tilde{\mathsf{E}} = \{e \in \{1, 2, \ldots, n_e\} : \xi_e \neq 0\}.$$

The global matrix $G_{\mathrm{AE}} \in \mathbb{R}^{n_{b,\mathrm{AE}}\times n_e}$ is composed of submatrices $G_{\mathrm{AE},j,e} \in \mathbb{R}^{n_{b,\mathrm{AE}}}$. The dimension $n_{b,\mathrm{AE}}$ is the sum of the bottleneck dimension r overall n_f force snapshots:

$$n_{b,\mathrm{AE}} = r * n_f \ll n_b = k * n_f. \tag{5.22}$$

Since the bottleneck dimension r is smaller than the dimension of the linear ROM k and both are multiplied by the usually large number of training snapshots n_f, the dimension is smaller than the linear dimension n_b, and the optimization problem becomes drastically smaller. The submatrix $G_{\mathrm{AE},j,e}$ is the product of the Jacobian matrix $\mathbf{J}(z_j)^T$ at the current solution point z_j, and the reduced basis Φ with the force snapshot at t_j of element e:

$$G_{\mathrm{AE},j,e} = \mathbf{J}(z_j)^T \boldsymbol{\Phi}_e^T f_j^{(e)} \in \mathbb{R}^r. \tag{5.23}$$

The vector b_{AE} is the column sum of matrix G_{AE}:

$$b_{\mathrm{AE}} = \sum_{e=1}^{n_e} G_{\mathrm{AE},j,e} \in \mathbb{R}^{n_{b,\mathrm{AE}}}. \tag{5.24}$$

Regularization can be included in the optimization problem to avoid overfitting when working with NNs. The NNLS problem in each iteration of Algo. 4 can be extended by including regularization:

$$\xi^* = \arg\min_{\xi \geq 0} \left(\|G\xi - b\|_2^2 + \|\beta\xi\|_2^2 \right). \tag{5.25}$$

We refer to Bach [1] for further details about regularization in ECSW.

5.3 Neural Networks

In this section, we introduce NNs with all relevant topics, such as data preparation, NN training, and various AE architectures. We choose AEs as a nonlinear dimensionality reduction method due to the easily accessible libraries, including automatic differentiation (AD) frameworks, which efficiently compute gradients of the NNs. Also, NNs have proven to be powerful regression models.

5.3.1 Artificial Neuron

The base element of artificial NNs is the artificial neuron, as depicted in Fig. 5.5. It takes the inputs x_i for $i = 1, \dots, n$, weights them with weighting factors w_i, and sums them up. A bias b is added before the sum is passed to the activation function $f(z)$ $\mathbb{R} \to \mathbb{R}$. Hence, the output $y \in \mathbb{R}$ of a single artificial neuron is obtained as:

$$y = f(\sum_{i=1}^{n} x_i w_i + b) = f(\underline{w}^T \underline{x} + b). \tag{5.26}$$

The role of the activation function f is crucial. If linear activation functions are used, the NN can perform

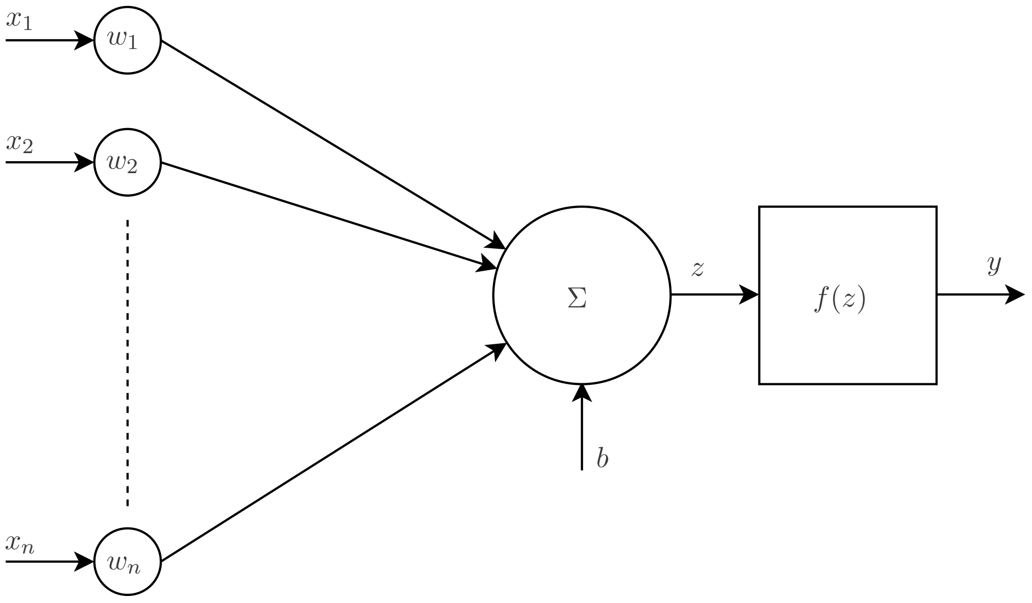


Figure 5.5 Schematic representation of an artificial neuron.

purely linear operations as the composition of linear functions is also linear. Therefore, we need to consider nonlinear activation functions to obtain a nonlinear dimensionality reduction model. Some popular activation functions are depicted in Fig. 5.6, and the analytical formulas are given in Tab. 5.1. The linear activation function maps the input to the output. The rectified linear unit (ReLU) maps all negative values to zero and is linear for positive inputs. However, the first derivative has a jump at zero, and the second derivative has a Dirac impulse at zero. The exponential linear unit (ELU) is similar to the ReLU function except that the transition from negative to positive values is continuously differentiable if $\alpha = 1.0$. Also, the negative branch is mapped to negative values. Another variant of ReLU is the leaky ReLU function, which allows small positive gradients for negative inputs. The non-vanishing gradient mitigates the vanishing gradient problem, which can occur during training. Also, the negative branch is not mapped to zero, which preserves the invertability of the function. It is also possible to use custom activation functions, such as a quadratic function. So far, all activation functions are unbounded for large values. In contrast, the sigmoid function is bounded and saturates for large input values. However, the saturation can cause vanishing gradients.

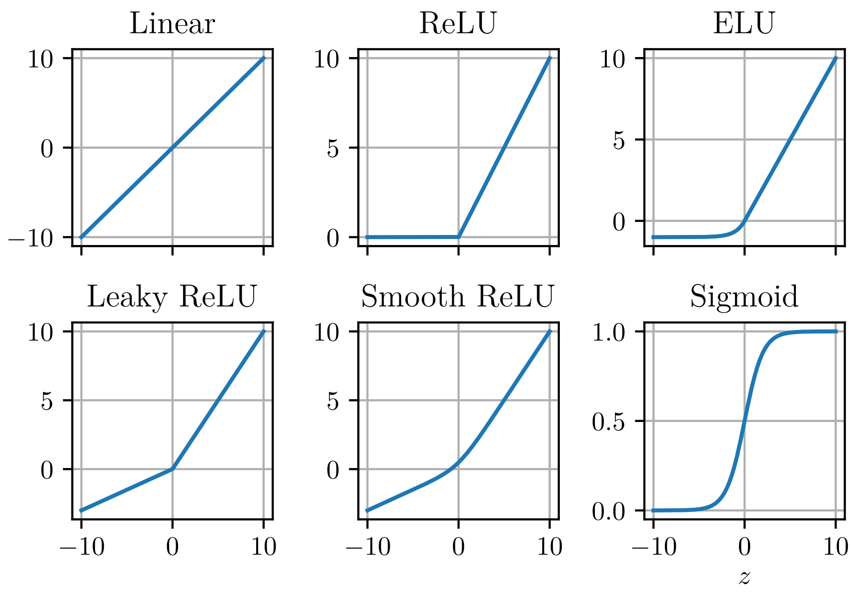


Figure 5.6 Different activation functions.

Table 5.1 Summary of activation functions visualized in Fig. 5.6.

Name	Formula
Linear	$f(z) = z$
ReLU	$f(z) = \max(0, z)$
ELU, $\alpha = 1.0$	$f(z) = \begin{cases} z, & z > 0 \\ \alpha(e^z - 1), & z \leq 0 \end{cases}$
Leaky ReLU, $\alpha = 0.3$	$f(z) = \begin{cases} z, & z > 0 \\ \alpha z, & z \leq 0 \end{cases}$
Smooth ReLU, $\alpha = 0.3$	$f(z) = \alpha z + (1 - \alpha) \log(1 + \exp z)$
Sigmoid	$f(z) = \frac{1}{1+e^{-z}}$

The artificial neurons are combined into layers to obtain a NN, which are again stacked together. Once the architecture of the NN is determined, the trainable parameters of the NN are updated in the training phase. The trainable parameters are usually the weights and the bias. Parameters concerning the architecture, such as the dimension and the number of layers, are pre-defined and termed hyper-parameters. Next, we explain how the data is prepared for the NN training.

5.3.2 Normalization

We describe how to pre-process the data before passing it to the NN. We introduce two different methods to normalization the data. The first method is the classical normalization, usually a min-max scaling. The second method is standardization, which is also often called normalization. First, we describe normalization. Normalization is a min-max scaling, which scales the data from a to b, where a is the minimum value and b is the maximum value in the data range. The min and max values are usually set to 0,1 or -1,1. The formula to scale one feature x in the data is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} * (a - b) - b, \tag{5.27}$$

where x' is the scaled data. Normalization can work better than standardization if the variance of the data set is very small or the underlying distribution is not Gaussian.
Standardization scales the data to a mean of zero and a standard deviation of 1. The scaled feature x hence reads:

$$x' = \frac{x - \overline{x}}{\sigma}, \tag{5.28}$$

where $\overline{x}$ is the mean and σ the standard deviation of all samples of the feature. In the remainder of this Thesis, only standardization is used to pre-process the data.

5.3.3 NN Training

A loss function is minimized in the training phase. We consider one loss functions in this work. However, for completeness, we introduce also the mean squared error (MSE), which is a commonly used loss function in regression tasks. The MSE is defined as:

$$L_{MSE}(\tilde{y}, y) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (\tilde{y}_i - y_i)^2, \tag{5.29}$$

where $y_i \in \mathbb{R}^n$ is the n-dimensional target output and $\tilde{y}_i \in \mathbb{R}^n$ is the predicted output. The second error definition is the relative Frobenius norm, which is equal to the approximation error definition in the linear case eq. (2.72), except for the square root. The error is defined as

$$L_F(\tilde{y}, y) = \sum_{j=1}^{n} \sum_{i=1}^{n_{\text{train}}} \frac{\sqrt{(\tilde{y}_{ij} - y_{ij})^2}}{\sqrt{y_{ij}^2}}, \tag{5.30}$$

where y_{ij} denotes the i-th sample of the j-th sample. This error definition delivered the best results in this work. The NNs are trained using the stochastic gradient descent (SGD) method or more sophisticated variants. In this method, the weights and biases are updated using the gradient of the loss function and a learning rate $lr \in \mathbb{R}$:

$$w_i = w_i - lr\frac{\partial L}{\partial w_i}, \tag{5.31}$$

or equally

$$b_i = b_i - lr\frac{\partial L}{\partial b_i}. \tag{5.32}$$

Neural Networks are compositions of functions. The derivative of compositions of functions can be obtained using the chain rule. That is, the gradient of the loss function with respect to a weight in the first layer of the network is the product of all gradients. We can illustrate this in a simple example:

$$y_i = y_i(y_{i-1}(y_{i-2}(w_k))), \tag{5.33}$$

where y_i is the output of layer i and w_k the currently considered weight. The gradient is then the product of all derivatives:

$$\frac{\partial y_i}{\partial w_k} = \frac{\partial y_i}{\partial y_{i-1}} \frac{\partial y_{i-1}}{\partial y_{i-2}} \frac{\partial y_{i-2}}{\partial w_k}. \tag{5.34}$$

The algorithm that computes the gradient of the loss function with respect to the weights based on the chain rule is named backpropagation, as the error is propagated backward through the NN. Refinements, such as adapting the learning rate during training, exist.

5.3.4 Autoencoders

In this subsection, we explain the concept of an AE architecture. An AE is a NN architecture used for unsupervised learning. It is used to learn a low-dimensional representation of the data, also called encoding, which captures the most relevant features. A simple AE is shown in Fig. 5.7. The AE consists of an encoder and a decoder. The encoder consists of multiple fully connected layers which transform the high-dimensional input into a low-dimensional representation. The low dimension of the bottleneck layer enforces the compressed representation. Nonlinear transformations due to nonlinear activation functions transform the input state in the hidden layers. The decoder also consists of fully connected layers. The hidden layers decode the compressed representation of the bottleneck layer. A nonlinear transformation is achieved if nonlinear activation functions are used. Finally, the output layer is usually a linear layer for continuous data. The linear layer allows for a large range of values. The input and target output values are the same for an AE, as it tries to compress the input state and decompress it again. Hyper-parameters of the AE are the width of the hidden layers, the latent dimension (bottleneck dimension), the number of layers, the activation function, the type of normalization, training parameters, and many other hyper-parameters.

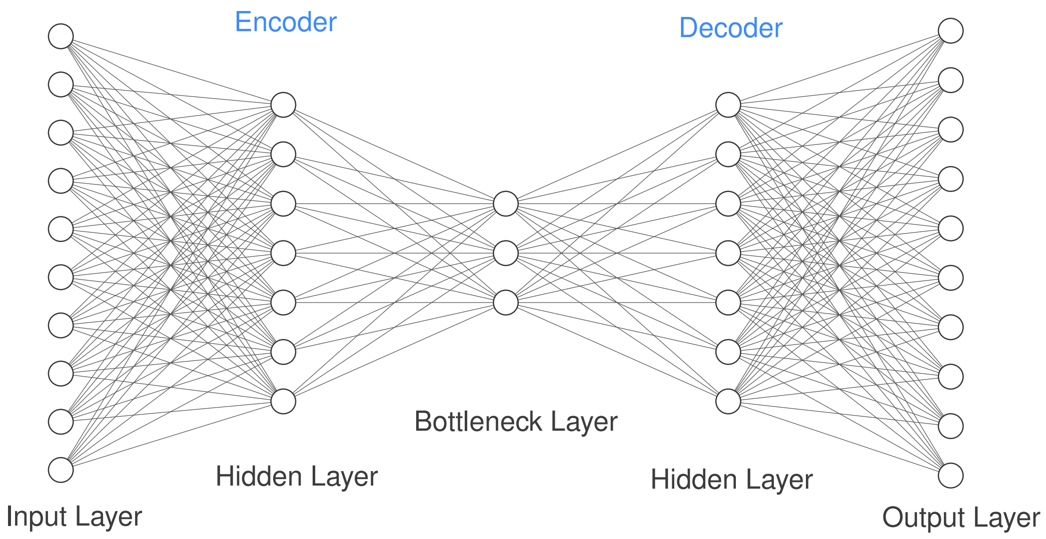


Figure 5.7 Architecture of an AE.

Classical AE

Using the AE introduced in the previous sections is considered the classical AE in this work. The input of the AE is the linearly reduced state $\hat{x}$, and we train the AE to reproduce this state. This means the target value is also the reduced state $\hat{x}$. Both loss functions, the MSE and the Frobenius norm, can be used and are assessed in the results section. To avoid overfitting, regularization can be added. We apply L^2 regularization on a per-layer basis. That is, the additional loss term per layer is obtained as:

$$L_{L^2} = \beta * \|\underline{w}\|_2^2. \tag{5.35}$$

The regularization parameter is named β and is usually small. The layer weights, also called the kernel, are denoted by $\underline{w}$. Regularization makes the AE more robust to unseen data and helps the model to generalize well.

Denoising AE

To make the AE more robust to small changes in the input that will occur due to errors in the ROM during the simulation, we propose to use the Denoising Auto Encoder (DAE). The DAE is a model to remove noise from custom data, such as images. The DAE differs from the normal AE in that the input and the output are unequal. The input to the DAE is the samples contained in the linearly reduced global snapshot matrix

$$\hat{\mathbf{U}}_{\text{glob}} = \mathbf{\Phi}^T \mathbf{U}_{\text{glob}} \in \mathbb{R}^{k \times N_p N_s}. \tag{5.36}$$

Before passing the samples to the DAE training, Gaussian noise is added to the data:

$$\hat{\mathbf{U}}^*_{\text{glob}} = \hat{\mathbf{U}}_{\text{glob}} + \mathcal{N}(\mu,\, \sigma^2), \tag{5.37}$$

where μ is the mean and σ^2 the variance of the Gaussian noise. We choose a zero mean, and the variance is an additional hyper-parameter to choose. The input is the noisy data, however, the target is the unaltered samples $\hat{\mathbf{U}}_{\text{glob}}$.

Contractive Loss AE

The last AE we discuss in this work is the contractive loss AE (CAE) [167], which adds a penalty term to the loss function that is equal to the Frobenius norm of the Jacobian matrix of the encoder with respect to the input. Similar to the AEs using regularization or noise, the CAE also aims to make the AE robust against small changes in the input. However, compared to the DAE, which uses randomness to achieve a robust AE, the CAE is based on the Jacobian of the NN and is deterministic. The penalty term for the CAE is the Jacobian of the encoder Θ with respect to the input x:

$$\|\mathbf{J}_\Theta(x)\|_F^2 = \sum_{ij} \left(\frac{\partial z_j}{\partial x_i}\right)^2. \tag{5.38}$$

A complete loss function for a CAE with Frobenius error would therefore read:

$$L_{CAE}(\tilde{x}, x) = L_F(\tilde{x}, x) + \alpha_J \|\mathbf{J}_\Theta(x)\|_F^2, \tag{5.39}$$

where the target output $y = x$ is the input and the regularization term is weighted with α_J.

5.3.5 Implementation Details

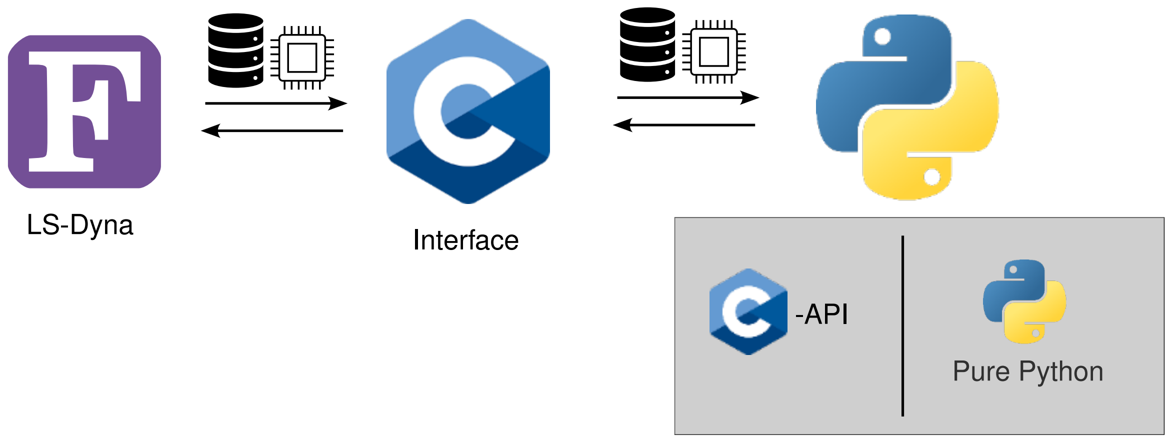


Figure 5.8 Structure of the Fortran-Python interface.

The implementation has presented a significant challenge, which is why we mention it in a subsection and give more details in Appendix A.4. The challenge arises because queries of the AE are required during the runtime of the solver (LS-Dyna). High-level application programming interfaces (APIs) exist for NNs in Python. Tensorflow [168] with its higher-level API, Keras [169], are popular ML platforms, and we decided to use them due to their powerful Python APIs. Using Keras, it is simple to quickly create NN and implement different variants of AEs. However, once the models are trained, the FEM solver needs to query the trained model in each iteration. Therefore, an efficient implementation is required to avoid computational overhead due to the data transfer. For us, the only possibility is to transfer the data directly inside the main memory. Writing the data to the disk and rereading it or a server-client approach for inter-process communication seems unsuitable. Directly passing data in the main memory is possible using the programming language C. Both applications can communicate with C as Fortran is interoperable with C using the iso-c-bindings, and the reference implementation of the Python interpreter is also in C. Further, Python offers a C-API, which can be used to control the interpreter. The structure of the Fortran-Python interface is shown in Fig. 5.8. In each iteration of the LS-Dyna solver, the prediction of the decoder of the AE is required. Starting from the very left side in Fig. 5.8, the data is passed to the C-interface, which correctly interprets the data. Next, the interface uses the Python C-API to create corresponding Python objects of the transferred data. Finally, the code written in pure Python can be executed and used for inference using the C-API. Equally, as the data is transferred from Fortran to Python, it is received again by C, interpreted correctly, and transferred back to Fortran. Once the AE is queried and the values are passed, the Fortran code continues and the next steps in the FEM solver are performed.

5.4 Results

In this section, we present the results of the AE ROM. The results are structured as follows: First, using a classical AE, we present the results of the training procedure. While keeping the dimensions of the AE model rather small, we investigate the effect of regularization, different activation functions, SVD and bottleneck dimensions, and the number of neurons in the hidden layers. Once a suitable AE model is identified, the classical AE is compared with the DAE and CAE, which can be considered AEs that apply a different type of regularization. Finally, hyper-reduction is applied, and the reduced mesh is shown for different hyper-reduction tolerances. The additionally introduced error is investigated, and the results are summarized and concluded.

5.4.1 AE Training

The AE is trained on the linearly reduced data. The global snapshot matrix $\mathbf{U}_{\text{glob}}$ is linearly reduced as described in eq. (5.36). The same snapshots are used as in Chapter 3 to construct the global snapshot matrix. This means there are $N_p = 27$ training simulations, as shown in Figure 3.1, and snapshots are taken every $0.01\,\text{ms}$ during $35\,\text{ms}$ for each parameter configuration. This results in a total of $N_pN_s = 27 * 3,501 = 94,527$ snapshots. A global SVD is applied on $\mathbf{U}_{\text{glob}}$ to obtain the global reduced basis $\mathbf{\Phi}$ which is then used for the linear reduction. The reduced data is then fed to the AE to train it. Before we describe the hyper-parameters of one specific AE, we summarize the training parameters in Tab. 5.2. The initial learning rate lr_0 is lowered to lr_{min} during the epochs e_0 and e_1. The learning rate is modified

Table 5.2 AE training parameters.

Training Parameter	Value
Optimization Algorithm	Adam [170]
Batch size	500
Validation portion	20%
Max epochs	500
Early stopping	20 epochs
lr_0	0.001
lr_{min}	0.0001
e_0	30
e_1	150

according to

$$lr = \begin{cases} lr_0, & e < e_0 \\ lr_0 \exp\left((e - e_0)\frac{\ln(\frac{lr_{\min}}{lr_0})}{e_1 - e_0}\right), & e_0 \leq e \leq e_1 \\ lr_{\min}, & e > e_1 \end{cases} . \tag{5.40}$$

The AE is trained for a maximum of 500 epochs. An epoch describes one cycle in which the network has seen all training data samples. The training data per epoch is 80% of the whole training data. The remaining 20% are reserved as validation data and randomly chosen in each epoch. The gradient of the loss function is computed using all samples in one batch, and the weights and biases are updated using the gradient. For efficiency, the network training stops if the loss function is not minimized further for 20 epochs. In case the network training stops early, the best weights are restored. To ensure reproducible results, the random generator's seed is set to 0 for the weight initialization and the noise generation in the DAE case.
Next, the AE architecture, which is used for the study of influential hyper-parameters is presented and tabulated in Tab. 5.3. A dimension for the outer linear ROM of $k_{SVD} = 30$ and a bottleneck dimension of $k_{AE} = 7$ is chosen. The hyper-parameters are primarily chosen to achieve small ROMs, which are fast to evaluate and, therefore, suitable to study the effects of hyper-parameter changes. The described AE is first

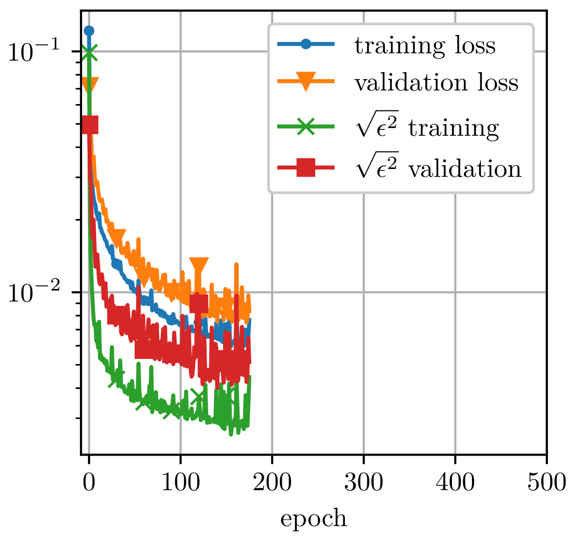
Table 5.3 AE hyper-parameters.

Hyper-parameter	**Encoder**	**Decoder**
k_{SVD}	30	30
k_{AE}	7	7
Layer width	80	80
Hidden layers	2	2
L^2-regularization	0.0001	0.0001
Activation function	Leaky ReLU	Leaky ReLU
Loss function(AE)	Frobenius	

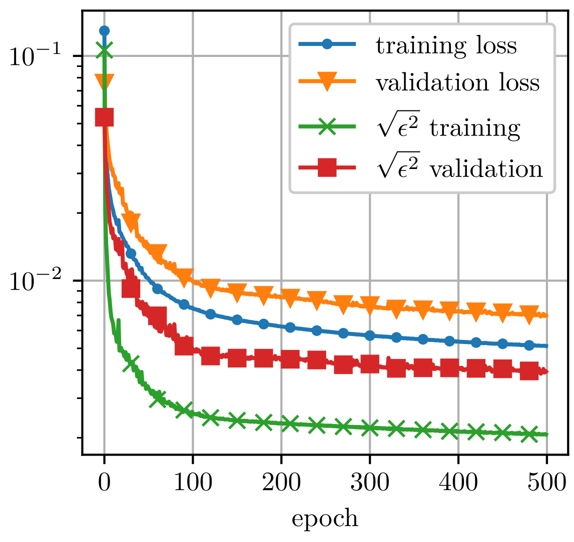
trained using a fixed lr of $lr = lr_0 = 0.001$ and compared to the training using a dynamic lr as described in eq. (5.40). Fig. 5.9 shows the results of the AE training. The figures show the loss and the relative Frobenius norm of the approximated data $\sqrt{\epsilon^2}$ during the epochs for the training and validation data set. The discrepancy between loss and Frobenius norm results from the loss, including the regularization term. The Frobenius norm error is defined as:

$$\sqrt{\epsilon^2} = \frac{\left\| \hat{\mathbf{U}} - \Gamma\left(\Theta\left(\hat{\mathbf{U}}\right)\right) \right\|_F}{\left\| \hat{\mathbf{U}} \right\|_F}. \tag{5.41}$$

Except for the square root, the error definition is equivalent to the error definition of the linear dimensionality reduction (eq. (2.72)). To emphasize the similarity to the error of the linear dimensionality reduction, the notation with root and square is chosen. Although it is unusual to train a network with the square root of the error, we obtained the best and most stable results using this error definition. Fig. 5.9a shows the training history for a fixed learning rate. It can be seen that the learning rate is too large, and the loss becomes unstable. The network's weights and biases are updated according to the loss of one epoch, calculated using the snapshots of the considered batches. The weight update is also linearly dependent on the learning rate. The weight update is too large if the learning rate is too large. The weights can become too specific for the current batch, and the new randomly chosen training and validation data sets are predicted badly. Adjusting the learning rate mitigates this issue, as shown in Fig. 5.9b. The error decays smoothly during training. However, the training takes longer.



(a) AE training using a fixed lr=0.001.



(b) AE training using the adjusted lr.

Figure 5.9

5.4.2 Regularization

Regularization prevents overfitting. Overfitting is when a model learns the training data too specifically but fails to generalize and makes accurate predictions on new data. The first proposed AE (see Tab. 5.3) contained a small portion of weight regularization. In the following, we show results for an AE without regularization, a small amount of regularization, a reasonable amount of regularization, and too much regularization. Tab. 5.4 tabulates four different AEs using a different amount of L^2 weight regularization. All AEs share the same architecture with an input dimension of 30, a bottleneck dimension of 7, Leaky ReLU activation functions, two hidden layers in the encoder and two hidden layers in the decoder, a layer width of 80, and the proposed relative Frobenius norm error. First, we compare the achieved approximation error. The total approximation error for the composition of SVD and AE is defined as:

Table 5.4 AE hyper-parameters for regularization study.

Hyper-parameter	**AE1**	**AE2**	**AE3**	**AE4**
k_{SVD}	30	30	30	30
k_{AE}	7	7	7	7
Layer width	80	80	80	80
Hidden layers Θ	2	2	2	2
Hidden layers Γ	2	2	2	2
L^2-regularization	0	0.00001	0.00005	0.0001
Activation function	Leaky ReLU	Leaky ReLU	Leaky ReLU	Leaky ReLU
Loss function(AE)	Frobenius	Frobenius	Frobenius	Frobenius

$$\epsilon_{tot}^2 = \frac{\left\| \mathbf{U} - \mathbf{\Phi}\Gamma\left(\Theta\left(\mathbf{\Phi}^T\mathbf{U}\right)\right)\right\|_F^2}{\|\mathbf{U}\|_F^2}, \tag{5.42}$$

which is the logical extension of the linear error to include the AE. Therefore, the values are comparable to the linear results. Tab. 5.5 summarizes the achieved approximation errors for the different AEs. The error introduced by the linear approximation is termed ϵ_{SVD}^2 and is equal for all AEs, as all use the same

Table 5.5 Approximation error for the regularized AE variants.

Error	**AE1**	**AE2**	**AE3**	**AE4**
ϵ^2_{SVD}	1.499×10^{-6}	1.499×10^{-6}	1.499×10^{-6}	1.499×10^{-6}
ϵ^2_{AE}	1.044×10^{-5}	6.507×10^{-6}	4.269×10^{-6}	4.696×10^{-6}
ϵ^2_{tot}	1.194×10^{-5}	8.006×10^{-6}	5.768×10^{-6}	6.195×10^{-6}

SVD dimension k_{SVD}. The error solely introduced by the AE is named ϵ^2_{AE}. AE3 achieves the lowest error and AE1 the highest one. The error is composed of two parts. One part of the error is governed by the approximation of the 80% training data and the other part by the 20% validation data in the global training data set. The validation data governs how well the AE generalizes to unseen data. AE3 with the medium regularization factor of all tested values achieves the lowest error, and AE1 without regularization yields the largest error. To relate the AE approximation errors to the linear ones of the previous chapters, we plot the error values compared to the SVD approximation error in Fig. 5.10. The vertical line indicates the AE-dimension $k_{AE} = 7$. All AEs yield lower errors than the SVD with seven dimensions. The global SVD error is the curve that decays for larger approximation ranks k. Due to the logarithmic y-scale, all AE errors are close to each other. Next, we plug the trained AEs into the AE ROM and assess the time-dependent

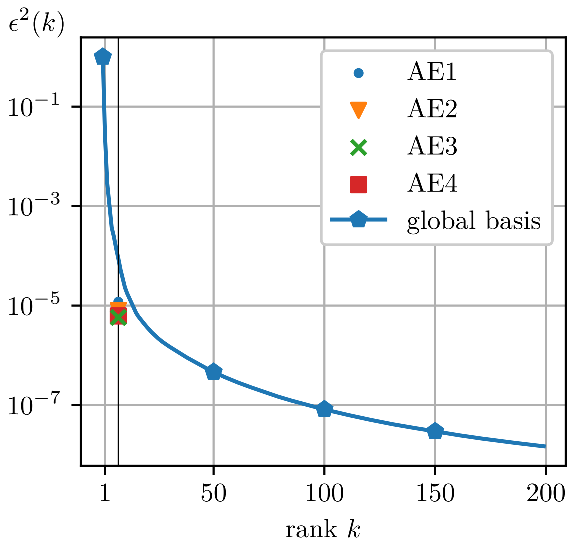
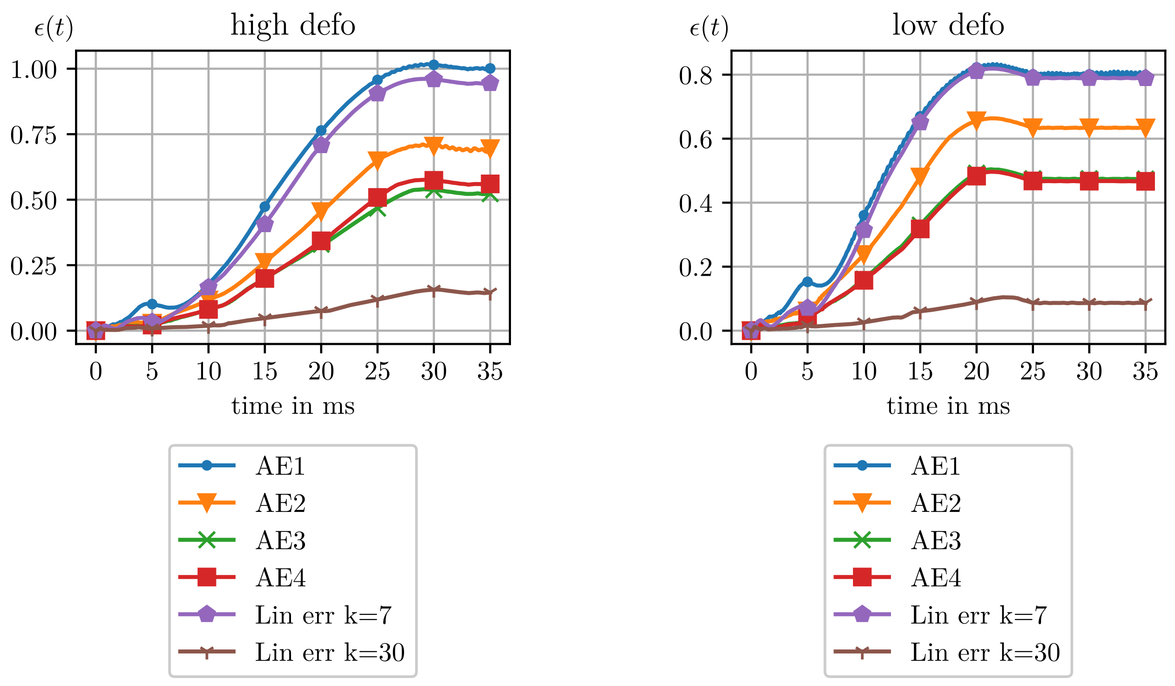


Figure 5.10 Offline accuracy of the AEs in comparison to the linear global SVD.

displacement error $\epsilon(t)$. The error is plotted over time for all AEs in Fig. 5.11. The high deformation test case is plotted in Fig. 5.11a and the low deformation test case is shown in Fig. 5.11b. In addition to the AE ROMs, the online error for a global linear ROM with $k = 30$ and $k = 7$ are also compared. The AE ROMs show the same error behavior as the linear ROM. The error accumulates over time and reaches a steady state after the rebound phase, where it decreases due to the solutions' approach. Equal to the linear ROMs, the error for the high deformation test case is larger than for the low deformation test case. AE3 yields the lowest accuracy in the high deformation test case. However, in the low deformation test case, AE4 is slightly more accurate. The offline approximation error is defined globally. Therefore, although the overall error is smaller for AE3, it can be larger in specific regions of the solution space, leading to less accurate results for different parameter variations. AE1 without regularization yields large errors. The error is larger than the error of the linear ROM using the same bottleneck dimension. The linear ROM using the input dimension of the AE is more accurate than the AE ROMs. However, the approximation error of the SVD alone is smaller than the approximation error of the AEs using seven dimensions. To further understand the large error of the AE ROM without regularization, we plot the latent



(a) Results of the AE ROM for the high deformation test case.

(b) Results of the AE ROM for the low deformation test case.

Figure 5.11

space variables z_i for $i = 1, .., k_{AE}$ of the AE ROM over time for the AE without and with regularization in Fig. 5.12 and Fig. 5.13, respectively. Without regularization, the ROM solution quickly deviates from the reference solution and reaches a steady state. The steady state is quickly reached if the tube behaves too stiff and the energy of the impacting plate is consumed, or the plate springs back with more kinetic energy. Locking is observed if the solution's function space cannot provide the required deformation or if the tangent space to the manifold, spanned by the Jacobian matrix of the AE, is inaccurate and does not provide the correct deformation directions. Locking is, therefore, related to the approximation quality of the AE. Without regularization, the AE overfits the data and is not robust to small errors. These errors strongly influence the prediction and lead again to wrong predictions. The system locks, and the latent space trajectories quickly deviate from the reference solution. Also, oscillations towards the end of the simulation time indicate a high stiffness and errors in the approximation of the AE. The regularized AE ROM can follow the reference trajectories much longer. The steady state is reached at later simulation times, and no high-frequency oscillations are observed. To conclude, regularization makes the AEs more robust to small errors, resulting in more accurate AE ROMs.

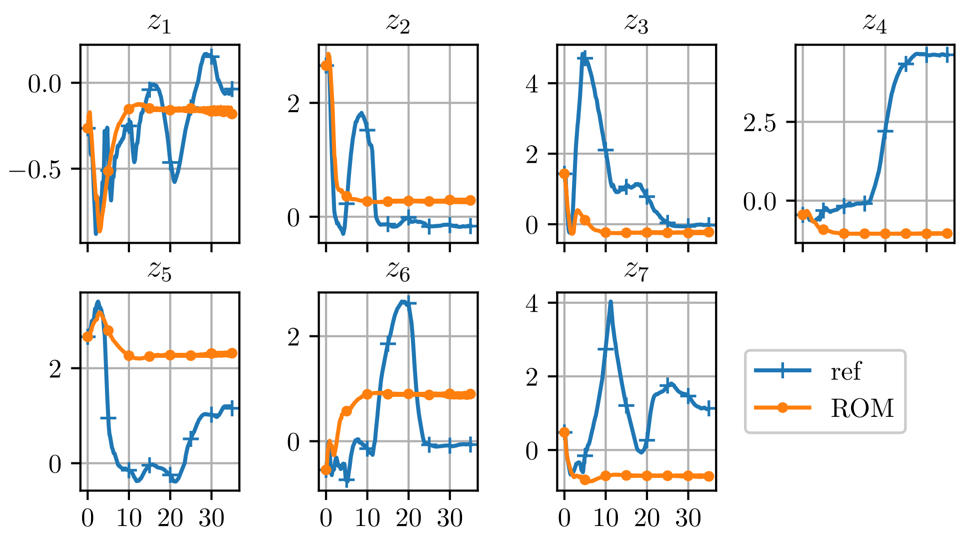


Figure 5.12 Latent space variables over time for AE1 without regularization.

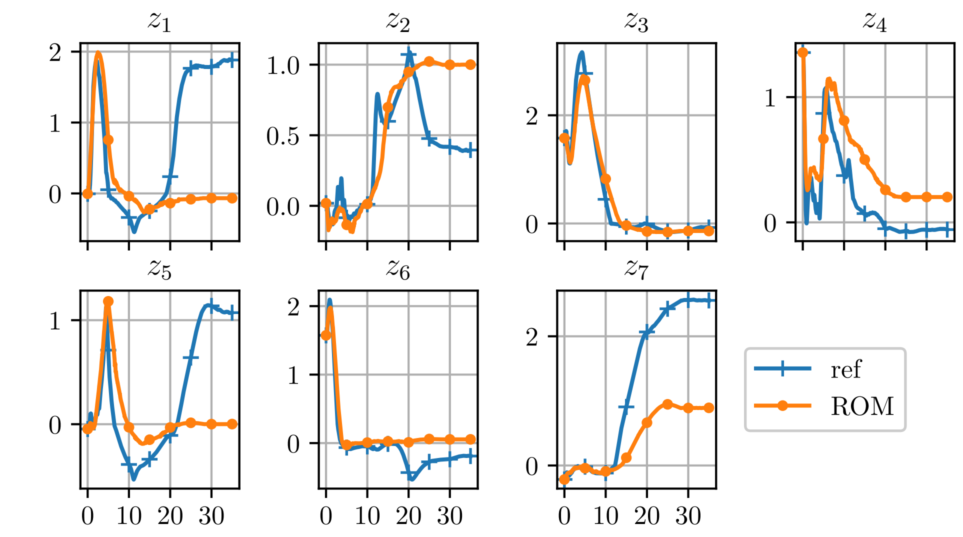


Figure 5.13 Latent space variables over time for AE3 with regularization.

5.4.3 Influence of Activation Function

So far, we considered nonlinear activation functions consisting of linear parts. This means the Hessian is zero, and the associated term in eq. (5.18) is also zero. We utilize a smooth version of the leaky ReLU function to investigate the effect of second-order terms. The smooth ReLU, as listed in Tab. 5.1, is continuously differentiable, and the Hessian matrix differs from zero. To study the influence of the curvature term, we choose the best AE from the previous section (AE3 in Tab. 5.4) and replace the activation function with the smooth ReLU function. The new hyper-parameters of the AE using the smooth ReLU activation function read: First, we assess the achieved approximation error. Tab. 5.7 summarizes the three different error measures, as defined before (see Tab. 5.10). The smooth ReLU AE yields larger approximation errors than AE3 using leaky ReLU activation functions after training. This is directly mirrored in Fig. 5.14,

Table 5.6 AE hyper-parameters with smooth ReLU activation function.

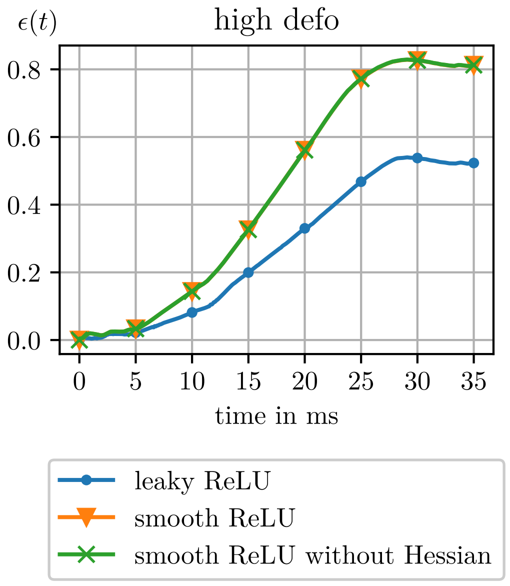
Hyper-parameter	
k_{SVD}	30
k_{AE}	7
Layer width	80
Hidden layers Θ	2
Hidden layers Γ	2
L^2-regularization	0.00005
Activation function	Smooth ReLU
Loss function(AE)	Frobenius

where the leaky ReLU AE ROM yields lower errors than the smooth version. To still assess the influence

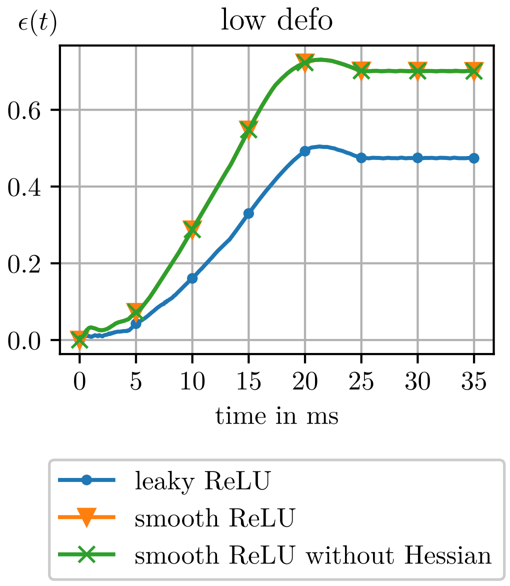
Table 5.7 Approximation error for the regularized AE variants.

Error	**AE3**	**AE with smooth ReLU**
ϵ^2_{SVD}	1.499×10^{-6}	1.499×10^{-6}
ϵ^2_{AE}	4.269×10^{-6}	16.878×10^{-6}
ϵ^2_{tot}	5.768×10^{-6}	18.377×10^{-6}

of the curvature term, the smooth ReLU AE ROM is evaluated neglecting the curvature term. In the high deformation test case Fig. 5.14a as well as in the low deformation test case 5.17b, the effect of the curvature of the manifold is negligible. The error of the ROM, including the curvature term, is equal to the error of the ROM without the Hessian matrix. The region of the activation function with curvature is small



(a) Time dependent error of the AE ROMs with and without curvature for the high deformation test case.



(b) Time dependent error of the AE ROMs with and without curvature for the low deformation test case.

Figure 5.14

and is located around zero. We further assess the impact of the curvature term by observing the term:

$$\|f_{\text{curv}}\|_2 = \left\|\tilde{\mathbf{M}}\mathbf{H}\dot{z}\dot{z}\right\|_2 . \tag{5.43}$$

Fig. 5.15 shows $\|f_{\mathrm{curv}}\|_2$ during the simulation time for the high deformation and low deformation test case. In both cases, the curvature term differs from zero only before 15 ms. Finally, we emphasize that the AE ROM term, including the curvature, is the projection of f_{curv},

$$\mathbf{J}^T f_{\mathrm{curv}}, \tag{5.44}$$

onto the tangent space spanned by the columns of $\mathbf{J}$. Since the AE ROM, including the curvature term and the AE ROM without it, do not differ, we conclude that the projection of the depicted term is negligible compared to the projection of the internal force term.

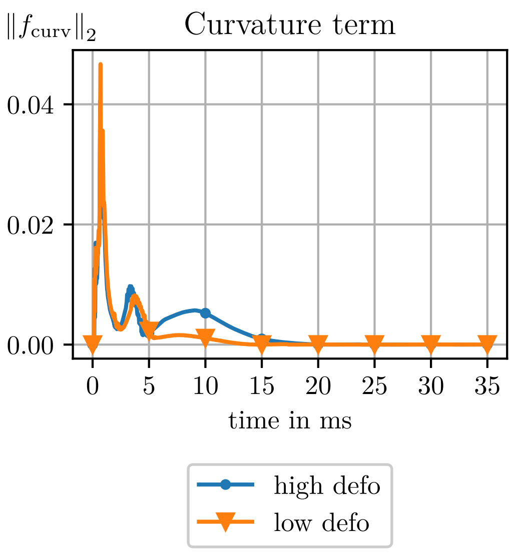


Figure 5.15 Norm of curvature term in the AE ROM.

5.4.4 Influence of Dimensions

Next, we investigate how varying the dimensions of the input k_{SVD}, of the bottleneck k_{AE}, and the hidden layers influences the accuracy of the AE and the derived AE ROM. We introduce three new AEs in which one parameter is varied. The AEs are named according to:

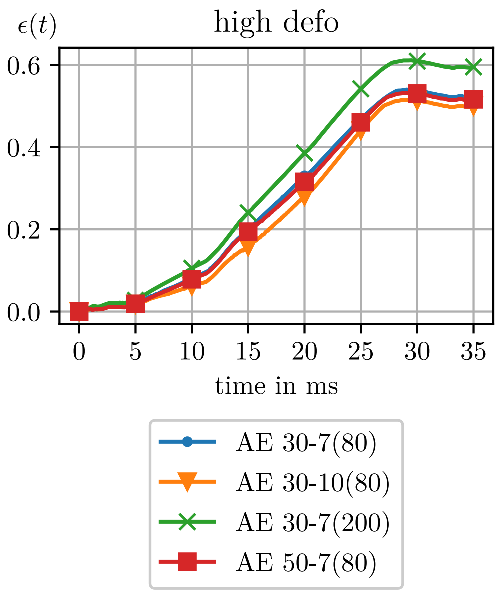
$$\mathbf{AE}\ k_{SVD}\text{–}k_{AE}\,(\text{layerwidth})\,.$$

The first AE is the best regularized AE, named AE3 in Tab. 5.4. Then, the bottleneck dimension is increased to 10, the layer width is increased to 200, and finally, the input dimension is increased to 50. The dimensionality variations' impact on the approximation error can be seen in Tab. 5.8. The only way to decrease the linear approximation error ϵ^2_{SVD} is to increase the input dimension, which can be seen for AE 50-7(80). However, increasing the linear approximation accuracy yields only a small decrease in total error. Also, increasing the layer width does not significantly improve the total approximation error. This is also because the weight regularization parameter is not adjusted, although the number of weights increases. Increasing the bottleneck dimension yields an improvement in the total approximation error. The corresponding time-dependent online errors can be seen in Fig. 5.16a and Fig. 5.16b for the high deformation and low deformation test case, respectively. The increased bottleneck dimension leads to an improvement in online accuracy, especially for the low-deformation test case. Increasing the layer width without modifying the regularization parameter yields a decrease in online error while maintaining a similar offline error. Increasing the input dimension yields a minor accuracy improvement. We conclude that the

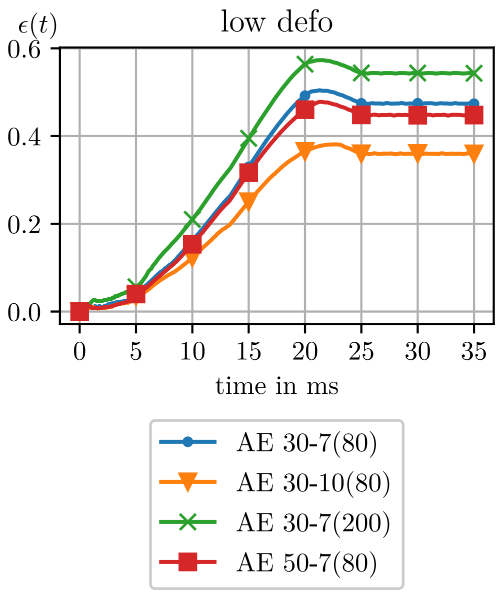
Table 5.8 Approximation error for the AE variants with different dimensions.

Error	AE 30-7(80)	AE 30-10(80)	AE 30-7(200)	AE 50-7(80)
ϵ^2_{SVD}	1.499×10^{-6}	1.499×10^{-6}	1.499×10^{-6}	4.632×10^{-7}
ϵ^2_{AE}	4.269×10^{-6}	3.775×10^{-6}	4.153×10^{-6}	5.164×10^{-6}
ϵ^2_{tot}	5.768×10^{-6}	5.274×10^{-6}	5.652×10^{-6}	5.627×10^{-6}

offline approximation error greatly influences the online error. However, maintaining a smooth and well-behaved latent representation is of similar importance, as a small offline accuracy improvement does not necessarily lead to a better AE ROM.



(a) Time dependent error of AE ROMs with different dimensions for the high deformation test case.



(b) Time dependent error of AE ROMs with different dimensions for the low deformation test case.

Figure 5.16

5.4.5 CAE and DAE

Before applying hyper-reduction, the standard AE is compared to the CAE and DAE. Both AE types represent different types of regularization. The DAE makes the AE robust by first inserting noise in the data and learning the unaltered data. The CAE promotes a robust latent space by regularizing the AE with the Frobenius norm of the encoder's Jacobian matrix. The encoder's Jacobian relates (infinitesimal) increments of the input to increments in the latent space representation.
For the following comparison, we choose the hyper-parameters of the AEs according to Tab. 5.9. The AE training yields the offline approximation error as tabulated in 5.10 using these hyper-parameters. The CAE yields the lowest approximation error and DAE the largest one. However, the CAE yields less accurate results in the online phase than the AE, as shown in Fig. 5.17. The AE ROMs are evaluated for the high deformation test case in Fig. 5.17a and the low deformation test case in Fig. 5.17b. The error of the linear ROM using $k_{SVD} = 10$ is plotted for comparison. The final deformation of the DAE in the high deformation test case is similar to the final deformation obtained by the CAE. However, the DAE yields a nonphysical deformation history. We assume this is due to the stochastic nature of this method. Also, the DAE yields large errors for the low deformation test case. Although the offline error of the CAE is smaller than that

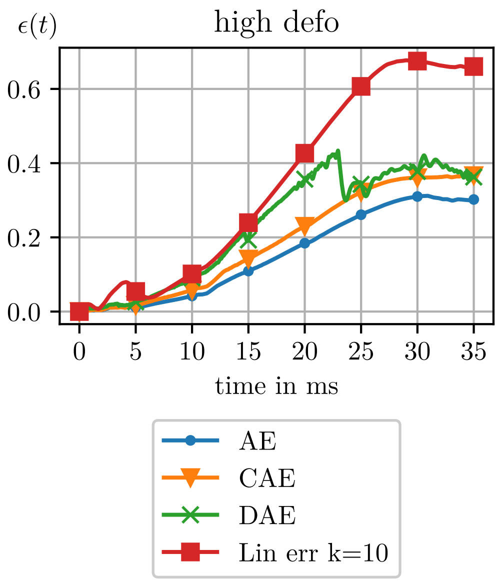
Table 5.9 AE hyper-parameters for AE-type comparison.

Hyper-parameter	AE	CAE	DAE
k_{SVD}	50	50	50
k_{AE}	10	10	10
Layer width	150	150	150
Hidden layers Θ	2	2	2
Hidden layers Γ	2	2	2
L^2-regularization	5×10^{-5}	0.0001	0.0625
Activation function	Leaky ReLU	Leaky ReLU	Leaky ReLU
Loss function(AE)	Frobenius	Frobenius	Frobenius

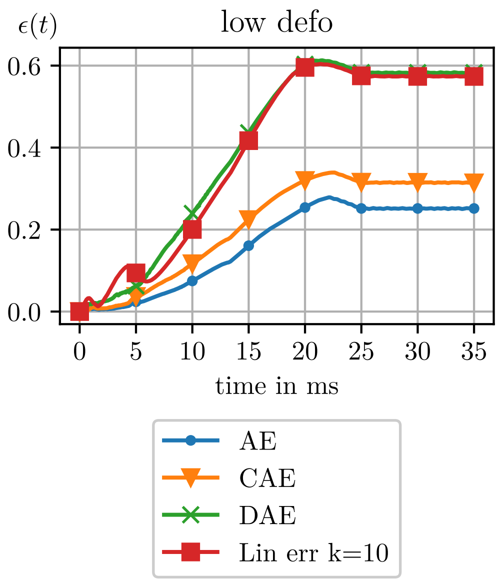
of the AE, the CAE has a slightly larger online error than the AE. This supports our assumption that the properties of the embedding play an equally important role than the approximation accuracy itself. The following subsection shows the compatibility of ECSW hyper-reduction with the AE ROM.

Table 5.10 Approximation error for the different AE variants.

Error	AE	CAE	DAE
ϵ^2_{SVD}	4.632×10^{-7}	4.632×10^{-7}	4.632×10^{-7}
ϵ^2_{AE}	6.868×10^{-6}	3.252×10^{-6}	9.525×10^{-6}
ϵ^2_{tot}	7.331×10^{-6}	3.715×10^{-6}	9.988×10^{-6}



(a) AE ROM accuracy for different AE variants tested on the high deformation test case.



(b) AE ROM accuracy for different AE variants tested on the low deformation test case.

Figure 5.17

5.4.6 Hyper-reduction

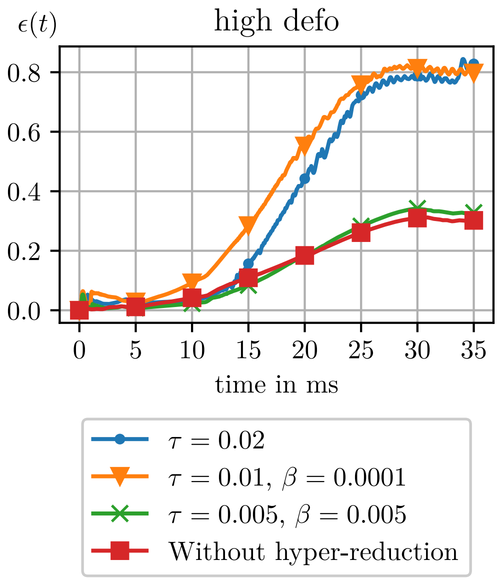
Finally, we also show the compatibility of ECSW hyper-reduction with the AE ROM. We choose the most accurate AE, the L^2 weight-regularized AE, as stated in Tab. 5.9. Like the previous chapters, hyper-reduction is the only way to achieve computational speedup for the AE ROM. Since the evaluation of the AE and the computation of the Jacobian matrix in each iteration are associated with additional computational costs, the small reduced mesh size due to the low dimensions of the AE ROM must compensate for this additional effort.
Equal to previous chapters, force snapshots are collected every $0.01\,\mathrm{ms}$ for all 27 training simulations. These snapshots and the trained AE are used to construct the matrix G_{AE} eq. (5.23) and the vector b_{AE} eq. (5.24). Using these assembled quantities, we compute the reduced mesh by solving eq. (3.61). The reduced mesh sizes are tabulated in Tab. 5.11 for different tolerances τ and regularization factors β. The

Table 5.11 Hyper-reduction offline results for the most accurate AE.

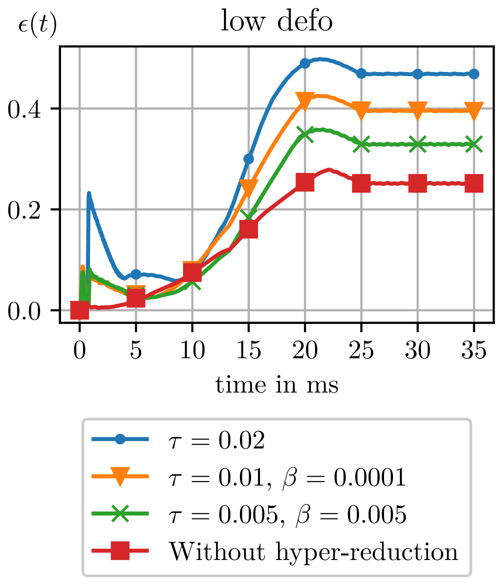
Tolerance / Regularization	$\tau = 0.02, \beta = 0.0$	$\tau = 0.01, \beta = 0.0001$	$\tau = 0.005, \beta = 0.005$
$\lvert\tilde{\mathrm{E}}\rvert$	124	183	276
Reduction in %	93.34%	90.18%	85.19%

optimization problem scales with the bottleneck dimension of the AE. Therefore, we obtain small reduced meshes even though all snapshots are included in a global sense. The reduced mesh size scales with the tolerance τ. A small tolerance also yields a larger reduced mesh. Regularization is also used to stabilize the hyper-reduced ROM and make it more accurate. The time-dependent error for the hyper-reduced AE ROMs is shown in Fig. 5.18a and Fig. 5.18b for the high deforming and low deforming test case, respectively. The most accurate hyper-reduced ROM is also the ROM with the lowest tolerance. Additional error is mainly introduced in the low deformation test case. The additional error in the high deformation test case is small. Larger hyper-reduction tolerances lead to unrobust and partially nonphysical ROMs. Problems arise when the AE is sensitive to small errors and the latent space variables deviate from the reference trajectories on which the hyper-reduced ROM was trained. This can be seen in Fig. 5.19, where the latent space trajectories of the ROM, the hyper-reduced ROM, and the reference are compared for all ten latent space variables and the low deformation test case. Especially in the first milliseconds of the simulation, the AE ROM and the hyper-reduced AE ROM cannot follow the strongly varying reference trajectory. This is mirrored by the jump in error occurring during the first millisecond of the simulation.

Finally, the reduced mesh with a reduction of 85.19% of all deformable elements is shown in Fig. 5.20a for the hyper-reduction parameters $\tau = 0.005$ and $\beta = 0.005$ in the initial configuration. The elements are colored according to their weights. The final deformed configuration for the high deformation test case is shown in Fig. 5.20b. The reference solution is colored turquoise for comparison.



(a) Hyper-reduced AE ROMs tested on the high deformation test case.



(b) Hyper-reduced AE ROMs tested on the low deformation test case.

Figure 5.18

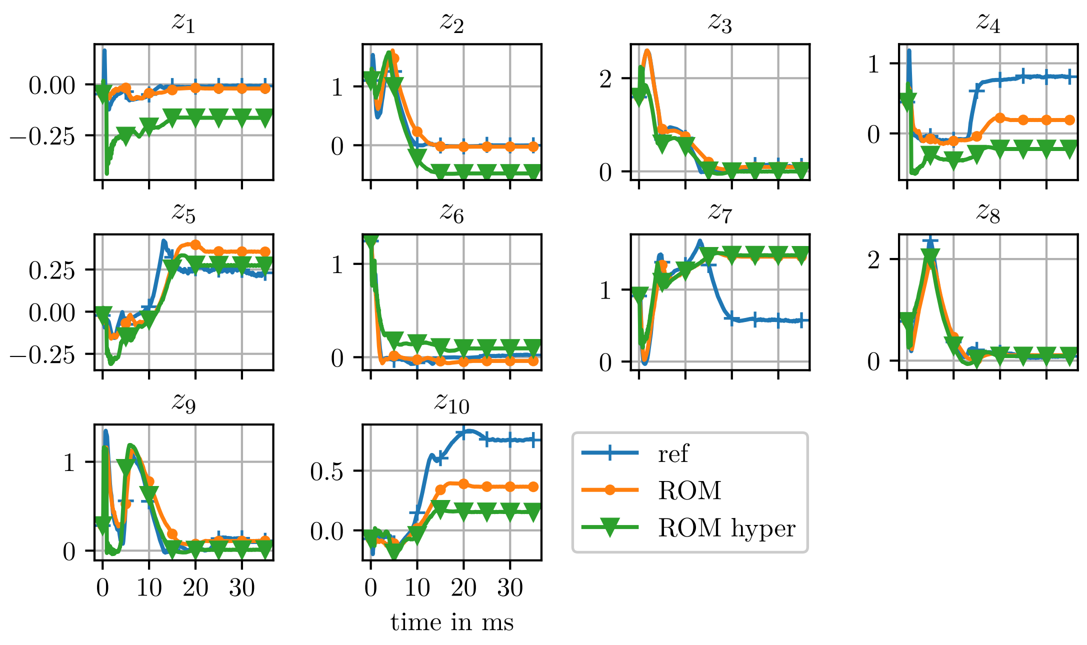
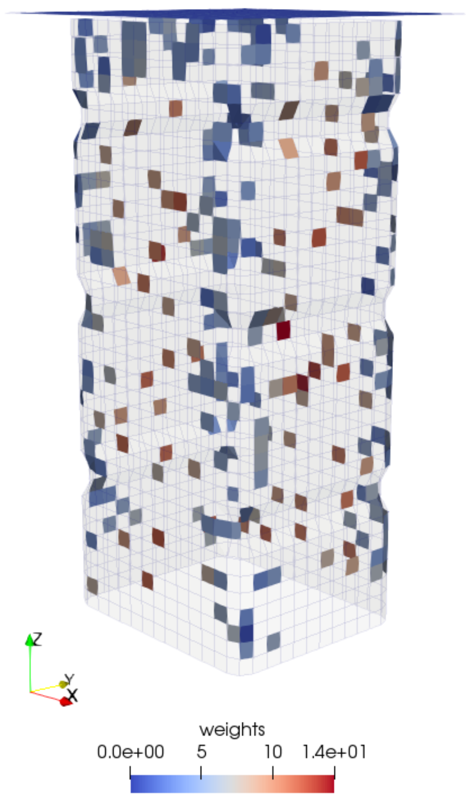
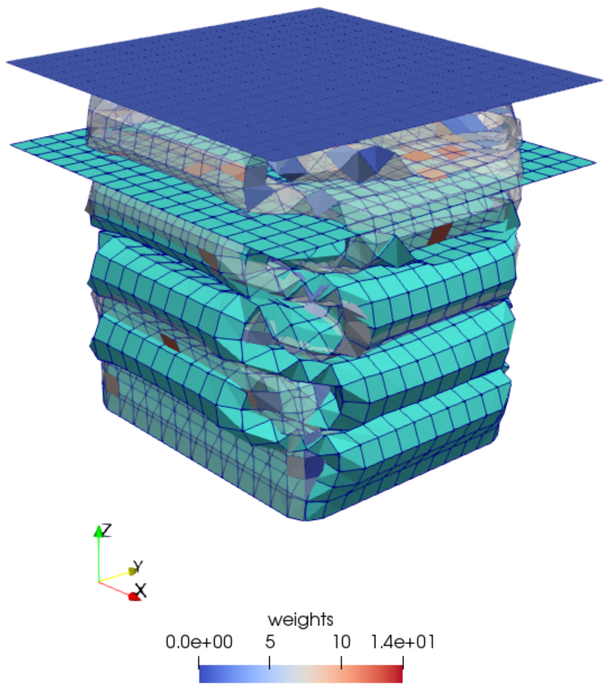


Figure 5.19 Latent space trajectories of the AE ROM and most accurate hyper-reduced AE ROM in comparison to the reference solution for the low deformation test case.



(a) Hyper-reduced AE ROM for $\tau = 0.005$ and $\beta = 0.005$.



(b) Hyper-reduced AE ROM with $\tau = 0.005$ and $\beta = 0.005$ tested on the high deformation test case. The reference solution is colored turquoise.

Figure 5.20

5.5 Conclusion

In this chapter, we introduced projection-based MOR on nonlinear manifolds based on AEs. This chapter aims to prove the concept of combining SVD with an AE to reduce dimensions further. The additional dimension reduction enables smaller reduced meshes, which is one major speedup mechanism in nonlinear explicit FEM. The implementation posed a challenge, so we included a small section. Finite element codes have usually grown historically and are, therefore, programmed in Fortran. Modern ML libraries often utilize a high-level programming language like Python to make the use easy and comfortable. This enables the user to quickly implement a new model. However, it also obscures the internals of the ML library. We present an exemplary implementation that uses the Python C-API to pass the data directly inside the main memory.

The potential computational speedup is due to two mechanisms. The first mechanism is demonstrated in this chapter and is the small reduced mesh. ECSW hyper-reduction scales with the dimension of the ROM. An AE can further reduce the ROM's dimensions compared to a global linear SVD. A ROM based on a global SVD requires many dimensions to achieve reasonable accuracy. However, small to no speedups are expected for large dimensions due to the resulting large reduced mesh. An AE with a bottleneck dimension of 10 already yields a reduced mesh, which contains 15% of the original model's elements. The second mechanism is the increased critical time step. Although not investigated in this work, a derivation for partially linear activation functions, such as the leaky ReLU function, would be easily possible. These two mechanisms must compensate for the additional effort required to evaluate the AE, including the Jacobian and Hessian matrix computations. Using partially linear functions speeds up the computation enormously since the Hessian computation is neglected. Computing the Hessian matrix is associated with high costs, as one Hessian calculation equals multiple Jacobian evaluations. Also, the AE ROM term associated with the manifold's curvature must be evaluated. However, our example shows that the Hessian can be neglected in our example. We trained an AE on a smooth version of the leaky ReLU function and compared the AE ROM, including the Hessian, with the AE neglecting the Hessian term. The solutions nearly overlap, and the evaluation of the Hessian term also showed that the solution transverses curved regions of the manifold in only a tiny fraction of the entire solution.

The curvature is small and for most of the solution parts zero. This could be seen as an indicator that the transformation is not nonlinear enough. Also, the achieved approximation errors are not as low as expected. In addition, the approximation error for the training and validation data differs during training. The AE cannot learn a latent representation during training that generalizes well enough to the validation data. The observation supports that the AE architecture is not yet suitable to fully transform the data in an accurate low-dimensional representation. In addition, we observe that the approximation error is not the only indicator for an accurate ROM. We draw the same conclusion as in the previous chapters: some additional knowledge about the system must be used. Otherwise, the AE is just a purely data-driven method and is not informed about the physics. Similarly to the linear case, more dimensions must be used to accurately predict the system's evolution, which prevents a fast ROM.

To summarize, pMOR on nonlinear manifolds is the logical extension of pMOR on linear manifolds. NNs, and more specifically, AEs, are a popular dimensional reduction method. Powerful and efficient frameworks exist that enable automatic differentiation (AD). AD is a convenient way to analytically differentiate the nonlinear mapping and provides the required derivatives for the ROM. AEs yield low-dimensional representations of the solution data, which can be used to build projection-based ROMs. While global linear dimensional reduction fails to provide a low-dimensional mapping to create a ROM, the AE can provide such a mapping. The low-dimensional mapping enables effective hyper-reduction, which can compensate for the additional costs due to the AE evaluation. Also, the global reduced mesh avoids evaluating zero elements, as needed for local reduced bases with history variables. An additional speedup mechanism is the enlargement of the critical time step, which is not discussed in this thesis.

The prospect of low-dimensional fast ROMs for parametric problems encourages future research. The research can be located both in the implementation and mathematical foundations. More efficient Jacobian or Hessian calculation implementations can be introduced by using Jacobian vector products (JVP)

or Hessian vector products (HVP). Also, the interaction between the FEM solver and the ML library can be improved and parallelized. New mathematical foundations emerged towards the end of this thesis, which were published lately [165, 166] and not available during the formation of this work. Romor et al. [165] use convolutional AEs to formulate a ROM and test it on a 2d non-linear conservation law and a 2d shallow water model. Otto et al. [166] present AEs with invertible activation functions and constraints on the weight matrices. Also, an oblique projection is proposed. The necessity of an oblique projection is in agreement with our observation that the accuracy of the AE alone is not sufficient for an accurate ROM.
In conclusion, despite many unresolved questions, MOR on nonlinear manifolds is an exciting field of research, and many advances can be expected in the next few years.

6 Outlook and Conclusion

6.1 Limits and Outlook

As we delve deeper into the intricacies of our research, it is also important to acknowledge the boundaries that shaped the scope of our research and point out future directions. Delineating the boundaries helps us to maintain intellectual rigor and prompts critical reflection on the precision and generalizability of our findings. Thereby, the transparency and integrity of our work are improved, and future researchers are supported by a detailed understanding of the context within which our contributions are situated. By accepting our limits and discussing the limits with curiosity, we contribute to the scientific dialogue that propels scientific research forward.
In accordance with the chapter layout, we discuss further research directions:

Identify the latent dynamics: In the pMOR approach, a subspace or manifold is calculated based on existent solution data. Afterward, the governing equations are transformed to the reduced space to obtain the latent dynamics. However, freedom is involved in the transformation process, such as the projection step, in which an orthogonal or oblique projection can be chosen. Therefore, it is not automatically guaranteed to obtain the correct latent dynamics. In contrast, data-driven approaches identify the latent dynamics directly once the dimensionality reduction is performed. Methods such as operator inference and Sindy restrict the system to certain types of function, whereas, for instance, recurrent neural networks allow for complex and unknown dynamics. An application to highly complex systems involving history-dependent variables and contact forces remains. Identifying the dynamics in reduced space circumvents hyper-reduction, as no high dimensions are involved anymore.

Learn or identify the nonlinear force term: Similar to learning the whole dynamics of the system in reduced space, a ML model can learn single terms of the system. Learning the reduced force term would deliver large speedups as no HDM dimensions are involved in the force evaluation. This is a similar approach to data-driven constitutive modeling. However, we want to learn the reduced force term directly. This means that operations, such as spatial integration and projection, which are required to solve the HDM, are involved in the operator we want to learn. Nevertheless, successful applications exist for simple systems with hyperelastic materials.

MOR of contact forces: The share of contact force evaluation in the total computation time can exceed 50% for large crash models. That is, if hyper-reduction reduces the number of elements to be evaluated by 50%, then the computation time is reduced by 25%. The example assumes a ratio of 50:50 between internal force evaluation and contact force evaluation. The example further illustrates the necessity of MOR for contact forces. However, MOR for contact mechanics is a field of research itself and is out of scope for this work.

Physics-informed dimensionality reduction: POD is a purely data-driven method. Although the data variance can be interpreted as energy, the results in this thesis do not allow for a direct correlation between the approximation error of POD and the accuracy of the resulting ROM. The balanced truncation method introduces concepts such as observability and controllability and yields pROMs with quality and stability guarantees. Due to the computation of a Gramian matrix, which is computationally intensive, the method is unsuitable for large systems. Also, the method was initially applied to linear systems. However, recent developments transfer balanced truncation to nonlinear systems [73].

Local ROB with critical time step adjustment: The adjustment of the critical time step was impossible due to missing source code access. Bach [1] has proven that the critical time step of the hyper-reduced

ROM is larger or equally large as the critical time step of the HDM. Experience emphasizes that the critical time step correlates with the dimension of the ROM. The dimension of the lROBs is small compared to a global ROM. This would result in large speedups. Enlarging the critical time step is a potent speedup mechanism, affecting all parts of the solver, including the contact force evaluation.

Hyper-reduction with history dependency: In the local ROB approach, we introduced three variants of ECSW hyper-reduction and identified the global-local most suitable for our purpose. A global reduced mesh is used with local weights. That is, the weights are cluster-specific, and the mesh is global. This is necessary due to the history dependency. If we assume that an element was not active in the initial cluster of the simulation and the cluster changes during the simulation, then the element is activated. In this case, the deformation history is missing. This means although the element is weighted with zero due to the local refinement of the weights, the element must be evaluated and consumes computation time. An idea to avoid evaluating zero elements is to reconstruct the field of internal variables using a gappy POD approach. This would transfer the global-local approach to a fully local-local approach. Then, the local reduced mesh is also smaller than the global mesh, which results in further speedups.

Space-time local ROB: MOR based on local ROB clusters the solution in state space. Since crash systems usually deform monotonically in one direction, this is similar to clustering along the temporal axis. Depending on the velocity, certain solution parts are assigned to clusters. However, the computed modes are still global to the whole model. However, the solution can be divided locally in space too. Recently, Anderson [171] proposed space-local ROBs. Separating the spatial domain could yield better results, especially in models where deformations appear locally. Space-local MOR is closely related to partial MOR. However, in partial MOR, the modes are also defined globally and couple the deformations of parts. Cutting the connection in partial MOR is therefore described as the next potential improvement.

Partial MOR, decouple ROMs: Partial MOR excludes sections or parts from the reduction process. The unreduced part is a HDM and the neighboring ROMs can be considered as boundary conditions. In this thesis, all reduced parts are combined into one ROM. This means the ROMs are coupled, and one reduced part cannot deform independently. In our crash box example, the deformation of the upper region of the crash box is coupled with the deformation of the lower parts. Cutting the connection by using space-local ROBs could yield better results.

Suitable AE architectures: The offline approximation error is one parameter that influences the online accuracy of the AE ROM. However, the achievable offline errors can be improved. The discrepancy between the training and validation errors is a hint of the AE's bad generalization capability. Since the crash box is a time-dependent problem, including the time history with convolutions could improve the accuracy. Recurrent AEs require a fixed time step. A fixed time step prevents an adjustment, so convolutions are prioritized. Using constraints on the weight matrices, other activation functions, or an oblique projection could improve the accuracy of the AE ROM while maintaining similar offline accuracy.

Fast higher-order derivatives of AE: The speedup strongly depends on the AE evaluation, including the Jacobian and Hessian matrix computation. Accelerating the evaluation could be achieved by sparsifying the AE, using Jacobian vector products instead of explicitly calculating the Jacobian, or using analytical results of the derivatives.

6.2 Conclusion

The main conclusions are summarized in this final section. More detailed conclusions are given at the end of each technical Chapter 2, 3, 4, and 5. In this thesis, we extended pMOR for explicit FEM applied to highly nonlinear systems in crash to parametric problems. First, a detailed discussion of pMOR considering residual minimization provided the foundation of this work. By using fast and efficient randomized algorithms for PCA and ECSW hyper-reduction (Bach [1]), we could consider a parametric crash box problem globally. A global PCA cannot provide a low-dimensional basis to capture the variance in the whole

data set. High dimensions are required to construct the ROM. Large dimensions negatively impact the achievable speedup. The two mechanisms considered in this thesis influenced by the large dimension are the projection step and ECSW hyper-reduction. The impact of hyper-reduction is the largest. Both, the offline and online phases are negatively affected. The optimization problem of the offline phase is enlarged, which yields a larger reduced mesh. Accordingly, more elements must be evaluated in the online phase, and the expected speedup is small.
Therefore, global ROMs do not yield the required speedups and are unsuitable for the intended purpose, such as robustness studies, uncertainty quantification, or optimization. Suppose pROMs based on a global linear dimensionality reduction are considered the least complex models, and pROMs based on a global nonlinear dimensionality reduction are the most complex models. In that case, pROMs based on locally linear ROBs are the next complexity level from global linear to global nonlinear ROMs. MOR based on lROBs divides the training samples into clusters and creates linear lROBs for each cluster. An efficient distance measure determines the nearest cluster, and the local ROM to be used is determined and changed if needed. Clustering is based on a distance measure. We compared the standard k-means clustering with spherical k-means clustering, which normalizes the data and accounts for the magnitude difference of snapshots during the beginning and toward the end of the simulation. While lROB based on spherical k-means clustering yields slightly more accurate results for ROMs without hyper-reduction, hyper-reduced models provide comparable accuracy. To summarize, the lROB method enables low-dimensional ROMs for parametric problems and outperforms the global linear method regarding dimensionality and accuracy. The prerequisites for a fast ROM are created, and hyper-reduction can be applied. Cluster-specific hyper-reduction is performed, which yields small reduced meshes with high accuracy. However, the cluster-specific reduced meshes are combined into a global reduced mesh with cluster-specific weights. History dependency in the constitutive model enforces a global mesh. This yields an evaluation of elements, which are weighted afterward with zero. Developing a method to avoid the evaluation of zero elements is a promising future research direction. In conclusion, parametric pROMs for crash problems were developed and achieved speedups up to 32%. Adjusting the critical time step and avoiding zero elements will result in speedups larger than 50%.
The development of parametric ROMs enables their application as boundary conditions. This use case is important as optimizing single parts embedded in a large model often requires the solution of the entire model. This is computationally infeasible, which is why simplified models replace the surrounding model with fixed boundary conditions. However, the underlying assumption of these boundary conditions is that the solution of the modified part in the model does not change the behavior of the boundary conditions itself. That is, the model's feedback to the surrounding model is neglected. We show that parametric ROMs based on lROB can serve as reduced boundary conditions. The crash box model is divided into reduced and unreduced parts, in which the unreduced part is modified. The geometry variations lead to solution changes of increasing severity. A geometry variation that primarily affects the modification's local area can be treated without a parametric ROM. This observation justifies the simplified approach, replacing the adjacent model with fixed boundary conditions. However, modifications impacting the global solution lead to a different buckling behavior, and parametric ROMs are required. We have shown that depending on the sensitivity of the problem, a more or less accurate ROM can be chosen. Especially in sensitive buckling problems, an accurate ROM using more dimensions must be selected to reproduce the buckling behavior correctly. Still, speedups of approx. 29% for an accurate ROM and 37% for a less accurate ROM can be expected related to the reduced parts. The speedup estimation includes evaluating zero elements and no adjustment of the critical time step.
Finally, we conducted a PoC for pMOR based on nonlinear dimensionality reduction. Finding the true latent dimension of a problem can further reduce the dimension of the ROM and positively influence hyper-reduction, resulting in even smaller reduced meshes. However, nonlinear dimensionality reduction increases complexity. No stability and accuracy estimates are available anymore. Nevertheless, we can present the first results using an AE for pMOR for explicit FEM in crash. The low dimension of the AE yields a reduction of 85% of all elements. However, the accuracy of the resulting AE ROM may be insufficient for industrial applications.

To summarize, this thesis presents the extension of pMOR to parametric problems for crash and impact. Methods reaching from global linear approximations to global nonlinear approximations are presented and investigated. While globally linear methods are still in research, locally linear MOR is suitable for parametric problems and yields fast and accurate ROMs. Nevertheless, as presented in Section 6.1, the potential for improvement promotes further research in this direction.

A Appendix

A.1 List of Supervised Study Projects

As part of this dissertation, the student works listed below were created at the Associate Professorship of Computational Solid Mechanics between 2020 and 2023 under the author's essential scientific, technical and content-related guidance. The author would like to thank all students for their commitment to supporting this academic work.

Name	**Thesis Title**	**Programme of Study**	**Year**
Evgeni Todorov	Restoring Force Surface Models for Finite Element Modeling in Crash Analysis	Computational Science and Engineering	2021
Maternus Herold	Projection-based Model Order Reduction using Autoencoders in nonlinear Finite Element Analysis	Informatics	2022
Syed Saadat Shakeel	Model Order Reduction in Crash using nonlinear Dimensional Reduction, eingeflossen in Kapitel 5	Computational Mechanics	2023

A.2 Hyper-reduction Offline Results

This section tabulates the offline results for the global-local hyper-reduced ROMs with dimension $k = 30$ and $k = 40$. The results for the ROM with $k = 30$ are tabulated in Tab. A.1 and Tab. A.2 and the results for the ROM with $k = 40$ are tabulated in Tab. A.3 and Tab. A.4.

Table A.1 Summary of global-local hyper-reduction for ROMs with 3 and 6 clusters and an approximation rank of $k = 30$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\lvert\tilde{E}\rvert$	α_i	τ_i	$\lvert\tilde{E}\rvert$	α_i	τ_i
3	0.01	1	916	0.92	0.454%	896	0.84	0.456%
	0.01	2	916	0.69	0.344%	896	0.95	0.607%
	0.01	3	916	0.89	0.484%	896	0.49	0.209%
	0.02	1	714	0.87	0.870%	705	0.80	0.966%
	0.02	2	714	0.68	0.895%	705	0.88	1.137%
	0.02	3	714	0.79	0.939%	705	0.50	0.495%
	0.03	1	596	0.84	1.568%	587	0.76	1.585%
	0.03	2	596	0.65	1.596%	587	0.85	1.709%
	0.03	3	596	0.77	1.397%	587	0.50	0.902%
6	0.01	1	1128	0.77	0.321%	1141	0.82	0.414%
	0.01	2	1128	0.75	0.227%	1141	0.80	0.281%
	0.01	3	1128	0.55	0.339%	1141	0.44	0.236%
	0.01	4	1128	0.69	0.409%	1141	0.72	0.413%
	0.01	5	1128	0.70	0.357%	1141	0.51	0.304%
	0.01	6	1128	0.54	0.240%	1141	0.39	0.190%
	0.02	1	892	0.72	0.606%	879	0.73	0.846%
	0.02	2	892	0.70	0.419%	879	0.75	0.624%
	0.02	3	892	0.53	0.645%	879	0.47	0.508%
	0.02	4	892	0.65	0.677%	879	0.70	0.716%
	0.02	5	892	0.63	0.672%	879	0.52	0.596%
	0.02	6	892	0.56	0.530%	879	0.40	0.519%
	0.03	1	762	0.70	0.909%	749	0.68	1.230%
	0.03	2	762	0.69	0.587%	749	0.71	1.061%
	0.03	3	762	0.51	0.945%	749	0.48	0.731%
	0.03	4	762	0.65	0.900%	749	0.68	1.035%
	0.03	5	762	0.58	0.968%	749	0.52	0.867%
	0.03	6	762	0.54	1.056%	749	0.41	0.877%

Table A.2 Summary of global-local hyper-reduction for ROMs with 9 clusters and an approximation rank of $k = 30$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\|\tilde{E}\|$	α_i	τ_i	$\|\tilde{E}\|$	α_i	τ_i
9	0.01	1	1190	0.60	0.256%	1257	0.35	0.277%
	0.01	2	1190	0.70	0.294%	1257	0.68	0.223%
	0.01	3	1190	0.47	0.260%	1257	0.69	0.248%
	0.01	4	1190	0.60	0.340%	1257	0.42	0.275%
	0.01	5	1190	0.49	0.528%	1257	0.58	0.433%
	0.01	6	1190	0.39	0.191%	1257	0.65	0.369%
	0.01	7	1190	0.51	0.217%	1257	0.41	0.329%
	0.01	8	1190	0.27	0.699%	1257	0.53	0.312%
	0.01	9	1190	0.56	0.341%	1257	0.35	0.183%
	0.02	1	933	0.59	0.412%	969	0.37	0.538%
	0.02	2	933	0.63	0.663%	969	0.66	0.398%
	0.02	3	933	0.49	0.648%	969	0.65	0.612%
	0.02	4	933	0.53	0.627%	969	0.42	0.490%
	0.02	5	933	0.48	0.778%	969	0.57	0.690%
	0.02	6	933	0.41	0.530%	969	0.58	0.685%
	0.02	7	933	0.48	0.444%	969	0.42	0.539%
	0.02	8	933	0.28	0.966%	969	0.54	0.519%
	0.02	9	933	0.56	0.551%	969	0.37	0.561%
	0.03	1	798	0.55	0.706%	826	0.39	0.810%
	0.03	2	798	0.61	0.947%	826	0.62	0.656%
	0.03	3	798	0.49	0.988%	826	0.61	0.939%
	0.03	4	798	0.45	0.993%	826	0.44	0.697%
	0.03	5	798	0.47	1.005%	826	0.55	0.919%
	0.03	6	798	0.40	0.919%	826	0.52	1.070%
	0.03	7	798	0.44	0.870%	826	0.42	0.805%
	0.03	8	798	0.30	1.135%	826	0.54	0.713%
	0.03	9	798	0.55	0.751%	826	0.36	1.003%

Table A.3 Summary of global-local hyper-reduction for ROMs with 3 and 6 clusters and an approximation rank of $k = 40$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\lvert\tilde{\mathsf{E}}\rvert$	α_i	τ_i	$\lvert\tilde{\mathsf{E}}\rvert$	α_i	τ_i
3	0.01	1	970	0.92	0.458%	949	0.83	0.452%
	0.01	2	970	0.71	0.349%	949	0.95	0.575%
	0.01	3	970	0.91	0.486%	949	0.52	0.193%
	0.02	1	769	0.86	0.880%	752	0.81	0.952%
	0.02	2	769	0.71	0.896%	752	0.87	1.095%
	0.02	3	769	0.82	0.934%	752	0.51	0.534%
	0.03	1	652	0.85	1.363%	647	0.79	1.488%
	0.03	2	652	0.69	1.500%	647	0.87	1.643%
	0.03	3	652	0.76	1.429%	647	0.52	0.936%
6	0.01	1	1196	0.78	0.297%	1217	0.85	0.398%
	0.01	2	1196	0.76	0.249%	1217	0.79	0.307%
	0.01	3	1196	0.57	0.285%	1217	0.47	0.226%
	0.01	4	1196	0.70	0.397%	1217	0.73	0.414%
	0.01	5	1196	0.72	0.384%	1217	0.53	0.284%
	0.01	6	1196	0.56	0.277%	1217	0.40	0.178%
	0.02	1	962	0.74	0.633%	977	0.71	0.881%
	0.02	2	962	0.73	0.439%	977	0.75	0.559%
	0.02	3	962	0.56	0.568%	977	0.48	0.477%
	0.02	4	962	0.67	0.666%	977	0.70	0.699%
	0.02	5	962	0.66	0.714%	977	0.54	0.503%
	0.02	6	962	0.56	0.630%	977	0.41	0.487%
	0.03	1	817	0.70	0.973%	841	0.64	1.387%
	0.03	2	817	0.70	0.687%	841	0.71	0.924%
	0.03	3	817	0.55	0.865%	841	0.50	0.629%
	0.03	4	817	0.67	0.929%	841	0.68	1.003%
	0.03	5	817	0.61	1.073%	841	0.53	0.780%
	0.03	6	817	0.55	1.065%	841	0.41	0.755%

Table A.4 Summary of global-local hyper-reduction for ROMs with 9 clusters and an approximation rank of $k = 40$.

# cluster	τ	cluster	k-means			spherical k-means		
		i	$\|\tilde{E}\|$	α_i	τ_i	$\|\tilde{E}\|$	α_i	τ_i
9	0.01	1	1332	0.63	0.219%	1338	0.36	0.248%
	0.01	2	1332	0.70	0.242%	1338	0.72	0.192%
	0.01	3	1332	0.49	0.212%	1338	0.71	0.256%
	0.01	4	1332	0.67	0.292%	1338	0.45	0.246%
	0.01	5	1332	0.59	0.411%	1338	0.62	0.399%
	0.01	6	1332	0.40	0.176%	1338	0.71	0.309%
	0.01	7	1332	0.57	0.208%	1338	0.44	0.305%
	0.01	8	1332	0.47	0.295%	1338	0.55	0.293%
	0.01	9	1332	0.58	0.260%	1338	0.36	0.168%
	0.02	1	1073	0.61	0.382%	1080	0.39	0.408%
	0.02	2	1073	0.65	0.582%	1080	0.68	0.378%
	0.02	3	1073	0.49	0.522%	1080	0.67	0.513%
	0.02	4	1073	0.58	0.564%	1080	0.46	0.405%
	0.02	5	1073	0.56	0.644%	1080	0.60	0.609%
	0.02	6	1073	0.41	0.477%	1080	0.61	0.661%
	0.02	7	1073	0.53	0.408%	1080	0.44	0.497%
	0.02	8	1073	0.47	0.565%	1080	0.56	0.438%
	0.02	9	1073	0.56	0.465%	1080	0.39	0.338%
	0.03	1	933	0.60	0.507%	923	0.41	0.628%
	0.03	2	933	0.62	0.852%	923	0.66	0.622%
	0.03	3	933	0.50	0.741%	923	0.64	0.920%
	0.03	4	933	0.55	0.766%	923	0.48	0.576%
	0.03	5	933	0.54	0.889%	923	0.59	0.842%
	0.03	6	933	0.41	0.829%	923	0.56	1.025%
	0.03	7	933	0.52	0.542%	923	0.45	0.656%
	0.03	8	933	0.45	0.709%	923	0.56	0.663%
	0.03	9	933	0.57	0.602%	923	0.37	0.930%

A.3 Local ROB: Hyper-reduction Online Results - Parameter Study

This section includes further evaluations of the parameter study for the hyper-reduced local ROMs presented in subsection 3.3.6. The figures differ from those in Subsection 3.3.6 in that they use the same y-axis scaling.

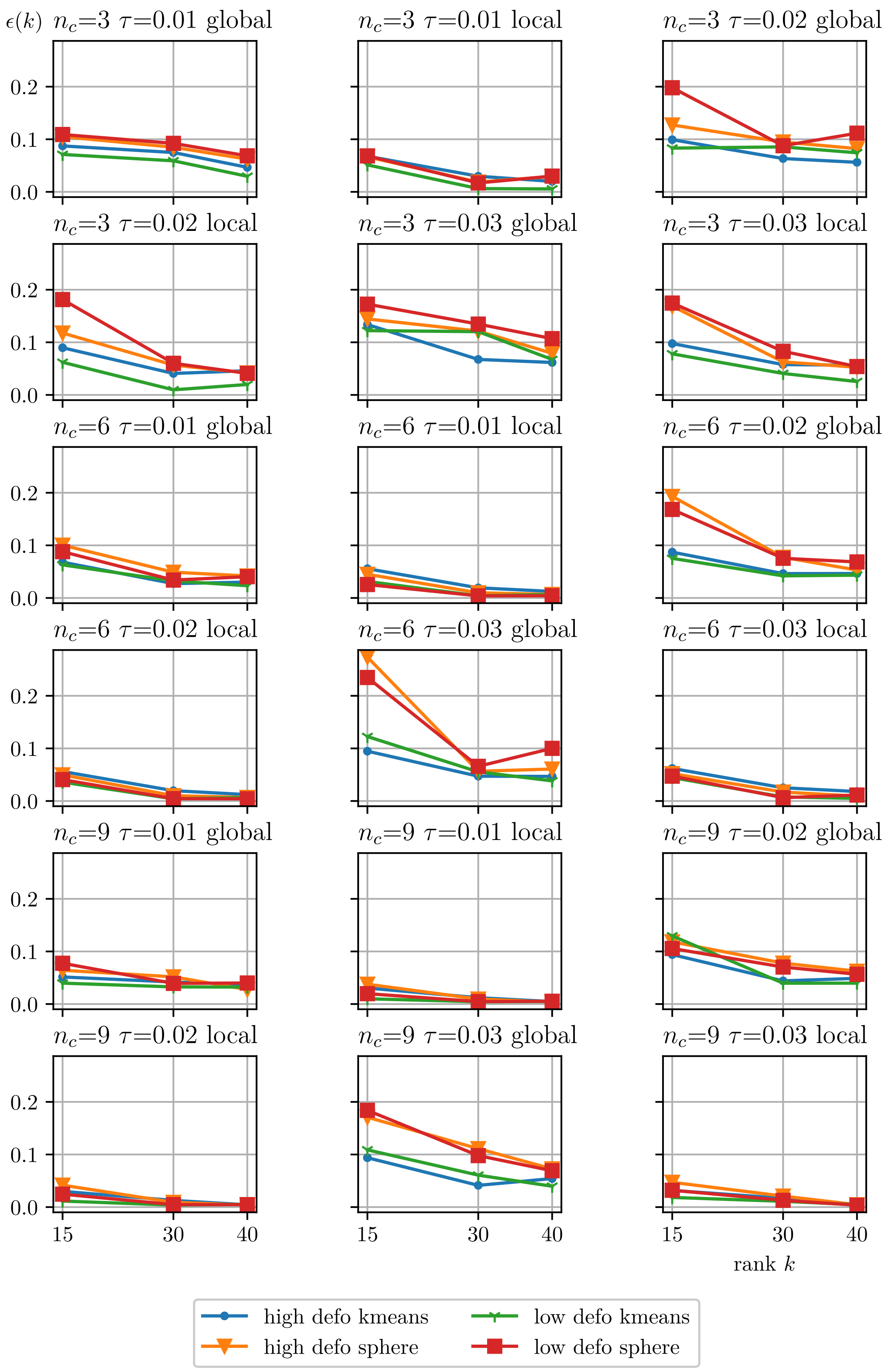


Figure A.1 Hyper-reduction comparison between the k-means ROM and spherical k-means ROM for both test cases.

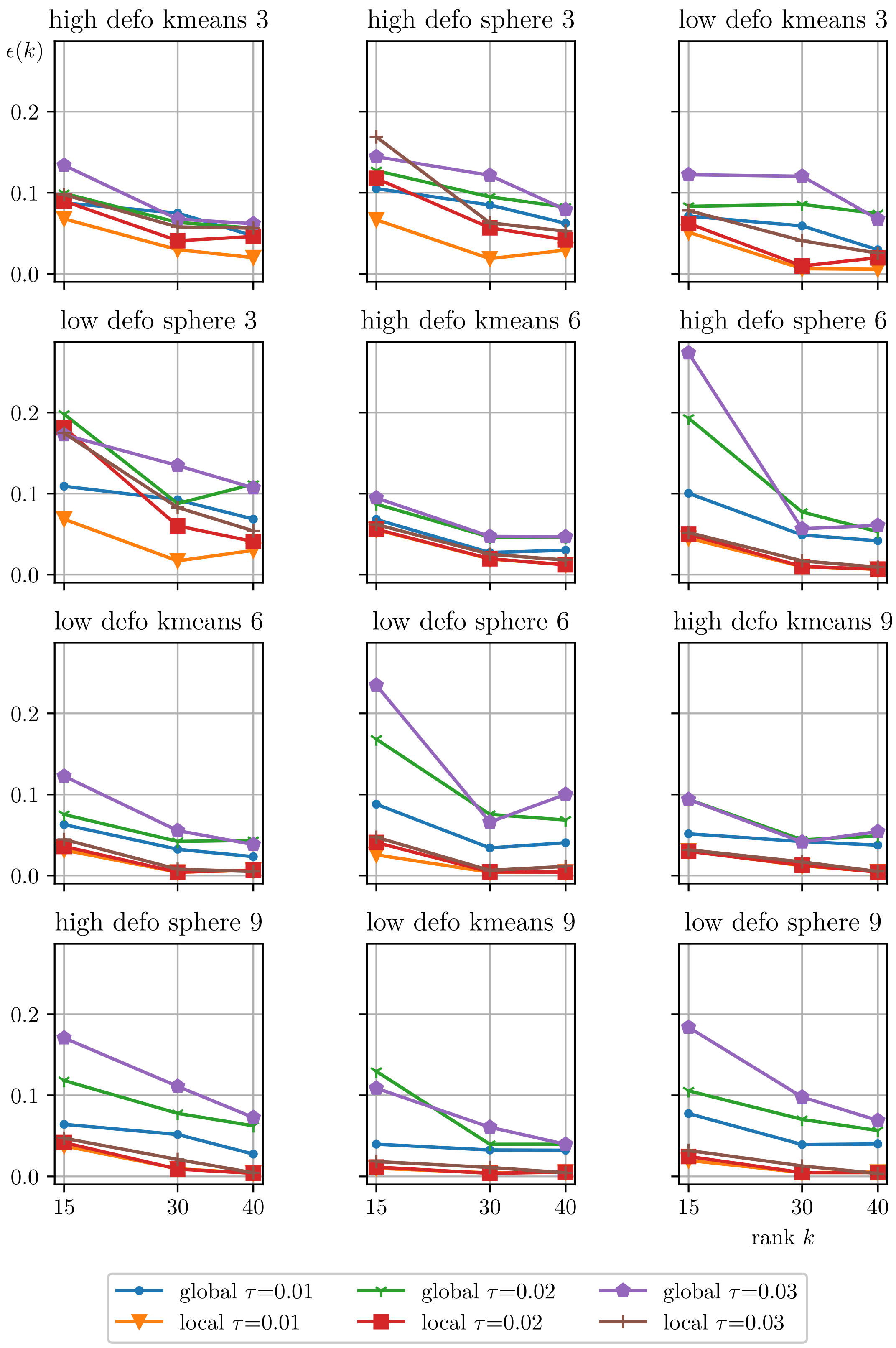


Figure A.2 Hyper-reduction comparison of the global-global approach versus the global-local approach.

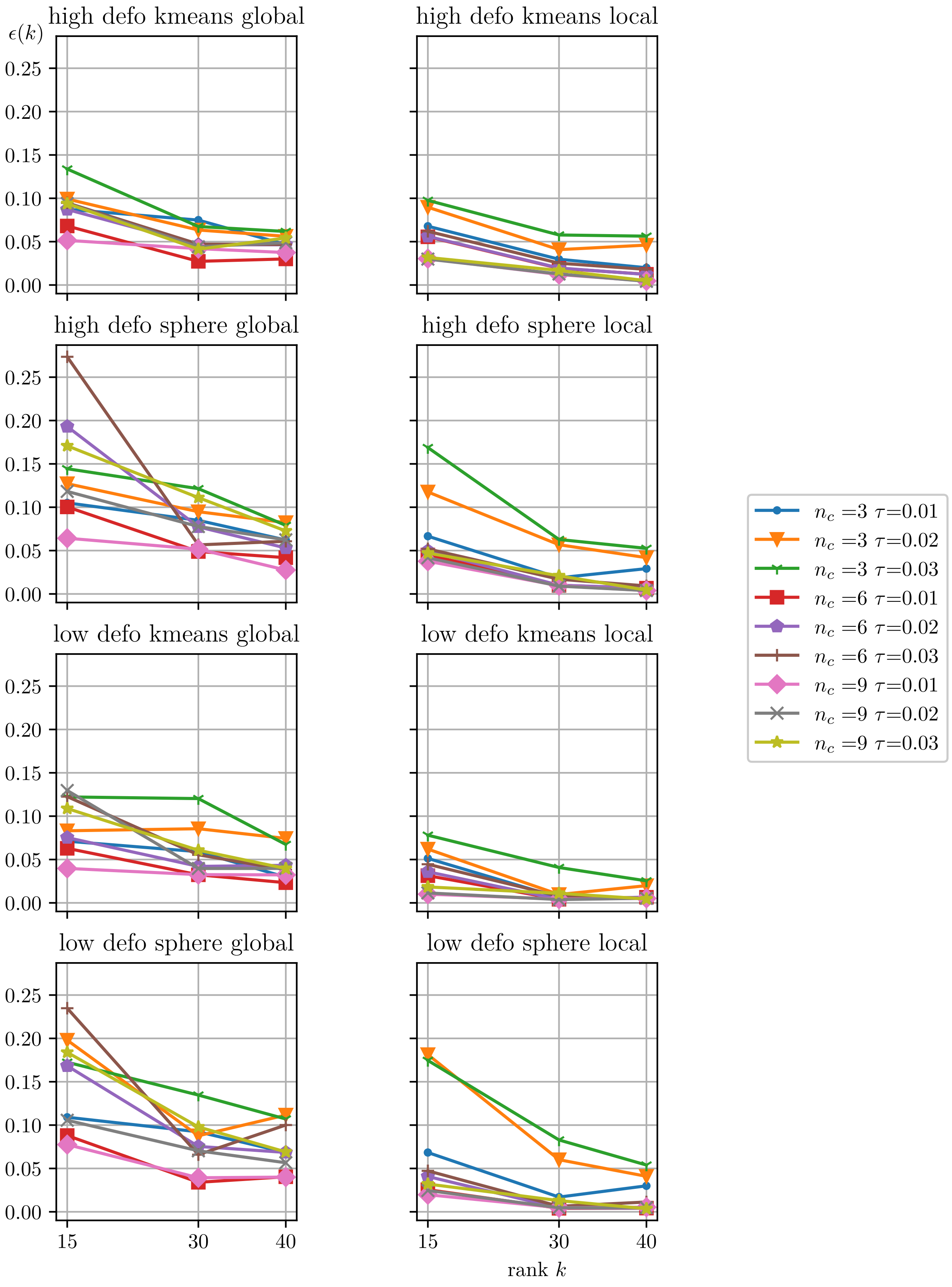


Figure A.3 Comparison of all cluster numbers and tolerances τ for one hyper-reduced ROM for one test case in each subplot.

A.4 Fortran C Interface

An exemplary implementation of the Fortran C interface, as illustrated in Fig. 5.8, is presented for the interested reader. Using the example of the predict function, code snippets of the interface are shown. The predict function in Fortran is the entry point of the interface and is called in each solver iteration. We begin with the declaration of the predict function in Fortran, as seen in Listing A.1. As the subroutine definition is not inside a program, another subroutine, function, or module, the subroutine is external, and the subroutine declaration is wrapped in an interface. The Iso-C-binding module is used to ensure interoperability with the C programming language. The bind attribute in line 3 ensures that the subroutine is interoperable with C. Also, the usage of the types c_ptr and c_int serve the same purpose. Fortran passes by default references, which can be considered pointers in C. Since c_ptr is already a pointer type, the value attribute is passed, forcing Fortran to pass the variable's value. The value attribute is also used to pass the dimensions of the arrays by value to the function definition in C, which is shown in Listing A.4.

```fortran
1  interface predict
2      subroutine predict(state,jac_mat,hes_mat,dim1,dim2)
3                                                 bind(C,name="predict")
4          use, intrinsic :: iso_c_binding, only: c_ptr, c_int
5          implicit none
6          type(c_ptr), value :: state
7          type(c_ptr), value :: jac_mat
8          type(c_ptr), value :: hes_mat
9          !DIMENSIONS
10         integer(c_int), value :: dim1
11         integer(c_int), value :: dim2
12     end subroutine predict
13 end interface predict
```

Listing A.1 Declaration of a generic predict function

Before we discuss the implementation of the predict function, which will use the Python C-API, we need to initialize the interpreter first. Since the declaration in Fortran of a function implemented in C has already been shown, we will forego it and show the C code directly. The init_python function is shown in Listing A.2. Assume the python file module.py is saved in the directory "folder." The Python file contains the pure-python function Pypredict. In Python, every variable, every function, and every module is a Python object (PyObject) defined in Python.h. The variables pModule and pFunc will be used in other functions, which is why they are defined globally. In the function init_python in Listing A.2, the python C-API functions can be used after the Python interpreter is initialized. For instance, strings can be directly executed. Finally, the function is identified in the module and assigned to pFunc. The reference count of unused variables is decremented since Python has a garbage collector, which will free the memory allocated by unused variables.

```c
#include <stdio.h>
#include <Python.h>

//Python objects of module and function
//Defined in global scope
PyObject *pModule = NULL;
PyObject *pFunc = NULL;

//Strings of module and function
char module[] = "folder.module";
char func[] = "Pypredict";

```

```c
void init_python(){

    //For decoded string
    PyObject *pName;

    //Initialize Python Interpreter
    Py_Initialize();

    //Now, the C-API can be used
    //Execute String
    PyRun_SimpleString("import sys");

    //Get the module
    pName = PyUnicode_DecodeFSDefault(module);
    pModule = PyImport_Import(pName);


    //Check if module was found
    if(pModule != NULL){
        //Get function of module
        pFunc = PyObject_GetAttrString(pModule, func);
    }

    //Decrease reference count
    Py_DECREF(pName);
}
```

Listing A.2 Initialization of the python Interpreter.

Once the interpreter is set up, the prediction of the AE can be obtained using Python. We must utilize the C-API and tell the interpreter to call the Python function stored in pFunc. However, before we can call the Python function, we need to wrap the data passed by Fortran in a suitable Python object. In this work, we decided to use the Numpy [152] C-API to directly create Numpy arrays, which are passed to the Python function. The benefit is that the memory address is passed directly, and there is no need to copy the data. However, this is beyond the scope of this report. Therefore, we quickly illustrate the idea by creating a Python list based on a one-dimensional C array in Listing A.3. Even a float value is an object inside Python. Therefore, the C double must be converted to a PyFloat before inserting it into the list.

```c
// Function to convert a C array to a Python list
PyObject* array_to_list(double* array, int dim1) {
    PyObject* pyList = PyList_New(dim1);

    for (int i = 0; i < dim1; ++i) {
        //Also a simple float is a Python object
        PyObject* pyValue = PyFloat_FromDouble(array[i]);
        //Insert the float at position i
        PyList_SetItem(pyList, i, pyValue);
    }

    return pyList;
}
```

Listing A.3 Function which transfers data from a one-dimensional array to a Python list.

Finally, the C-API is used to call the function, which will execute the function defined in pure Python. This is the function which will use Keras to query the AE. The call to the Python function happens in line 13 of Listing A.4. In the conversion of the raw data to the Python list, we used that dim1≤dim2, and before writing the results of the Python function back to state, the first dim1 values of state can be used to pass the latent variable state to the decoder.

```c
1  void predict(void *state, void *jac, void *hes, int dim1, int dim2){
2  
3      PyObject *pArgs, *pReturnValue;
4      PyObject *pstate, *pjac, *phes;
5  
6      pstate = array_to_list((double*) state, dim1);
7  
8      //Create tuple of arguments which are passed to pFunc
9      pArgs = PyTuple_New(1);
10     PyTuple_SetItem(pArgs, 0, pstate);
11 
12     //Call the Python function
13     pReturnValue = PyObject_CallObject(pFunc, pArgs);
14 
15     Py_DECREF(pArgs);
16 
17     //Obtain the return values
18     pstate = PyTuple_GetItem(pReturnValue,0);
19     pjac = PyTuple_GetItem(pReturnValue,1);
20     phes = PyTuple_GetItem(pReturnValue,2);
21 
22     //Copy data from PyList to double array
23     for (int i = 0; i < dim2; ++i) {
24         PyObject* listItem = PyList_GetItem(pstate, i);
25         state[i] = PyFloat_AsDouble(listItem);
26     }
27 
28     /*
29     Do the same for Jacobian and Hessian Matrix here...
30     */
31 
32     Py_DECREF(pReturnValue);
33     return;
34 }
```

Listing A.4 Implementation of the predict function declared in Fortran.

The function Pypredict is now accessible by Fortran, and an exemplary implementation is shown in Listing A.5. The one-dimensional state is converted to a Python list to illustrate the conversion back to a C array in Listing A.4. The two- and three-dimensional arrays for the Jacobian and Hessian matrices are left as Numpy arrays.

```python
def Pypredict(x):
    #convert state x to suitable shape
    x =np.array(x)
    x = x.reshape((1, −1))
    x = tf.convert_to_tensor(x, dtype=tf.dtypes.float64)

    #Query the AE model
    state, J, H = Decoder(x)

    #Obtain numeric values
    state = list(state.numpy())

    J = J.numpy()
    H = H.numpy()

    return state, J, H
```

Listing A.5 Pure-Python implementation of the function Pypredict.

The function which finally queries the AE uses the automatic differentiation(AD) framework of Tensorflow, as can be seen in Listing A.6.

```python
@tf.function
def Decoder(x):
    with tf.GradientTape() as t2:
        t2.watch(x)
        with tf.GradientTape() as t1:
            t1.watch(x)
            #Prediction of the AE
            y = model(x)
        J = t1.batch_jacobian(y, x)
    H = t2.batch_jacobian(J, x)

    return y[0,:], J[0,:,:], H[0,:,:,:]
```

Listing A.6 Using AD to obtain the Jacobian and Hessian matrix.

Finally, we mention important things that were not sufficiently discussed in the short illustration of the interface. Care must be taken for the data types of the numeric data. If double values are passed to Python, they must be declared as such. Mixing floats and double values leads to the wrong interpretation of the data and may cause segmentation faults. Also, Fortran uses a different memory layout than C. This layout becomes important when working with higher-dimensional arrays.
We touched slightly on memory management by decreasing the reference counts of some objects. When using the Python C-API, it is important to correctly increase or decrease the reference counts of objects. Otherwise, memory leaks or segmentation faults can occur. It is also important to close the interpreter and release all memory at the end of the application. Python itself is not thread-safe, which is why Python has the global interpreter lock (GIL). The GIL prevents multiple threads from executing Python code at the same time. Care must be taken when interfacing with a multi-threaded application such as LS-Dyna. The example shown completely dispenses with checks, but they are recommended to avoid application crashes and help in the error search. For instance, checks could be introduced if the module and function are found and correctly loaded. Also, it checks if the function call was successful or if the returned Python type is the expected one.
Since we include the Python header Python.h, we also need to provide it at compile time. The Python configure script can be used to find the required includes:

```
python3-config --prefix|--exec-prefix|--includes|--libs|--cflags|--ldflags
               |--extension-suffix|--help|--abiflags|--configdir|--embed
```

Listing A.7 Python configure script.

During linking, it is necessary to link against the Python library, which is achieved by passing the option:

```
-lpython3.8
```

to the linker, where 3.8 denotes the used Python version.

Bibliography

[1] C. Bach. “Data-driven model order reduction for nonlinear crash and impact simulations”. Dissertation. München: Technische Universität München, 2019. ISBN: 978-3-8440-7297-6.

[2] C. Czech, M. Lesjak, C. Bach, and F. Duddeck. “Data-driven models for crashworthiness optimisation: intrusive and non-intrusive model order reduction techniques”. In: *Structural and Multidisciplinary Optimization* 65.7 (2022). DOI: 10.1007/s00158-022-03282-1.

[3] P. Benner, S. Gugercin, and K. Willcox. “A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems”. In: *SIAM Review* 57.4 (2015), pp. 483–531. DOI: 10.1137/130932715.

[4] M. Eldred and D. Dunlavy. “Formulations for Surrogate-Based Optimization with Data Fit, Multifidelity, and Reduced-Order Models”. In: *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. DOI: 10.2514/6.2006-7117.

[5] J. Fehr, P. Holzwarth, and P. Eberhard. “Interface and model reduction for efficient explicit simulations- a case study with nonlinear vehicle crash models”. In: *Mathematical and Computer Modelling of Dynamical Systems* 22.4 (2016), pp. 380–396. DOI: 10.1080/13873954.2016.1198385.

[6] D. Grunert and J. Fehr. “Identification of nonlinear behavior with clustering techniques in car crash simulations for better model reduction”. In: *Advanced Modeling and Simulation in Engineering Sciences* 3.1 (2016), pp. 1–19. DOI: 10.1186/s40323-016-0072-x.

[7] M. F. Hussain, R. R. Barton, and S. B. Joshi. “Metamodeling: Radial basis functions, versus polynomials”. In: *European Journal of Operational Research* 138.1 (2002), pp. 142–154. DOI: 10.1016/S0377-2217(01)00076-5.

[8] M. D. Spiridonakos and E. N. Chatzi. “Metamodeling of dynamic nonlinear structural systems through polynomial chaos NARX models”. In: *Computers & Structures* 157 (2015), pp. 99–113. DOI: 10.1016/j.compstruc.2015.05.002.

[9] M. Guo and J. S. Hesthaven. “Reduced order modeling for nonlinear structural analysis using Gaussian process regression”. In: *Computer Methods in Applied Mechanics and Engineering* 341 (2018), pp. 807–826. DOI: 10.1016/j.cma.2018.07.017.

[10] M. Guo and J. S. Hesthaven. “Data-driven reduced order modeling for time-dependent problems”. In: *Computer Methods in Applied Mechanics and Engineering* 345 (2019), pp. 75–99. DOI: 10.1016/j.cma.2018.10.029.

[11] M. Kast, M. Guo, and J. S. Hesthaven. “A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems”. In: *Computer Methods in Applied Mechanics and Engineering* 364 (2020), p. 112947. DOI: 10.1016/j.cma.2020.112947.

[12] J. P. C. Kleijnen. “Kriging metamodeling in simulation: A review”. In: *European Journal of Operational Research* 192.3 (2009), pp. 707–716. DOI: 10.1016/j.ejor.2007.10.013.

[13] J. S. Hesthaven and S. Ubbiali. “Non-intrusive reduced order modeling of nonlinear problems using neural networks”. In: *Journal of Computational Physics* 363 (2018), pp. 55–78. DOI: 10.1016/j.jcp.2018.02.037.

[14] J. Kneifl, D. Grunert, and J. Fehr. “A nonintrusive nonlinear model reduction method for structural dynamical problems based on machine learning”. In: *International Journal for Numerical Methods in Engineering* 122.17 (2021), pp. 4774–4786. DOI: 10.1002/nme.6712.

[15] J. Kneifl and J. Fehr. “Machine Learning Algorithms for Learning Nonlinear Terms of Reduced Mechanical Models in Explicit Structural Dynamics”. In: *PAMM* 20.S1 (2021), e202000353. DOI: 10.1002/pamm.202000353.

[16] J. Kneifl, J. Hay, and J. Fehr. “Real-time Human Response Prediction Using a Non-intrusive Data-driven Model Reduction Scheme”. In: *IFAC-PapersOnLine* 55.20 (2022), pp. 283–288. DOI: 10.1016/j.ifacol.2022.09.109.

[17] A. Fuchs, Y. Heider, K. Wang, W. Sun, and M. Kaliske. “DNN2: A hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions”. In: *Computers & Structures* 249 (2021), p. 106505. DOI: 10.1016/j.compstruc.2021.106505.

[18] S. F. Masri, A. G. Chassiakos, and T. K. Caughey. “Identification of Nonlinear Dynamic Systems Using Neural Networks”. In: *Journal of Applied Mechanics* 60.1 (1993), pp. 123–133. DOI: 10.1115/1.2900734.

[19] J.-S. Pei and E. C. Mai. “Constructing Multilayer Feedforward Neural Networks to Approximate Nonlinear Functions in Engineering Mechanics Applications”. In: *Journal of Applied Mechanics* 75.6 (2008), p. 061002. DOI: 10.1115/1.2957600.

[20] R. Zhang, Y. Liu, and H. Sun. “Physics-informed multi-LSTM networks for metamodeling of nonlinear structures”. In: *Computer Methods in Applied Mechanics and Engineering* 369 (2020), p. 113226. DOI: 10.1016/j.cma.2020.113226.

[21] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113.

[22] B. Peherstorfer and K. Willcox. “Dynamic data-driven model reduction: adapting reduced models from incomplete data”. In: *Advanced Modeling and Simulation in Engineering Sciences* 3.1 (2016), pp. 1–22. DOI: 10.1186/s40323-016-0064-x.

[23] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. “Lift & Learn: Physics-informed machine learning for large-scale nonlinear dynamical systems”. In: *Physica D: Nonlinear Phenomena* 406 (2020), p. 132401. DOI: 10.1016/j.physd.2020.132401.

[24] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451. DOI: 10.1073/pnas.1906995116.

[25] N. Sawant, B. Kramer, and B. Peherstorfer. “Physics-informed regularization and structure preservation for learning stable reduced models from data with operator inference”. In: *Computer Methods in Applied Mechanics and Engineering* 404 (2023), p. 115836. DOI: 10.1016/j.cma.2022.115836.

[26] A. A. Kaptanoglu, J. L. Callaham, A. Aravkin, C. J. Hansen, and S. L. Brunton. “Promoting global stability in data-driven models of quadratic nonlinear dynamics”. In: *Phys. Rev. Fluids* 6.9 (2021), p. 094401. DOI: 10.1103/PhysRevFluids.6.094401.

[27] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440. DOI: 10.1038/s42254-021-00314-5.

[28] M. Raissi, P. Perdikaris, and G. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707. DOI: https://doi.org/10.1016/j.jcp.2018.10.045.

[29] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature Machine Intelligence* 3.3 (2021), pp. 218–229. DOI: 10.1038/s42256-021-00302-5.

[30] J. Mann and N. J. Kutz. “Dynamic mode decomposition for financial trading strategies”. In: *Quantitative Finance* 16.11 (2016), pp. 1643–1655. DOI: 10.1080/14697688.2016.1170194.

[31] P. J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28. DOI: 10.1017/S0022112010001217.

[32] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley. “Modal Analysis of Fluid Flows: An Overview”. In: *AIAA Journal* 55.12 (2017), pp. 4013–4041. DOI: 10.2514/1.J056060.

[33] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition”. In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346. DOI: 10.1007/s00332-015-9258-5.

[34] Z. Lai and S. Nagarajaiah. “Sparse structural system identification method for nonlinear dynamic systems with hysteresis/inelastic behavior”. In: *Mechanical Systems and Signal Processing* 117 (2019), pp. 813–842. DOI: 10.1016/j.ymssp.2018.08.033.

[35] M. Stender, S. Oberst, and N. Hoffmann. “Recovery of Differential Equations from Impulse Response Time Series Data for Model Identification and Feature Extraction”. In: *Vibration* 2.1 (2019), pp. 25–46. DOI: 10.3390/vibration2010002.

[36] G. Thiele, A. Fey, D. Sommer, and J. Krüger. “System identification of a hysteresis-controlled pump system using SINDy”. In: *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*. 2020, pp. 457–464. DOI: 10.1109/ICSTCC50638.2020.9259776.

[37] G. F. Sirca and H. Adeli. “System identification in structural engineering”. In: *Scientia Iranica* 19.6 (2012), pp. 1355–1364. DOI: 10.1016/j.scient.2012.09.002.

[38] R. Ceravolo, S. Erlicher, and L. Z. Fragonara. “Comparison of restoring force models for the identification of structures with hysteresis and degradation”. In: *Journal of Sound and Vibration* 332.26 (2013), pp. 6982–6999. DOI: 10.1016/j.jsv.2013.08.019.

[39] E. Kim and M. Cho. “Equivalent model construction for a non-linear dynamic system based on an element-wise stiffness evaluation procedure and reduced analysis of the equivalent system”. In: *Computational Mechanics* 60.5 (2017), pp. 709–724. DOI: 10.1007/s00466-017-1435-y.

[40] R. Perez, X. Q. Wang, and M. P. Mignolet. “Nonintrusive Structural Dynamic Reduced Order Modeling for Large Deformations: Enhancements for Complex Structures”. In: *Journal of Computational and Nonlinear Dynamics* 9.3 (2014), p. 031008. DOI: 10.1115/1.4026155.

[41] J. Lee, J. Lee, H. Cho, E. Kim, and M. Cho. “Reduced-order modeling of nonlinear structural dynamical systems via element-wise stiffness evaluation procedure combined with hyper-reduction”. In: *Computational Mechanics* 67.2 (2021), pp. 523–540. DOI: 10.1007/s00466-020-01946-7.

[42] P. M. Slaats, J. de Jongh, and A. Sauren. “Model reduction tools for nonlinear structural dynamics”. In: *Computers & Structures* 54.6 (1995), pp. 1155–1171. DOI: 10.1016/0045-7949(94)00389-K.

[43] S. R. Idelsohn and A. Cardona. “A reduction method for nonlinear structural dynamic analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 49.3 (1985), pp. 253–279. DOI: 10.1016/0045-7825(85)90125-2.

[44] J. Barbič and D. L. James. “Real-time subspace integration for St. Venant-Kirchhoff deformable models”. In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 982–990. DOI: 10.1145/1073204.1073300.

[45] P. Tiso, E. Jansen, and M. Abdalla. “Reduction method for finite element nonlinear dynamic analysis of shells”. In: *AIAA Journal* 49.10 (2011), pp. 2295–2304. DOI: 10.2514/1.J051003.

[46] P. Tiso. “Optimal second order reduction basis selection for nonlinear transient analysis”. In: *Modal Analysis Topics, Volume 3*. Springer, 2011, pp. 27–39. DOI: 10.1007/978-1-4419-9299-4_3.

[47] O. Weeger, U. Wever, and B. Simeon. "On the use of modal derivatives for nonlinear model order reduction". In: *International Journal for Numerical Methods in Engineering* 108.13 (2016), pp. 1579–1602. DOI: 10.1002/nme.5267.

[48] A. K. Noor, C. M. Andersen, and J. M. Peters. "Reduced basis technique for nonlinear vibration analysis of composite panels". In: *Computer Methods in Applied Mechanics and Engineering* 103.1-2 (1993), pp. 175–186. DOI: 10.1016/0045-7825(93)90045-Y.

[49] K. Hildebrandt, C. Schulz, C. v. Tycowicz, and K. Polthier. "Interactive surface modeling using modal analysis". In: *ACM Transactions on Graphics (TOG)* 30.5 (2011), pp. 1–11. DOI: 10.1145/2019627.2019638.

[50] O. A. Bauchau and D. Guernsey. "On the choice of appropriate bases for nonlinear dynamic modal analysis". In: *Journal of the American Helicopter Society* 38.4 (1993), pp. 28–36. DOI: 10.4050/JAHS.38.28.

[51] D. A. Tortorelli and P. Michaleris. "Design sensitivity analysis: Overview and review". In: *Inverse Problems in Engineering* 1.1 (1994), pp. 71–105. DOI: 10.1080/174159794088027573.

[52] F. van Keulen, R. T. Haftka, and N. H. Kim. "Review of options for structural design sensitivity analysis. Part 1: Linear systems". In: *Computer Methods in Applied Mechanics and Engineering* 194.30 (2005), pp. 3213–3243. DOI: 10.1016/j.cma.2005.02.002.

[53] R. M. Hintz. "Analytical Methods in Component Modal Synthesis". In: *AIAA Journal* 13.8 (1975), pp. 1007–1016. DOI: 10.2514/3.60498.

[54] N. M. M. Maia and Montalvão e Silva, Júlio Martins, eds. *Theoretical and Experimental Modal Analysis*. Taunton, Somerset, England: Research Studies Press, 1997. ISBN: 0863802087.

[55] R. R. Craig and M. C. C. Bampton. "Coupling of substructures for dynamic analyses". In: *AIAA Journal* 6.7 (1968), pp. 1313–1319. DOI: 10.2514/3.4741.

[56] S. Rubin. "Improved Component-Mode Representation for Structural Dynamic Analysis". In: *AIAA Journal* 13.8 (1975), pp. 995–1006. DOI: 10.2514/3.60497.

[57] J. L. Lumley. "The structure of inhomogeneous turbulent flows". In: *Atmospheric Turbulence and Radio Wave Propagation* (1967). URL: https://ci.nii.ac.jp/naid/10012381873/.

[58] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24.6 (1933), pp. 417–441. DOI: 10.1037/h0071325.

[59] J. J. Gerbrands. "On the relationships between SVD, KLT and PCA". In: *Pattern Recognition* 14.1 (1981), pp. 375–381. DOI: 10.1016/0031-3203(81)90082-0.

[60] G. W. Stewart. "On the Early History of the Singular Value Decomposition". In: *SIAM Review* 35.4 (1993), pp. 551–566. DOI: 10.1137/1035134.

[61] L. Sirovich. "Turbulence and the dynamics of coherent structures. III. Dynamics and scaling". In: *Quarterly of Applied Mathematics* 45.3 (1987), pp. 583–590. DOI: 10.1090/qam/910464.

[62] M. Udell and A. Townsend. "Why Are Big Data Matrices Approximately Low Rank?" In: *SIAM Journal on Mathematics of Data Science* 1 (2019), pp. 144–160. DOI: 10.1137/18M1183480.

[63] J. B. Rutzmoser, F. M. Gruber, and D. J. Rixen. "A comparison on model order reduction techniques for geometrically nonlinear systems based on a modal derivative approach using subspace angles". In: *Proceedings of the 11th International Conference on Engineering Vibration*. 2015.

[64] R. J. Kuether, M. R. Brake, and M. S. Allen. "Evaluating Convergence of Reduced Order Models Using Nonlinear Normal Modes". In: *Model Validation and Uncertainty Quantification, Volume 3*. Springer, Cham, 2014, pp. 287–300. DOI: 10.1007/978-3-319-04552-8_28.

[65] R. J. Kuether and M. S. Allen. "Validation of Nonlinear Reduced Order Models with Time Integration Targeted at Nonlinear Normal Modes". In: *Nonlinear Dynamics, Volume 1*. Springer, Cham, 2016, pp. 363–375. DOI: 10.1007/978-3-319-15221-9_33.

[66] C. Eckart and G. Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/BF02288367.

[67] B. Moore. “Principal component analysis in linear systems: Controllability, observability, and model reduction”. In: *IEEE Transactions on Automatic Control* 26.1 (1981), pp. 17–32. DOI: 10.1109/TAC.1981.1102568.

[68] C. W. Rowley. “Model Reduction for Fluids, using Balanced Proper Orthogonal Decomposition”. In: *International Journal of Bifurcation and Chaos* 15.03 (2005), pp. 997–1013. DOI: 10.1142/S0218127405012429.

[69] P. Benner and P. Goyal. “Balanced truncation model order reduction for quadratic-bilinear control systems”. In: *arXiv preprint arXiv:1705.00160* (2017). DOI: 10.48550/arXiv.1705.00160.

[70] J. Scherpen. “Balancing for nonlinear systems”. In: *Systems & Control Letters* 21.2 (1993), pp. 143–153. DOI: 10.1016/0167-6911(93)90117-O.

[71] S. Lall, J. E. Marsden, and S. Glavaški. “A subspace approach to balanced truncation for model reduction of nonlinear control systems”. In: *International Journal of Robust and Nonlinear Control* 12.6 (2002), pp. 519–535. DOI: 10.1002/rnc.657.

[72] E. I. Verriest and W. S. Gray. “Nonlinear balanced realizations”. In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. Vol. 2. 2004, 1164–1169 Vol.2. DOI: 10.1109/CDC.2004.1430199.

[73] S. E. Otto, A. Padovan, and C. W. Rowley. *Model Reduction for Nonlinear Systems by Balanced Truncation of State and Gradient Covariance*. 2023. DOI: 10.48550/arxiv.2207.14387.

[74] P. Phalippou, S. Bouabdallah, P. Breitkopf, P. Villon, and M. Zarroug. “’On-the-fly’ snapshots selection for Proper Orthogonal Decomposition with application to nonlinear dynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 367 (2020), p. 113120. DOI: 10.1016/j.cma.2020.113120.

[75] D. Chelidze and W. Zhou. “Smooth orthogonal decomposition-based vibration mode identification”. In: *Journal of Sound and Vibration* 292.3-5 (2006), pp. 461–473. DOI: 10.1016/j.jsv.2005.08.006.

[76] E. A. Christensen, M. Brøns, and J. N. Sørensen. “Evaluation of proper orthogonal decomposition–based decomposition techniques applied to parameter-dependent nonturbulent flows”. In: *SIAM Journal on Scientific Computing* 21.4 (1999), pp. 1419–1434. DOI: 10.1137/S1064827598333181.

[77] E. J. Candès, X. Li, Y. Ma, and J. Wright. “Robust Principal Component Analysis?” In: *J. ACM* 58.3 (2011). DOI: 10.1145/1970392.1970395.

[78] H. Zou, T. Hastie, and R. Tibshirani. “Sparse Principal Component Analysis”. In: *Journal of Computational and Graphical Statistics* 15.2 (2006), pp. 265–286. DOI: 10.1198/106186006X113430.

[79] K. Lee and K. T. Carlberg. “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders”. In: *Journal of Computational Physics* 404 (2020), p. 108973. DOI: 10.1016/j.jcp.2019.108973.

[80] A. Kolmogoroff. “Über die beste Annäherung von Funktionen einer gegebenen Funktionenklasse”. In: *Annals of Mathematics* 37.1 (1936), pp. 107–110. ISSN: 0003486X. (Visited on 07/17/2023).

[81] D. Amsallem and C. Farhat. “Interpolation Method for Adapting Reduced-Order Models and Application to Aeroelasticity”. In: *AIAA Journal* 46.7 (2008), pp. 1803–1813. DOI: 10.2514/1.35374.

[82] R. Zimmermann. *Manifold interpolation and model reduction*. 2022. DOI: 10.48550/arXiv.1902.06502.

[83] O. Friderikos, E. Baranger, M. Olive, and D. Néron. *On the stability of POD Basis Interpolation via Grassmann Manifolds for Parametric Model Order Reduction in Hyperelasticity*. arXiv, 2020. DOI: 10.48550/arXiv.2012.08851.

[84] N. T. Son. “A real time procedure for affinely dependent parametric model order reduction using interpolation on Grassmann manifolds”. In: *International Journal for Numerical Methods in Engineering* 93.8 (2013), pp. 818–833. DOI: 10.1002/nme.4408.

[85] P.-A. Absil, R. Mahony, and R. Sepulchre. “Riemannian Geometry of Grassmann Manifolds with a View on Algorithmic Computation”. In: *Acta Applicandae Mathematicae* 80.2 (2004), pp. 199–220. DOI: 10.1023/B:ACAP.0000013855.14971.91.

[86] W. M. Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Rev. 2. ed. Amsterdam and New York: Academic Press, 2003. ISBN: 978-0121160517.

[87] A. Edelman, T. A. Arias, and S. T. Smith. “The Geometry of Algorithms with Orthogonality Constraints”. In: *SIAM Journal on Matrix Analysis and Applications* 20.2 (1998), pp. 303–353. DOI: 10.1137/S0895479895290954.

[88] R. S. Kulkarni. “Book Review: Differential geometry, Lie groups and symmetric spaces”. In: *Bulletin of the American Mathematical Society* 2.3 (1980), pp. 468–477. DOI: 10.1090/S0273-0979-1980-14772-2.

[89] D. Amsallem and C. Farhat. “Stabilization of projection-based reduced-order models”. In: *International Journal for Numerical Methods in Engineering* 91.4 (2012), pp. 358–377. DOI: 10.1002/nme.4274.

[90] J. B. MacQueen. “Some Methods for Classification and Analysis of MultiVariate Observations”. In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. University of California Press, 1967, pp. 281–297.

[91] D. Amsallem, M. J. Zahr, and K. Washabaugh. “Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction”. In: *Advances in Computational Mathematics* 41.5 (2015), pp. 1187–1230. DOI: 10.1007/s10444-015-9409-0.

[92] S. Grimberg, C. Farhat, R. Tezaur, and C. Bou–Mosleh. “Mesh sampling and weighting for the hyperreduction of nonlinear Petrov–Galerkin reduced–order models with local reduced–order bases”. In: *International Journal for Numerical Methods in Engineering* 122.7 (2021), pp. 1846–1874. DOI: 10.1002/nme.6603.

[93] B. Schölkopf, A. Smola, and K.-R. Müller. “Nonlinear Component Analysis as a Kernel Eigenvalue Problem”. In: *Neural Computation* 10.5 (1998), pp. 1299–1319. DOI: 10.1162/089976698300017467.

[94] S. T. Roweis and L. K. Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *science* 290.5500 (2000), pp. 2323–2326. DOI: 10.1126/science.290.5500.2323.

[95] J. B. Tenenbaum, V. de Silva, and J. C. Langford. “A global geometric framework for nonlinear dimensionality reduction”. In: *science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

[96] R. R. Coifman and S. Lafon. “Diffusion maps”. In: *Applied and Computational Harmonic Analysis* 21.1 (2006), pp. 5–30. DOI: 10.1016/j.acha.2006.04.006.

[97] G. Mishne, U. Shaham, A. Cloninger, and I. Cohen. “Diffusion nets”. In: *Applied and Computational Harmonic Analysis* 47.2 (2019), pp. 259–285. DOI: 10.1016/j.acha.2017.08.007.

[98] F. Anowar, S. Sadaoui, and B. Selim. “Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)”. In: *Computer Science Review* 40 (2021), p. 100378. DOI: 10.1016/j.cosrev.2021.100378.

[99] J. Schmidhuber. “Deep learning in neural networks: an overview”. In: *Neural networks : the official journal of the International Neural Network Society* 61 (2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

[100] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning Internal Representations by Error Propagation”. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[101] B. Bahmani and W. Sun. “Manifold embedding data-driven mechanics”. In: *Journal of the Mechanics and Physics of Solids* 166 (2022), p. 104927. DOI: `10.1016/j.jmps.2022.104927`.

[102] M. Mrosek, C. Othmer, and R. Radespiel. “Variational Autoencoders for Model Order Reduction in Vehicle Aerodynamics”. In: *AIAA AVIATION 2021 FORUM*. DOI: `10.2514/6.2021-3049`.

[103] F. J. Gonzalez and M. Balajewicz. “Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems”. In: *arXiv preprint arXiv:1808.01346* (2018). DOI: `10.48550/arXiv.1808.01346`.

[104] N. B. Erichson, M. Muehlebach, and M. W. Mahoney. “Physics-informed autoencoders for Lyapunov-stable fluid flow prediction”. In: *arXiv preprint arXiv:1905.10866* (2019). DOI: `10.48550/arXiv.1905.10866`.

[105] R. Maulik, B. Lusch, and P. Balaprakash. “Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders”. In: *Physics of Fluids* 33.3 (2021), p. 037106. DOI: `10.1063/5.0039986`.

[106] S. Dutta, P. Rivera-Casillas, B. Styles, and M. W. Farthing. “Reduced Order Modeling Using Advection-Aware Autoencoders”. In: *Mathematical and computational applications* 27.3 (2022). DOI: `10.3390/mca27030034`.

[107] S. Shen, Y. Yin, T. Shao, H. Wang, C. Jiang, L. Lan, and K. Zhou. *High-order Differentiable Autoencoder for Nonlinear Model Reduction*. 2021. DOI: `10.48550/arXiv.2102.11026`.

[108] L. Fulton, V. Modi, D. Duvenaud, D. I. W. Levin, and A. Jacobson. “Latent–space Dynamics for Reduced Deformable Simulation”. In: *Computer Graphics Forum* 38.2 (2019), pp. 379–391. DOI: `10.1111/cgf.13645`.

[109] D. Holden, B. C. Duong, S. Datta, and D. Nowrouzezahrai. “Subspace Neural Physics: Fast Data-Driven Interactive Simulation”. In: *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’19. New York, NY, USA: Association for Computing Machinery, 2019. DOI: `10.1145/3309486.3340245`.

[110] S. Jain, P. Tiso, J. B. Rutzmoser, and D. J. Rixen. “A quadratic manifold for model order reduction of nonlinear structural dynamics”. In: *Computers & Structures* 188 (2017), pp. 80–94. DOI: `10.1016/j.compstruc.2017.04.005`.

[111] J. B. Rutzmoser and D. J. Rixen. “A lean and efficient snapshot generation technique for the Hyper-Reduction of nonlinear structural dynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 325 (2017), pp. 330–349. DOI: `10.1016/j.cma.2017.06.009`.

[112] S. Jain and P. Tiso. “Hyper-Reduction Over Nonlinear Manifolds for Large Nonlinear Mechanical Systems”. In: *Journal of Computational and Nonlinear Dynamics* 14.8 (2019). DOI: `10.1115/1.4043450`.

[113] R. Geelen, S. Wright, and K. Willcox. “Operator inference for non-intrusive model reduction with quadratic manifolds”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115717. DOI: `10.1016/j.cma.2022.115717`.

[114] J. Barnett and C. Farhat. “Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction”. In: *Journal of Computational Physics* 464 (2022), p. 111348. DOI: `10.1016/j.jcp.2022.111348`.

[115] K. Carlberg, M. Barone, and H. Antil. “Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction”. In: *Journal of Computational Physics* 330 (2017), pp. 693–734. DOI: `10.1016/j.jcp.2016.10.033`.

[116] R. Everson and L. Sirovich. “Karhunen–Loeve procedure for gappy data”. In: *Journal of the Optical Society of America A* 12.8 (1995), pp. 1657–1664. DOI: `10.1364/JOSAA.12.001657`.

[117] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. “An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations”. In: *Comptes Rendus Mathématique* 339.9 (2004), pp. 667–672. DOI: 10.1016/j.crma.2004.08.006.

[118] S. Chaturantabut and D. C. Sorensen. “Nonlinear model reduction via discrete empirical interpolation”. In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764. DOI: 10.1137/090766498.

[119] P. Tiso and D. J. Rixen. “Discrete empirical interpolation method for finite element structural dynamics”. In: *Topics in Nonlinear Dynamics, Volume 1*. Springer, 2013, pp. 203–212. DOI: 10.1007/978-1-4614-6570-6_18.

[120] P. Tiso, R. Dedden, and D. Rixen. “A Modified Discrete Empirical Interpolation Method for Reducing Non-Linear Structural Finite Element Models”. In: *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - 2013*. New York, NY, USA: ASME, 2014. ISBN: 978-0-7918-5597-3. DOI: 10.1115/DETC2013-13280.

[121] N. C. Nguyen and J. Peraire. “An efficient reduced-order modeling approach for non-linear parametrized partial differential equations”. In: *International Journal for Numerical Methods in Engineering* 76.1 (2008), pp. 27–55. DOI: 10.1002/nme.2309.

[122] J. Baiges, R. Codina, and S. Idelsohn. “Explicit reduced-order models for the stabilized finite element approximation of the incompressible Navier–Stokes equations”. In: *International Journal for Numerical Methods in Fluids* 72.12 (2013), pp. 1219–1243. DOI: 10.1002/fld.3777.

[123] K. Martynov and U. Wever. “On polynomial hyperreduction for nonlinear structural mechanics”. In: *International Journal for Numerical Methods in Engineering* 118.12 (2019), pp. 701–717. DOI: 10.1002/nme.6033.

[124] D. Ryckelynck. “Hyper-reduction of mechanical models involving internal variables”. In: *International Journal for Numerical Methods in Engineering* 77.1 (2009), pp. 75–89. DOI: 10.1002/nme.2406.

[125] T. Ø. Aanonsen. “Empirical interpolation with application to reduced basis approximations”. Master’s Thesis. Trondheim, Ålesund, Gjøvik, Norway: Norwegian University of Science and Technology, 2009. URL: http://hdl.handle.net/11250/258487.

[126] H. Antil, S. E. Field, F. Herrmann, R. H. Nochetto, and M. Tiglio. “Two-step greedy algorithm for reduced order quadratures”. In: *Journal of Scientific Computing* 57.3 (2013), pp. 604–637. DOI: 10.1007/s10915-013-9722-z.

[127] M. A. Grepl, Y. Maday, N. C. Nguyen, and A. T. Patera. “Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 41.3 (2007), pp. 575–605. DOI: 10.1051/m2an:2007031.

[128] J. A. Hernández, J. Oliver, A. E. Huespe, M. A. Caicedo, and J. C. Cante. “High-performance model reduction techniques in computational multiscale homogenization”. In: *Computer Methods in Applied Mechanics and Engineering* 276 (2014), pp. 149–189. DOI: 10.1016/j.cma.2014.03.011.

[129] S. S. An, T. Kim, and D. L. James. “Optimizing cubature for efficient integration of subspace deformations”. In: *ACM transactions on graphics (TOG)* 27.5 (2008), pp. 1–10. DOI: 10.1145/1409060.1409118.

[130] C. Farhat, P. Avery, T. Chapman, and J. Cortial. “Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency”. In: *International Journal for Numerical Methods in Engineering* 98.9 (2014), pp. 625–662. DOI: 10.1002/nme.4668.

[131] C. Farhat, T. Chapman, and P. Avery. “Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models”. In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 1077–1110. DOI: 10.1002/nme.4820.

[132] J. B. Rutzmoser, D. J. Rixen, P. Tiso, and S. Jain. “Generalization of quadratic manifolds for reduced order modeling of nonlinear structural dynamics”. In: *Computers & Structures* 192 (2017), pp. 196–209. DOI: 10.1016/j.compstruc.2017.06.003.

[133] J. Rutzmoser. “Model Order Reduction for Nonlinear Structural Dynamics”. PhD thesis. Technische Universität München. URL: https://mediatum.ub.tum.de/1381943.

[134] D. Scheffold, C. Bach, F. Duddeck, G. Müller, and M. Buchschmid. “Vibration Frequency Optimization of Jointed Structures with Contact Nonlinearities using Hyper-Reduction”. In: *IFAC-PapersOnLine* 51.2 (2018), pp. 843–848. DOI: 10.1016/j.ifacol.2018.04.019.

[135] J. A. Hernandez, M. A. Caicedo, and A. Ferrer. “Dimensional hyper-reduction of nonlinear finite element models via empirical cubature”. In: *Computer Methods in Applied Mechanics and Engineering* 313 (2017), pp. 687–722. DOI: 10.1016/j.cma.2016.10.022.

[136] B. Brands, D. Davydov, J. Mergheim, and P. Steinmann. “Reduced-Order Modelling and Homogenisation in Magneto-Mechanics: A Numerical Comparison of Established Hyper-Reduction Methods”. In: *Mathematical and computational applications* 24.1 (2019), p. 20. DOI: 10.3390/mca24010020.

[137] J. Maierhofer and D. J. Rixen. “Model order reduction using hyperreduction methods (DEIM, ECSW) for magnetodynamic FEM problems”. In: *Finite Elements in Analysis and Design* 209 (2022), p. 103793. DOI: 10.1016/j.finel.2022.103793.

[138] S. Lall, P. Krysl, and J. E. Marsden. “Structure-preserving model reduction for mechanical systems”. In: *Physica D: Nonlinear Phenomena* 184.1-4 (2003), pp. 304–318. DOI: 10.1016/S0167-2789(03)00227-6.

[139] S. Grimberg, C. Farhat, and N. Youkilis. “On the stability of projection-based model order reduction for convection-dominated laminar and turbulent flows”. In: *Journal of Computational Physics* 419 (2020), p. 109681. DOI: 10.1016/j.jcp.2020.109681.

[140] D. Amsallem, M. J. Zahr, and C. Farhat. “Nonlinear model order reduction based on local reduced-order bases”. In: *International Journal for Numerical Methods in Engineering* 92.10 (2012), pp. 891–916. DOI: 10.1002/nme.4371.

[141] C. Bach, L. Song, T. Erhart, and F. Duddeck. *Stability conditions for the explicit integration of projection based nonlinear reduced-order and hyper reduced structural mechanics finite element models*. arXiv, 2018. DOI: 10.48550/arXiv.1806.11404.

[142] D. Ryckelynck. “A priori hyperreduction method: an adaptive approach”. In: *Journal of Computational Physics* 202.1 (2005), pp. 346–366. DOI: 10.1016/j.jcp.2004.07.015.

[143] D. Ryckelynck, F. Chinesta, E. Cueto, and A. Ammar. “On the a priori model reduction: Overview and recent developments”. In: *Archives of Computational Methods in Engineering* 13.1 (2006), pp. 91–128. DOI: 10.1007/BF02905932.

[144] A. Nouy. “A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations”. In: *Computer Methods in Applied Mechanics and Engineering* 199.23-24 (2010), pp. 1603–1626. DOI: 10.1016/j.cma.2010.01.009.

[145] D. Ryckelynck, L. Hermanns, F. Chinesta, and E. Alarcón. “An efficient ‘a priori’ model reduction for boundary element models”. In: *Engineering Analysis with Boundary Elements* 29.8 (2005), pp. 796–801. DOI: https://doi.org/10.1016/j.enganabound.2005.04.003.

[146] Y. Teng, M. A. Otaduy, and T. Kim. “Simulating Articulated Subspace Self-Contact”. In: *ACM Trans. Graph.* 33.4 (2014). DOI: 10.1145/2601097.2601181.

[147] M. Balajewicz, D. Amsallem, and C. Farhat. “Projection-based model reduction for contact problems”. In: *International Journal for Numerical Methods in Engineering* 106.8 (2016), pp. 644–663. DOI: 10.1002/nme.5135.

[148] T. Belytschko. *Nonlinear Finite Elements for Continua and Structures*. 2nd ed. New York: John Wiley & Sons Incorporated, 2013. ISBN: 978-1118632703.

[149] L. E. Malvern. *Introduction to the mechanics of a continuous medium*. Prentice-Hall series in engineering of the physical sciences. Englewood Cliffs, NJ: Prentice-Hall, 1969. ISBN: 978-0134876030.

[150] Livermore Software Technology Corporation. *LS-DYNA Theory Manual*. Livermore, CA, USA, 2018.

[151] K. Carlberg, C. Bou-Mosleh, and C. Farhat. “Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations”. In: *International Journal for Numerical Methods in Engineering* 86.2 (2011), pp. 155–181. DOI: 10.1002/nme.3050.

[152] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. “Array programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[153] C. Gu and J. Roychowdhury. “Model reduction via projection onto nonlinear manifolds, with applications to analog circuits and biochemical systems”. In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 112008, pp. 85–92. DOI: 10.1109/ICCAD.2008.4681556.

[154] B. I. Epureanu. “A parametric analysis of reduced order models of viscous flows in turbomachinery”. In: *Journal of Fluids and Structures* 17.7 (2003), pp. 971–982. DOI: 10.1016/S0889-9746(03)00044-6.

[155] T. Lieu and C. Farhat. “Adaptation of Aeroelastic Reduced-Order Models and Application to an F-16 Configuration”. In: *AIAA Journal* 45.6 (2007), pp. 1244–1257. DOI: 10.2514/1.24512.

[156] D. Amsallem and C. Farhat. “An Online Method for Interpolating Linear Parametric Reduced-Order Models”. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2169–2198. DOI: 10.1137/100813051.

[157] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[158] D. Amsallem and B. Haasdonk. “PEBL-ROM: Projection-error based local reduced-order models”. In: *Adv. Model. and Simul. in Eng. Sci.* 3 (2016). DOI: 10.1186/s40323-016-0059-7.

[159] I. S. Dhillon and D. S. Modha. “Concept Decompositions for Large Sparse Text Data Using Clustering”. In: *Machine Learning* 42.1 (2001), pp. 143–175. DOI: 10.1023/A:1007612920971.

[160] A. Singhal, C. Buckley, and M. Mitra. “Pivoted Document Length Normalization”. In: *SIGIR Forum* 51.2 (2017), pp. 176–184. DOI: 10.1145/3130348.3130365.

[161] J. Laska. *JASONLASKA/Spherecluster: Clustering Routines for the unit sphere*. 2019. URL: https://github.com/jasonlaska/spherecluster.

[162] C. Bach, D. Ceglia, L. Song, and F. Duddeck. “Randomized low–rank approximation methods for projection–based model order reduction of large nonlinear dynamical problems”. In: *International Journal for Numerical Methods in Engineering* 118.4 (2019), pp. 209–241. DOI: 10.1002/nme.6009.

[163] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995. DOI: 10.1137/1.9781611971217.

[164] J. Barnett, C. Farhat, and Y. Maday. *Neural-Network-Augmented Projection-Based Model Order Reduction for Mitigating the Kolmogorov Barrier to Reducibility of CFD Models*. 2022. DOI: 10.48550/arXiv.2212.08939.

[165] F. Romor, G. Stabile, and G. Rozza. “Non-linear Manifold Reduced-Order Models with Convolutional Autoencoders and Reduced Over-Collocation Method”. In: *Journal of Scientific Computing* 94.3 (2023). DOI: 10.1007/s10915-023-02128-2.

[166] S. E. Otto, G. R. Macchio, and C. W. Rowley. "Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.11 (Nov. 2023), p. 113130. DOI: 10.1063/5.0169688.

[167] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. "Contractive Auto-Encoders: Explicit Invariance during Feature Extraction". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Madison, WI, USA: Omnipress, 2011, pp. 833–840. ISBN: 9781450306195.

[168] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: https://www.tensorflow.org/.

[169] F. Chollet et al. *Keras*. https://keras.io. 2015.

[170] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. DOI: 10.48550/arXiv.1412.6980.

[171] S. Anderson, C. White, and C. Farhat. "Space–local reduced–order bases for accelerating reduced–order models through sparsity". In: *International Journal for Numerical Methods in Engineering* 124.7 (2023), pp. 1646–1671. DOI: 10.1002/nme.7179.