Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

Non-intrusive techniques for Reduced Order Modeling

MASTER FINAL THESIS

Final Thesis developed by:
Girish, Shreyas

Directed by:
Riccardo Rosssi Bernecoli

Master in:
Numerical Methods in Engineering

Barcelona, May 2025

Department of d'Enginyeria Civil i Ambiental (UPC)

Abstract

This thesis focuses on the development, implementation, and validation of non-intrusive Reduced Order Modeling (ROM) techniques for structural dynamic systems. Full-Order Models (FOM) are generated using Kratos Multiphysics for a two-dimensional cantilever beam subjected to varying multi-directional load cases. The displacement, velocity, and acceleration snapshots are collected to build the high-fidelity dataset. Proper Orthogonal Decomposition (POD) is employed to extract dominant modes, achieving a dimensionality reduction of over 98% while preserving more than 99.99% of the system energy.

Two ROM approaches are explored. The Direct ROM method constructs reduced mass and stiffness matrices through Galerkin projection using the reduced basis. The Operator Inference ROM, on the other hand, infers system operators directly from the snapshot data without requiring access to the full governing equations, offering a non-intrusive and data-driven alternative. Both ROMs are solved using Newmark integration coupled with Newton-Raphson iterations to capture nonlinear dynamic behavior.

Validation is performed by comparing the ROM outputs against the FOM solutions across displacement, velocity, acceleration, and force responses. Results show that the Direct ROM method achieves excellent agreement with the FOM, particularly for displacement predictions. The Operator Inference ROM demonstrates slightly higher reconstruction errors in displacement but outperforms the Direct ROM in velocity and acceleration predictions for certain cases. Additionally, the Operator Inference method shows promise for complex systems where access to full matrices is limited.

Finally, a detailed comparison of solution times and error metrics highlights the efficiency and potential trade-offs of both approaches, providing critical insights into the practical deployment of ROM techniques for real-time structural simulations. The findings establish Operator Inference ROM as a competitive alternative for rapid dynamic simulations while maintaining acceptable levels of accuracy.

Contents

List of Figures

List of Tables

1 Introduction

Numerical simulation is a powerful computational tool for modeling and evaluating the behavior of physical systems through mathematical representations. This approach approximates solutions by discretizing the governing equations and solving them numerically rather than relying on analytical solutions, which may be challenging or unattainable for certain complex situations. It is therefore essential in contemporary engineering and scientific inquiry as it facilitates system performance forecasting, assessment of design alternatives, and investigation of phenomena that are challenging to study experimentally. Consequently, numerical simulations are commonly employed in domains such as fluid dynamics, structural mechanics, heat transfer, and electromagnetism [1, 2].

Advancements in high-performance computing and simulation software have established methodologies such as the Finite Element Method (FEM), Finite Volume Method (FVM), and Finite Difference Method (FDM) as essential instruments for analyzing real-world issues. These technologies have significantly enhanced efficiency, reduced prices, and elevated safety across several sectors. In structural analysis, numerical simulation particularly enables engineers to predict the response of structures to mechanical forces, thermal influences, and dynamic excitations. FEM disaggregates intricate geometries into smaller, more tractable finite elements, facilitating localized approximations that collectively represent the behavior of a whole system. This is especially beneficial in disciplines such as aerospace, civil engineering, and biomechanics [3].

These simulations necessitate a balance of precision, computational efficiency, and adaptability. A modular implementation architecture, consisting of essential components like the Scheme, BuilderAndSolver, and Strategy, helps this equilibrium by differentiating between implementation specifics and solution methodologies. The Scheme supervises system contributions, the BuilderAndSolver constructs global matrices and applies boundary conditions, and the Strategy manages the iterative process. This systematic methodology not only optimizes the simulation process but also enhances adaptability by enabling users to utilize established procedures or modify them to suit personal requirements [4].

Supporting this robust basis, full-order models (FOMs) are commonly employed to ensure accurate and comprehensive simulation outcomes. Full-order simulation techniques offer full insights into system behavior by capturing all relevant scales and dynamics, making them vital for high-fidelity assessments in advanced engineering applications [5]. A detailed explanation is presented in the following section.

1.1 Full-Order Simulation:

A Full-Order Model (FOM) is a detailed, high-resolution numerical simulation of a physical system that meticulously resolves all governing equations with minimal simplifications. In structural simulations, detailed meshes are used for accurate spatial resolution and material modeling, including nonlinearities, anisotropic behavior, and, when needed, dynamic and multi-physics effects. These models, typically developed using the Finite Element Method (FEM), represent complex structural responses under various loads and boundary conditions [6].

For static analyses, the governing equation is given by

$$Ku = F$$

where K denotes the global stiffness matrix, u the displacement vector, and F the external force vector. This equation is often solved using direct methods such as LU decomposition, which efficiently determines the displacement field under constant or slowly varying loads. In contrast, dynamic problems incorporate inertia and damping effects, leading to the equation of motion:

$$M\ddot{u} + C\dot{u} + Ku = F$$

which is integrated over time using methods like the implicit, second-order accurate Newmark-Beta method. This method allows for control over numerical damping and stability through the tuning of parameters β and γ. Nonlinearities arising from material or geometric effects are managed at each time step using iterative techniques, such as the Newton-Raphson method, to ensure convergence.

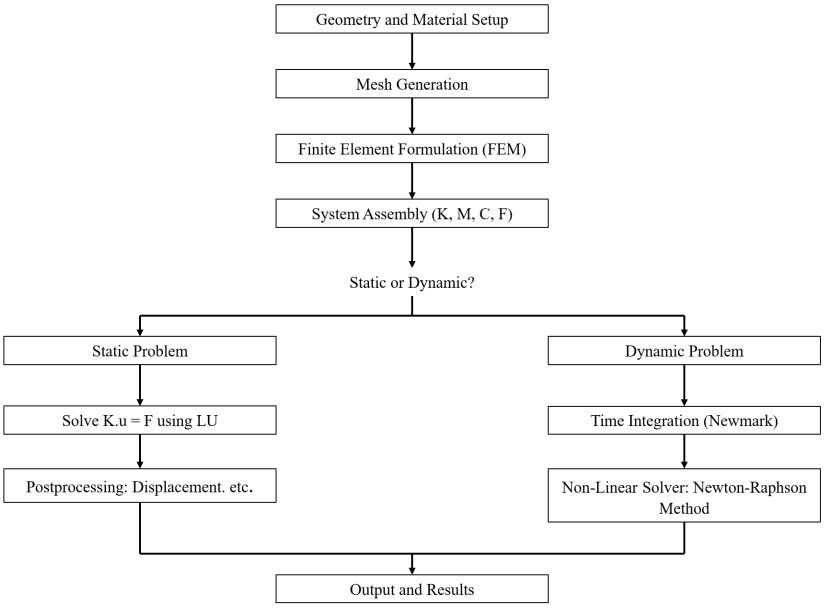


Figure 1: General workflow of Full Order Model (FOM).

Figure 1 illustrates the comprehensive workflow of Full-Order Models, including a sequential approach for both static and dynamic system analysis. The flowchart illustrates the discretization and resolution of the system's governing equations, commencing with mesh generation and finite element formulation, advancing to global matrix assembly, and concluding with equation resolution through LU decomposition for static scenarios or time integration methods for dynamic scenarios.

The rigorous formulation of Full-Order Models (FOMs) remains vital in high-stakes fields like aerospace, civil infrastructure, and biomedical engineering, where simulation

accuracy is non-negotiable. However, their computational intensity and high resource consumption have driven the pursuit of more efficient alternatives. Emerging research highlights a powerful non-intrusive, data-driven framework based on operator inference (OpInf), which constructs Reduced-Order Models (ROMs) directly from high-fidelity simulation data. By extracting reduced operators without altering core numerical solvers, this method preserves essential stability properties while drastically reducing computational costs. As a result, ROMs enable real-time performance for tasks such as optimization, design iteration, and control system integration.

The subsequent section will discuss Reduced-Order Models (ROMs), the focal point of this thesis, and examine their creation and implementation to achieve a balance between computing efficiency and simulation accuracy.

1.2 Reduced Order Simulation:

Reduced Order Modeling (ROM) is an advanced computational methodology devised to substantially diminish the complexity and computational burden associated with high-fidelity Full-Order Models (FOMs), while accurately preserving the critical dynamics and responses of physical systems. Finite Element Method (FEM)-based formulations typically require extensive domain discretization, generating large systems of equations that become computationally prohibitive, particularly during parametric studies, real-time evaluations, or optimization procedures. To address these computational challenges, ROM strategies project the comprehensive system onto a lower-dimensional subspace, capturing the most influential modes that dictate the system's behavior. Commonly utilized techniques such as Proper Orthogonal Decomposition (POD), Reduced Basis (RB) methods, and Krylov subspace methods systematically extract these dominant modes, resulting in significantly accelerated computational performance without sacrificing essential physical fidelity [7, 8]

The versatility of ROM has led to its widespread adoption across various engineering disciplines. Within structural engineering contexts, ROMs are particularly effective in evaluating stress distributions, deformation patterns, and dynamic responses of large-scale and intricate systems—including aerospace components, bridges, and mechanical assemblies—where rapid analysis is essential for iterative design and optimization cycles. Similarly, in fluid dynamics, ROM approaches efficiently identify principal flow structures within turbulent regimes, thereby enabling real-time predictive modeling and control. Moreover, ROM methodologies have demonstrated substantial value in thermal analysis, electromagnetics, and complex multiphysics problems, significantly reducing computational overhead and enabling extensive parametric analyses and uncertainty quantification studies that would otherwise be impractical with traditional FOMs.

An important consideration within structural dynamics modeling is the presence of nonlinear stiffness parameters, typically arising from nonlinear material behaviors or large structural deformations. Although high-fidelity simulations adequately capture such complexities, they impose significant computational demands, limiting their applicability in scenarios demanding rapid or numerous evaluations, such as real-time applications, optimization, or control-oriented tasks. Conventional model order reduction (MOR) techniques aim to address these computational constraints by approximating the high-dimensional

system with fewer state variables, thereby significantly reducing computational requirements. However, traditional MOR methods often necessitate direct access to detailed system operators, which are not always available, particularly when utilizing commercial simulation software [9].

To overcome these limitations, recent advancements propose a non-intrusive data-driven MOR approach based on operator inference (OpInf). This novel technique constructs Reduced-Order Models directly from simulation-generated datasets without explicitly accessing or modifying the underlying numerical solver frameworks. Operator inference methods thus retain the essential physical and stability characteristics inherent in the original system while substantially enhancing computational efficiency. This development not only enables broader applicability across diverse computational platforms but also facilitates real-time simulation capabilities, thereby expanding opportunities in rapid design iterations, optimization, and real-time control applications.

The subsequent sections will further elaborate on the methodology, implementation specifics, and practical implications of operator inference-based ROMs, detailing their efficacy in achieving computational efficiency while maintaining high fidelity in structural dynamics applications.

1.2.1 Integration with Static and Dynamic Structural Simulation

The conventional full-order model (FOM) workflow for structural simulation, applicable to both static and dynamic systems, begins with the specification of geometry, material properties, and mesh generation, subsequently leading to the assembly of global matrices (stiffness, mass, and damping) and the imposition of boundary conditions and loads. In static analysis, this results in solving $\mathbf{Ku} = \mathbf{F}$ by techniques such as LU decomposition, whereas in dynamic analysis, time-dependent governing equations are integrated utilizing the Newmark approach, with the Newton-Raphson iterative scheme applied to address nonlinearities. This extensive FOM procedure offers a thorough simulation of structural dynamics but at a significant computing expense [9].

Building upon this FOM foundation, Reduced Order Modeling (ROM) incorporates an additional series of stages designed to extract the most critical dynamic properties from the full-order simulation. Upon producing the full-order model and acquiring simulation data, commonly known as "snapshots," Reduced Order Modeling (ROM) techniques, including Proper Orthogonal Decomposition (POD) and Reduced Basis (RB) procedures, are utilized to ascertain a lower-dimensional basis that encapsulates the essential dynamics of the system. This reduced basis underpins the projection of the full-order governing equations onto a considerably smaller subspace, resulting in a reduced system of equations that preserves the fundamental behavior while necessitating significantly fewer processing resources for resolution. The FOM workflow offers comprehensive detail, whereas the ROM workflow condenses this information to its essential elements, rendering it especially beneficial for applications requiring swift simulation, optimization, or real-time analysis.

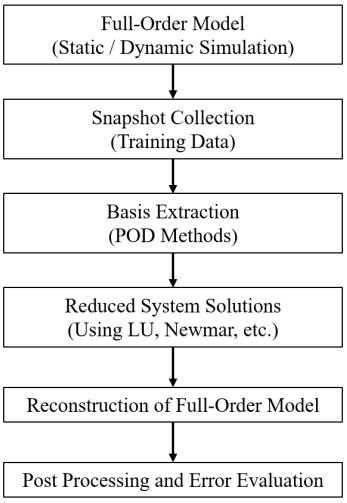


Figure 2: General workflow of Reduced Order Model (ROM).

The procedures for establishing and resolving a reduced order model diverge from those of a full-order model due to the incorporation of tasks such as snapshot collection, basis extraction, and projection. These phases are essential for creating an optimized system that preserves precision while markedly decreasing computing demands. The normal workflow for ROM is illustrated in the flowchart below, detailing the processes from snapshot creation to the resolution of the reduced system.

Structural dynamics models often incorporate nonlinear stiffness parameters due to nonlinear material properties or substantial deformations. High-fidelity simulations can capture this complexity; however, they incur significant processing costs, making them impractical for real-time applications or situations necessitating multiple evaluations, such as optimization or control tasks. Classic model order reduction (MOR) techniques have been developed to reduce computational expenses. These methods aim to streamline the simulation model by approximating it with a reduced number of variables, hence decreasing computational time and resource expenditure. Nonetheless, these conventional MOR approaches sometimes require access to the complete system operators, which may not always be attainable, particularly when utilizing commercial simulation software.

2 Scope of the Thesis

This thesis aims to thoroughly investigate and enhance non-intrusive Reduced Order Models (ROMs) that effectively characterize the behavior of nonlinear systems through sophisticated data-driven methodologies. The study concentrates on the advancement and implementation of Operator Inference-based Reduced Order Modeling methodologies specifically designed for structural mechanics contexts, encompassing static and linear dynamic structural analysis. The structural system being analyzed is a cantilever beam with a fixed base subjected to loading at three locations. The simulations are conducted using the Kratos Multiphysics program, selected for its superior computational capability and precision in modeling complex structural processes. The subsequent construction and validation of the ROM rely on the displacement data provided by the high-fidelity full-order model simulations.

2.1 Problem statement

This study explicitly addresses two types of structural mechanics problems: the static structural problem, represented by an equation, and the Linear Dynamic Structural Problem, defined by the equation of motion. The model is simplified by intentionally excluding damping effects from the dynamic analysis to facilitate a more detailed examination of the interactions between mass and stiffness components. This strategy choice facilitates a more focused examination of ROM creation and evaluation, devoid of the added complexity of dampening.

The structural geometry under consideration is strictly confined to a two-dimensional cantilever beam shape. This decision was made to balance computational feasibility and complexity, hence ensuring substantial validation of the established ROM approaches. The reduced 2D representation provides a clear, practical benchmark scenario with broad applicability that reflects fundamental structural mechanics problems. Furthermore, the material is classified as linear elastic, hence minimizing the intricacies of nonlinear material behaviors and greatly streamlining numerical computations, which is crucial due to restricted processing capabilities and time limitations.

Kratos Multiphysics is a robust computational platform capable of effectively handling the structured and extensive datasets required for Reduced Order Modeling approaches, hence all numerical simulations and associated analyses are conducted with it. FOM simulations generate systematically organized and preserved displacement datasets to ensure consistency, repeatability, and ease of access for further ROM development. The construction and validation of a robust and dependable ROM rely on this systematic methodology of dataset management.

This study employs singular value decomposition (SVD) primarily to extract dominant modes from the displacement datasets of the stationary structural condition. Galerkin projection methods subsequently provide a reduced stiffness matrix from the extracted modes. Comparative assessments utilizing established, highly precise full-order results derived from Kratos solver simulations corroborate the ROM outcomes. This comparison of FOM and ROM solutions aims to deliver both quantitative and qualitative assessments of

ROM correctness and computing efficiency, thereby demonstrating the potential practical benefits of employing ROM approaches in structural mechanics.

2.2 Research Questions

The work aims to examine critical problems to enhance the understanding of the possibilities and limitations of Operator Inference-based Reduced Order Modeling approaches.

1. In static structural contexts, how effectively do non-intrusive Operator Inference-based reduced-order modeling techniques predict structural behavior compared to comprehensive full-order simulations?

2. What influence can dataset size and quality exert on the anticipated reliability and precision of ROM methodologies, and what defines an ideal training dataset to ensure robust model performance?

3. Can the proposed ROM approach accurately predict structural responses under novel, unverified loading situations not present in the training dataset?

4. What computational performance advantages might reduced order modeling (ROM) techniques offer compared to traditional finite element modeling (FEM)? What is the significance of the alterations in computational effort?

5. Can the proposed ROM technique reliably predict structural responses despite minor fluctuations in the initial training parameters and conditions?

2.3 Expected Contributions

Three significant advancements predicted in this thesis will significantly add to the present knowledge base in lower order modeling and structural mechanics:

- Methodological innovation: developed especially for structural mechanics issues, the invention and thorough validation of a revolutionary non-intrusive Operator Inference ROM technique. This development greatly enhances present approaches, therefore enhancing theoretical and practical knowledge of ROM implementations.

- Enhanced computational efficiency: a clear description and computation of the potential computational time reductions employing ROM implementations. These efficiency gains especially help real-time monitoring systems, iterative design techniques, and quick parametric research.

- For suitable training data features, give comprehensive recommendations and instructions so you provide essential insights on balancing computer resources and predicted accuracy—qualities required for actual engineering decision-making.

- Robust validation systems—those that which matches ROM results to realistic FOM simulations—ensure the robustness and dependability of developed ROM procedures by means of unambiguous error measurements and acceptance criteria.
- Using pragmatic implementation advice and benchmarking, create and provide detailed Python-based scripts included into the Kratos Multiphysics platform and arrange benchmark scenarios. These deliverables will let researchers and industry experts replicate, adjust, and basically apply ROM approaches.

Finally, this work tries to combine theoretical discoveries with practical applications of reduced order modeling techniques in structural mechanics, thereby producing unambiguous, efficient, and dependable methodologies. By carefully addressing stated research objectives, setting scope boundaries, and describing projected contributions, the thesis offers a strong framework for next structural studies and optimization efforts, thereby considerably increasing computational efficiency and dependability.

3 Literature Review

Reduced order modeling (ROM) has become an essential tool in computational mechanics for approximating complex high-fidelity models with much lower computational cost. As full order models (FOMs), such as those based on finite element methods or high-resolution discretizations of partial differential equations (PDEs), tend to be prohibitively expensive for multi-query analysis, control, optimization, or real-time applications, ROM techniques aim to capture the dominant behavior of a physical system with a reduced number of degrees of freedom. The evolution of ROM techniques—from projection-based methods using Proper Orthogonal Decomposition (POD) to emerging data-driven and machine learning approaches—reflects the growing demand for efficient and robust simulation tools in engineering and science [10].

ROM strategies can be broadly classified into intrusive and non-intrusive approaches. Traditionally, ROM methods have relied on projecting the governing equations of a FOM onto a low-dimensional subspace. Techniques such as POD–Galerkin methods extract dominant modes from solution snapshots and derive reduced systems that inherit many properties of the original model. This projection-based reduction is well understood mathematically and guarantees stability and convergence under certain conditions [10]. Fundamentally, ROM seeks a mapping $u(t;\mu) \approx Vur(t;\mu)\mathbf{u}(t;\mu) \approx \mathbf{V}\mathbf{u}_r(t;\mu)$, where $V\mathbf{V}$ is the basis matrix computed by, for instance, POD, and $ur\mathbf{u}_r$ represents the reduced coordinates. Such methods effectively lower the computational cost of solving time-dependent PDEs by reducing the number of state variables while still capturing the essential dynamics of the system [10, 8] Traditional ROMs, often referred to as "intrusive" approaches, require access to and modification of the governing equations. In these methods, the high-dimensional equations are projected onto a reduced space, and additional techniques such as hyper-reduction (for example, the discrete empirical interpolation method) are applied when nonlinearity is present, to mitigate the computational burden during online evaluation. Although these methods are rigorously derived from first principles, they can struggle with highly nonlinear or parametrically complex systems due to the limitations inherent in linear subspace approximations [10]. In contrast, non-intrusive approaches treat the original FOM as a "black box" and rely on input–output data. Recent developments in data-driven methods have resulted in techniques such as deep learning-based ROMs (DL-ROMs), where neural networks learn the mapping from parameters to the solution manifold without requiring explicit knowledge of the governing equations. For instance, POD–LSTM–ROM approaches integrate long short-term memory networks to capture temporal dynamics for nonlinear, time-dependent problems [11]. Additionally, surrogate modeling and regression methods, including Gaussian process regression and support vector machines, are employed to interpolate system behavior directly from observed data. The primary advantage of non-intrusive methods lies in their flexibility and potential to handle strong nonlinearities without the need to reformulate or access the underlying PDEs [11].

One of the most widely used techniques in ROM development is Proper Orthogonal Decomposition (POD), which involves collecting high-fidelity solution snapshots for various parameter sets and time instances, and then applying Singular Value Decomposition

(SVD) to the snapshot matrix. This process identifies orthogonal modes that capture the maximum variance (or energy) of the data, and by truncating the modes to retain only the most significant ones, a reduced basis is formed. This procedure minimizes the projection error in an $L2L^2$ sense and yields an optimal low-dimensional representation of the solution space [10, 8]. However, POD has its limitations; when the underlying solution manifold is highly nonlinear, a linear combination of modes may not adequately capture the essential dynamics, potentially resulting in poor predictive performance outside the training regime.

There are several challenges associated with constructing and implementing ROMs. Handling nonlinearity is one of the foremost issues, as many physical systems exhibit strongly nonlinear behavior that is difficult to capture using a fixed linear subspace. Both traditional and data-driven methods can struggle if the reduced basis does not adequately represent the full solution space. Furthermore, while ROMs are designed to be efficient during online evaluation, the offline phase involving training or basis construction can be computationally expensive, especially when large datasets are involved. Ensuring that the ROM preserves the stability properties of the full model is nontrivial, as inaccuracies or instabilities in the reduced model can lead to significant errors in prediction. Capturing a wide range of parameter values also necessitates a robust and flexible basis that can generalize well, a challenge particularly acute in fixed projection methods [10, 8]. Additionally, data-driven ROMs, although flexible, often result in "black box" models that lack the interpretability associated with physics-based methods. Beyond traditional projection methods, several non-intrusive techniques have emerged. Machine learning-based ROMs leverage neural networks, including deep architectures such as convolutional and recurrent networks, to approximate the mapping from parameters to reduced coordinates. For example, POD–LSTM–ROM frameworks have demonstrated promise in predicting long-time dynamics of nonlinear systems by effectively capturing temporal correlations [11]. Another prominent data-driven technique is Dynamic Mode Decomposition (DMD), which decomposes complex systems into spatiotemporal coherent structures with associated eigenvalues. DMD is particularly useful for systems where capturing the evolution of modes over time is critical; it can extract coherent structures, estimate growth or decay rates, and facilitate forecasting by reconstructing future states from the temporal evolution of identified modes. However, the linear framework underlying DMD can be a limitation in highly nonlinear contexts, prompting ongoing research into extensions that incorporate nonlinear effects. Additional data-driven approaches, including surrogate models based on regression methods or Gaussian processes, have also been used to interpolate system behavior without explicit recourse to the underlying physics. Recent advancements in machine learning have paved the way for novel ROM approaches that learn the dynamics of a system directly from data. By incorporating architectures such as LSTM networks or autoencoders, these frameworks can enhance time extrapolation capabilities, allowing predictions to extend beyond the training window—a critical benefit in time-sensitive applications. Moreover, machine learning approaches reduce dependency on prior knowledge of the governing equations, making them applicable to complex, black-box systems. Once trained, such models can adapt more gracefully to parameter variations compared to traditional projection-based methods. However, these advantages come with challenges: the training phase is computationally demanding and requires a large volume of representa-

tive data, and ensuring physical consistency and interpretability in the resulting models remains a significant hurdle [11].

Dynamic Mode Decomposition (DMD) offers a powerful framework for analyzing and modeling complex dynamical systems by decomposing a system's behavior into modes characterized by fixed spatial patterns and exponential time dynamics. DMD effectively extracts dominant patterns, estimates stability characteristics through growth or decay rates, and facilitates forecasting by reconstructing future states based on the evolution of these modes. Despite its computational efficiency and straightforward implementation, DMD's reliance on a linear evolution of modes may limit its effectiveness when dealing with strongly nonlinear interactions. As such, research is actively exploring ways to extend DMD to better handle nonlinearities.

When evaluating ROM techniques within the context of computational mechanics, each method presents its own set of advantages and limitations. Traditional intrusive methods, such as POD–Galerkin, benefit from strong theoretical foundations and the ability to preserve physical laws if formulated correctly. However, these methods require direct access to and modification of the full system equations, and they may struggle to handle strong nonlinearities. Non-intrusive, data-driven methods, on the other hand, treat the full model as a black box and offer flexibility and adaptability to complex systems, though they come with high offline training costs and often suffer from a lack of physical interpretability. Dynamic Mode Decomposition excels at capturing temporal dynamics and is computationally efficient, yet its assumption of linear mode evolution can be a drawback in nonlinear regimes. Machine learning-based ROMs, while capable of handling nonlinearities and extending predictions beyond the training window with minimal reliance on explicit physics-based models, face challenges in terms of training expense, interpretability, and guarantees of stability [10, 8].

An emerging platform for integrating advanced ROM techniques is Kratos Multiphysics, an open-source framework designed for simulating complex multi-physics problems. Kratos' modular architecture facilitates the coupling of full order models with reduced order models, enabling real-time or near-real-time simulations. Its data management and scripting capabilities support the implementation of machine learning-based ROMs and other data-driven approaches, making it particularly valuable in multi-disciplinary applications where efficient computation is critical. Although the literature on ROMs within Kratos is still developing, the framework offers promising avenues for integrating advanced ROM techniques within a robust computational environment [10]. Despite significant progress in the field of ROM, several research gaps remain. One major challenge is the handling of extreme nonlinearities, where existing linear subspace techniques may fail to capture complex dynamics adequately. There is also a need for hybrid approaches that combine the interpretability of physics-based ROMs with the flexibility of data-driven methods. Uncertainty quantification in ROM predictions, especially when extrapolating beyond the training data, remains an ongoing challenge. Moreover, achieving scalability in multi-physics applications, particularly in seamlessly integrating ROM techniques into platforms like Kratos, requires further development to ensure both accuracy and efficiency. For machine learning-based ROMs, reducing training costs and enhancing model interpretability are key future directions that must be addressed [10, 8].

In summary, reduced order modeling is a vibrant and evolving field that plays a critical role in modern computational mechanics. While traditional intrusive methods such as POD–Galerkin offer robust theoretical foundations, non-intrusive data-driven techniques—including machine learning-based ROMs and dynamic mode decomposition—provide enhanced flexibility and efficiency in dealing with complex, nonlinear systems. Platforms like Kratos Multiphysics further enhance these methods by offering a flexible computational environment for multi-physics applications. Nonetheless, challenges such as ensuring model stability, reducing offline training costs, and improving interpretability persist, and future research in hybrid approaches and uncertainty quantification is expected to further bridge the gap between high-fidelity simulation and real-time application needs [11, 10, 8].

The research carried out in this Master's thesis focuses on solving important problems in the area of non-intrusive reduced-order modeling (ROM) for structural mechanics. One key issue it tackles is the lack of Operator Inference-based ROM methods that are specifically designed for structural problems like static and linear dynamic analysis. The study also investigates how the size and quality of the input data affect the accuracy and reliability of ROMs, which is a topic that has not been widely covered in previous work. In addition, it tests whether the developed ROMs can make correct predictions for new loading conditions that were not part of the original training data. The thesis also measures how much faster ROMs can perform compared to full-order models, giving insight into their real-world advantages. Lastly, it provides strong validation methods and practical tools, such as Python scripts built into the Kratos Multiphysics platform, that help make these models more useful and easier to apply in engineering and research.

4 Methodology

This chapter presents a detailed methodology for the development and validation of non-intrusive Reduced Order Models (ROMs) based on Operator Inference techniques, specifically tailored for both Static and Dynamic structural simulations. The methodology is an account of procedures and steps to achieve the high-fidelity simulations with computationally efficient reduced-order surrogate models. By employing a data-driven projection and inference strategy, the approach aims to retain the physical accuracy and predictive capability of full-scale simulations while drastically reducing computational time and resource requirements. Each stage of this methodology—right from data generation to model reduction and the subsequent validation—has been designed and executed to ensure reproducibility, accuracy, and flexibility.

The overall process involves several key components: constructing a robust Full Order Model (FOM) using Kratos Multiphysics, generating simulation data through parametric sweeps of loading conditions, preprocessing and organizing the data into structured formats, applying reduction techniques to extract dominant modes, and finally utilizing operator inference techniques to construct a reduced model. This ROM is then benchmarked against the original FOM to assess performance, accuracy, and computational benefits. The following sections explain each of these stages in detail.

4.1 Overall Approach

The initial phase of this research emphasized the preparation and execution of the Full Order Model, which serves as the backbone for data-driven ROM construction. This phase is crucial because the quality of data extracted from FOM simulations directly impacts the quality of the reduced-order model. The approach is divided into logical steps that include the definition of the model geometry, the specification of boundary and load conditions, the generation of the mesh, the assignment of material properties, the setup of the solver and the execution of simulations. These tasks are completed using the GiD graphical interface together with Kratos Multiphysics, a powerful open-source C++ multiphysics framework.

This step-by-step setup makes it easier to carefully manage each part of the simulation process. It also helps with automating the work by using scripts. The method is strong because it focuses on both flexibility and automation. This is done using JSON files for settings and Python scripts to run the process. Also, by changing input values over a range (called a parametric sweep), the training data includes many different behaviors. This is important to create a ROM that can make good predictions in different situations.

4.1.1 Overview of ROMs

Reduced Order Modeling (ROM) has emerged as a critical technique for accelerating simulations in computational mechanics, especially when repeated evaluations of a system are required, such as in optimization, control, or real-time simulation contexts. In this project, the ROM is constructed by projecting the high-dimensional outputs of a Full Order Model

onto a lower-dimensional subspace that retains the dominant modes of behavior. This projection is achieved using Proper Orthogonal Decomposition (POD), which identifies basis vectors from snapshot data that account for the most prominent features of the system.

To create this ROM, we begin with the Kratos Multiphysics environment to simulate the structural behavior of a 2D model under varying loading conditions. The Full Order Model is parameterized by external loading vectors, which are systematically varied to generate a rich dataset. Each simulation yields a solution vector—typically a displacement field across nodes—which is stored and used to build a snapshot matrix. This matrix, when subjected to Singular Value Decomposition (SVD), provides a set of orthonormal basis vectors that form the reduced space.

The core idea is to utilize these basis vectors to project high-dimensional operators such as mass, stiffness, and force matrices into a low-dimensional space. The governing equations of motion are then approximated in this reduced space, significantly decreasing the computational complexity. For this study, the focus is on both static (steady-state equilibrium) and dynamic (time-dependent) behaviors, requiring the ROM to replicate structural deformations and oscillatory responses accurately. By narrowing the scope to loading conditions as the sole varying parameter, the project simplifies the parametric space while retaining sufficient complexity for a meaningful reduction exercise.

4.1.2 Setting Up the Full-Order Model

The Full Order Model (FOM) used in this study is constructed using the GiD application, which provides a user-friendly graphical interface to define geometry, mesh the domain and assign simulation conditions. The geometry selected is a two-dimensional cantilever beam fixed at its base. As shown in the figure 3, the beam has a rectangular profile with clearly defined dimensions: a length of 1 meter and a height of 10 meters. It is discretized into structured triangular finite elements for easy computations. The geometry represents a classical test case in structural mechanics, offering predictable deformation patterns under load and simplifying validation.

The boundary conditions play a crucial role in governing the response of the structure. The beam is fully fixed along its bottom edge, meaning all translational and rotational degrees of freedom (DOFs) are constrained at the nodes on this face. This simulates a clamped support condition, ensuring zero displacement and rotation. On the free end or upper face of the beam, external loads are applied. These loads are parametrized in three distinct directions (for example, horizontal and vertical components) and vary between multiple simulations to generate a diverse set of structural responses.

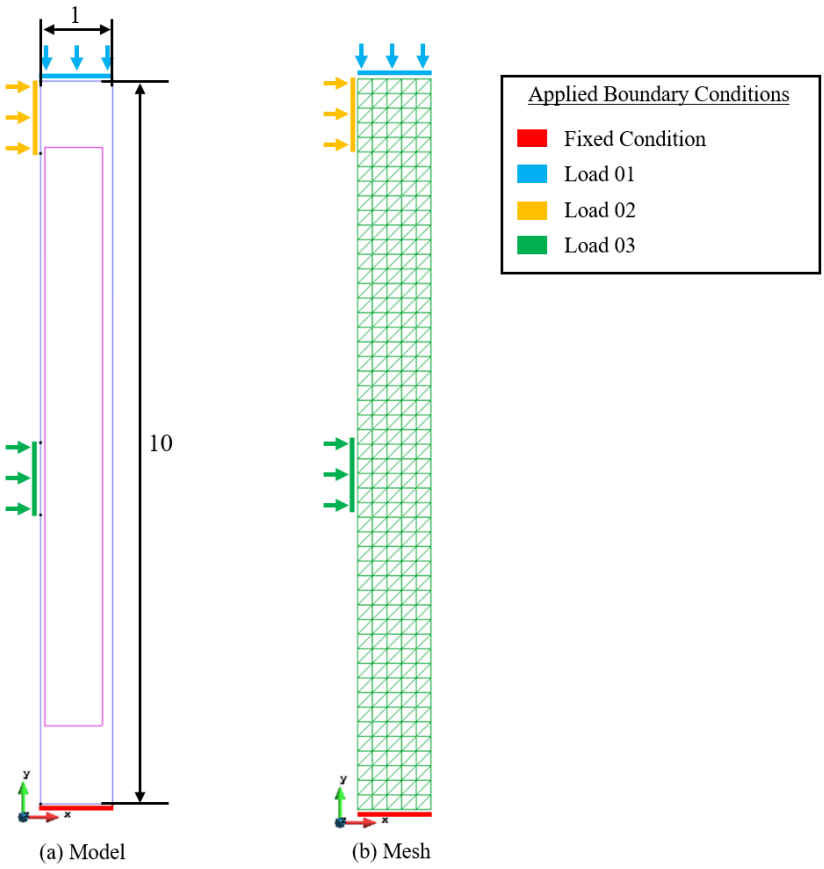


Figure 3: 2D structural model used for FOM simulations

The mesh used in the simulation is generated with the appropriate density to strike a balance between computational cost and solution accuracy. The domain is merged using structured triangular elements, ensuring numerical stability and allowing easy access to nodal displacement vectors and element-level matrices. The resulting mesh, geometry, and boundary conditions are exported into a series of configuration files that define the Kratos simulation environment. These files include:

- `.mdpa`: Contains mesh data, nodal coordinates, element definitions, and boundary condition tags.
- `ProjectParameters.json`: Specifies the simulation settings, including solver types, time stepping parameters, boundary conditions, and analysis options.
- `StructuralMaterials.json`: Defines the material properties such as Young's modulus, Poisson's ratio, and density.
- `MainKratos.py`: A Python driver script to initialize, configure, and run the Kratos solver.

Together, these files define a complete simulation pipeline. The `MainKratos.py` script serves as the entry point to the simulation. When executed, it sequentially processes the

configuration files, initializes the solver settings, loads the mesh and model parts, applies the material and boundary conditions, and executes the numerical analysis. The solver then writes the results—displacement fields, nodal forces, and system matrices—to the output folders for upcoming processing.

4.1.3 Data Generation and Pre-processing

The generation of a representative dataset for training the ROM is an important step that determines the accuracy and robustness of the reduced model. The `MainKratos.py` script is extended to enable iterative execution over a range of input loading conditions. This modification introduces automation and user-defined configurability into the simulation process, creating high-volume dataset without manual intervention. Instead of executing the default single-load case defined in the GUI, the new script is capable of reading a list of load combinations, updating the relevant parameters in `ProjectParameters.json`, and running Kratos simulations in sequence.

The script introduces a loop structure where, for each iteration, a new set of three loading values is generated randomly within a defined range. These values are injected into the configuration file, and the Kratos solver is triggered. This process is repeated for a definite number of iterations, each resulting in a unique simulation output. An example Python snippet demonstrating this automation is shown below:

```python
# Generate Random list of 'mu' values for parameterization
def generate_random_list_of_mu(num_sets=3, min_value=-300.0, max_value=400.0, size=3):
    """
    Generates a list of modulus value sets with two decimal precision.

    Args:
        num_sets (int): Number of sets to generate.
        min_value (float): Minimum value for random generation.
        max_value (float): Maximum value for random generation.
        size (int): Number of values in each set.

    Returns:
        list: A list containing `num_sets` sets of random modulus values.
    """
    random_list_of_mu = []
    for _ in range(num_sets):
        # Generate a random set of 'size' values within the given range
        random_set = [round(random.uniform(min_value, max_value), 2) for _ in range(size)]
        random_list_of_mu.append(random_set)
    return random_list_of_mu
```

Listing 1: Random Load Generation

Each simulation produces several outputs, including the system's mass matrix, stiffness matrix, displacement vector, and loading vector. These outputs are parsed and stored in a structured directory format under `Kratos_Results`, with subdirectories like `mass_results/`, `stiffness_results/`, `displacement_results/`, and `loading_results/`. To maintain a clean workspace, any auxiliary files or logs are moved into a `junk/` directory. This type of organization enhances data traceability and facilitates efficient post-processing during ROM training.

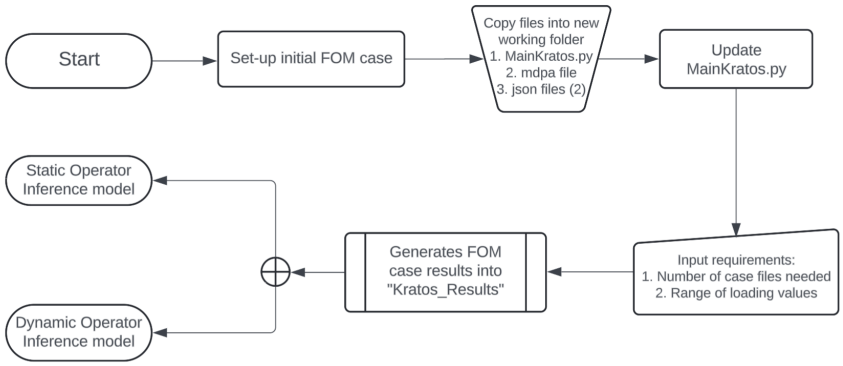


Figure 4: Automated pipeline for data generation using Kratos Multiphysics.

The following section explains the construction of reduced operators using Operator Inference techniques. This includes the projection of full-order mass, stiffness, and force matrices onto the reduced basis, formulation of the reduced equations of motion, and integration methods like Newmark-beta for dynamic cases. These procedures serve as the core of the ROM framework, transforming raw data into a simulation model.

4.2 Operator Inference Model - Static Linear Model

This section details the process of constructing a reduced-order model (ROM) for static structural systems using Operator Inference. The approach replaces the traditional projection of full-order operators with a data-driven method that learns the reduced operators directly from full-order simulation data.

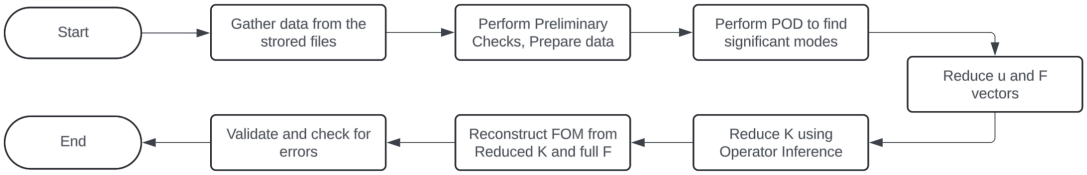


Figure 5: Overview of the static ROM development workflow using Operator Inference.

4.2.1 Initial Preparation

To begin the construction of the Operator Inference model for static problems, the model first accesses data written by the solver Kratos Multiphysics. These outputs, includ-

ing the stiffness matrix, displacement vectors, and force vectors, are organized into a well-structured directory (`Kratos_Results/`) and are important for building the reduced model.

The initial script performs preliminary checks to validate the data integrity. This includes confirming matrix sizes, loading shapes, and ensuring the displacement data is readable. The execution is carried out by running the `Linear_static_ROM.ipynb` file, which also generates the result folders.

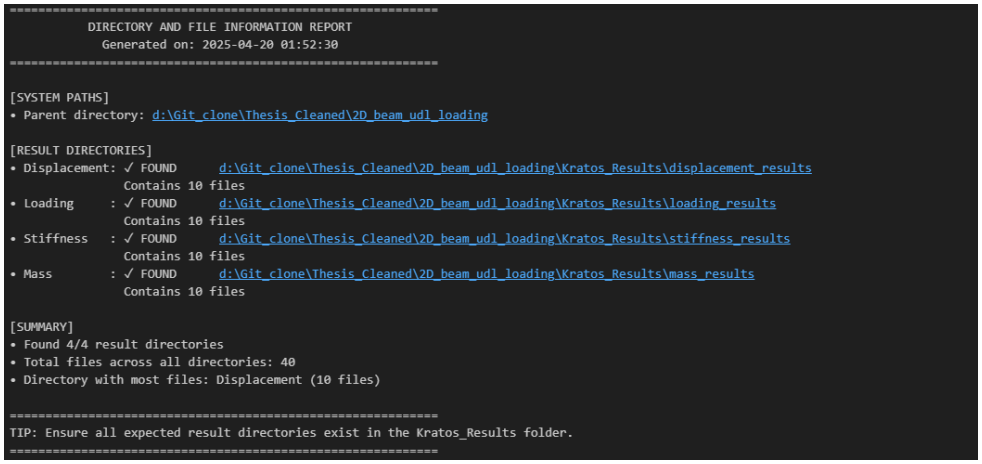


Figure 6: Initial Check for the essential data to perform ROM

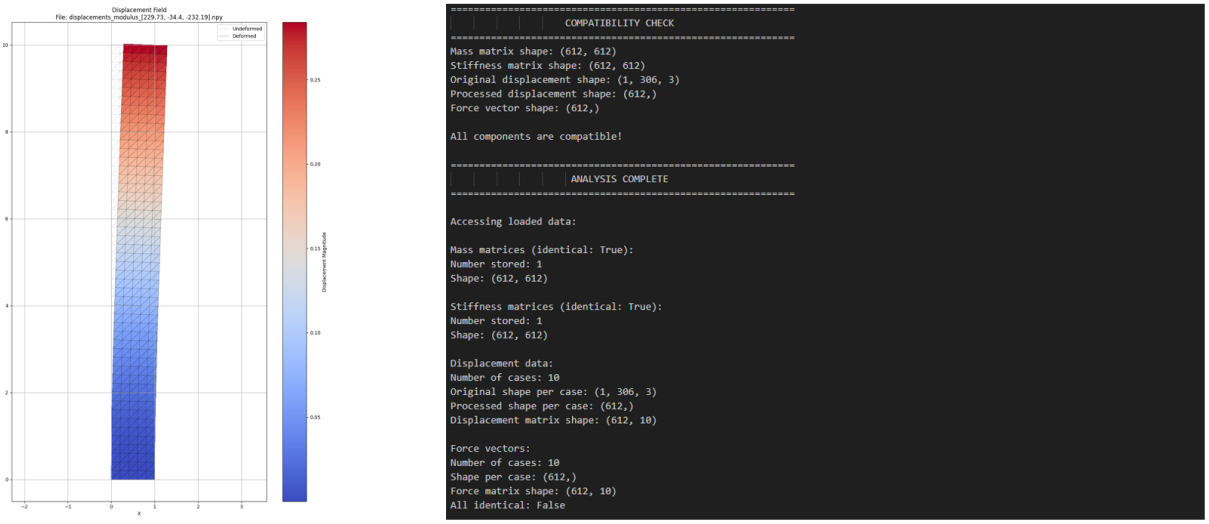
To ensure mesh consistency and the accuracy of simulation results, the mesh connectivity is also read from the `.mdpa` file. A random sample is picked from the displacement results, and a displacement contour is plotted as a verification step.

```
1  def execute_displacement_plot(displacement_folder, mdpa_file,
       scale_factor=2e5):
2      """
3      Function to execute the displacement plot without storing unnecessary
           variables.
4
5      Parameters:
6      -----------
7      displacement_folder: str, path to the folder containing displacement files
8      mdpa_file: str, path to the .mdpa file
9      scale_factor: float, scaling factor for displacements
10     """
11     try:
12         # Find all displacement files (those ending in .npy) in the given folder
13         displacement_files = [f for f in os.listdir(displacement_folder) if
               f.endswith(".npy")]
14
15         # Raise an error if no displacement files are found
16         if not displacement_files:
```

```
17            raise FileNotFoundError(f"No displacement files found in folder:
                  {displacement_folder}")
18
19        # Randomly select a displacement file from the folder
20        random_displacement_file = random.choice(displacement_files)
21        random_file_path = os.path.join(displacement_folder,
              random_displacement_file)
22
23        # Load the selected displacement file
24        displacements = np.load(random_file_path)
25
26        print(f"\nSelected displacement file: {random_displacement_file}")
27        print(f"Initial displacement data shape: {displacements.shape}")
28
29        # Process the displacement data (ensure correct shape for plotting)
30        displacements = np.squeeze(displacements) # Remove extra dimensions if
              any
31        print(f"Processed displacement shape: {displacements.shape}")
32
33        # Call the plot function to visualize the displacement field
34        plot_displacement_timestep(
35            x_data=displacements,
36            time_step=0,
37            mdpa_file=mdpa_file,
38            random_displacement_file=random_displacement_file,
39            scale_factor=scale_factor
40        )
41
42    except Exception as e:
43        print(f"\nError in execute_displacement_plot: {str(e)}")
44        import traceback
45        traceback.print_exc()
```

Listing 2: Code Snippet - Displacement Field Visualization

Additionally, the script ensures that the stiffness matrix and mass matrix which are used across different loading cases remain constant. This step is key as the Operator Inference technique assumes consistent full-order operators across all simulations.



(a) Sample Displacement Contour

(b) Preliminary Checks and data manipulation

Figure 7: Sample Displacement Contour and Preliminary checks

These preparatory steps ensure that only verified and clean data is passed onto the reduced model construction phase. This improves the reliability of the ROM training process.

4.2.2 ROM Construction

Singular Value Decomposition (SVD) is a critical step in constructing a reduced basis for ROMs. Given a matrix of full-order displacement snapshots $\mathbf{U} \in \mathbb{R}^{n \times m}$ (where n is the number of degrees of freedom and m the number of samples), we perform SVD as follows:

$$\mathbf{U} = \mathbf{\Phi} \mathbf{\Sigma} \mathbf{W}^T, \tag{1}$$

Here, $\mathbf{\Phi} \in \mathbb{R}^{n \times n}$ contains the left singular vectors (spatial modes), $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix of singular values, and $\mathbf{W} \in \mathbb{R}^{m \times m}$ holds the right singular vectors.

To construct the reduced-order model, only the first r columns of $\mathbf{\Phi}$ corresponding to the largest singular values are retained:

$$\mathbf{V} = \mathbf{\Phi}_r \in \mathbb{R}^{n \times r}, \tag{2}$$

This reduced basis captures the dominant behavior of the system which drastically reduces dimensionality. Typically, the number r is chosen such that a certain percentage of total energy (e.g., 99%) is retained.

Each full-order displacement $\mathbf{u}$ is now expressed in terms of the reduced coordinates:

$$\mathbf{u} \approx \mathbf{V} \tilde{\mathbf{u}}, \tag{3}$$

where $\tilde{\mathbf{u}} \in \mathbb{R}^r$ are the reduced displacement coefficients. This relation forms the backbone of ROMs and ensures faster predictions during inference.

4.2.3 Implementation Details

The next step is to formulate the reduced system using the observed force and displacement pairs from full-order simulations. The goal is to obtain a reduced stiffness matrix $\tilde{\mathbf{K}}$ using these pairs.

Data Projection: Project each full-order force vector $\mathbf{f}_i$ onto the reduced basis:

$$\tilde{\mathbf{f}}_i = \mathbf{V}^T \mathbf{f}_i \in \mathbb{R}^r. \tag{4}$$

This step ensures that the force vectors are expressed in the same reduced space as the displacements.

Operator Inference: The reduced system is constructed for each sample as:

$$\tilde{\mathbf{K}} \tilde{\mathbf{u}}_i = \tilde{\mathbf{f}}_i, \quad \text{for } i = 1, \ldots, m. \tag{5}$$

To determine $\tilde{\mathbf{K}}$, a least-squares fit is performed across all m samples:

$$\tilde{\mathbf{K}} = \tilde{\mathbf{F}} \tilde{\mathbf{U}}^{\dagger}, \tag{6}$$

where $\tilde{\mathbf{F}} = [\tilde{\mathbf{f}}_1, \ldots, \tilde{\mathbf{f}}_m]$ and $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_1, \ldots, \tilde{\mathbf{u}}_m]$.

Table 1: Reduced Order Model Properties and Projection Analysis

Reduced Order Model Properties and Projection Analysis

Matrix Properties	
Shape	(612, 10) (DOFs × snapshots)
Numerical Rank	3
Effective Rank (1e-10)	3
Reduced Basis Selection	
Energy Threshold	0.999 (99.9%)
SV Threshold	1.0e-19
Rank by Energy	3 (captures 100% energy)
Rank by SV Threshold	3
Final Selected Rank	3
Dimensionality Reduction	99.5% (612 → 3)
Reduced Basis Properties	
Shape	(612, 3) (DOFs × modes)
Reduced Order Projection Analysis	
u_r Norm	5.515e-05
Pseudo-Inverse Shape	(10, 3)
Programmatic Access to Results	
Reduced Basis Shape	(612, 3)
Projected Displacement Shape	(3, 10)
Projected Force Shape	(3, 10)
Pseudo-Inverse Shape	(10, 3)

Python Implementation: The following code block is used to compute $\tilde{\mathbf{K}}$ from collected simulation data:

```
    print("\n" + "="*80)
    print(" REDUCED STIFFNESS MATRIX COMPUTATION ".center(80))
    print("="*80)

    # Extract required matrices from reduced_results
    try:
        V_r = reduced_results['svd']['V_r']
        f_r = reduced_results['projected_force']
        u_r_pinv = reduced_results['pseudo_inverse']
    except KeyError as e:
        print(f"Error: Missing required matrix {e} in reduced_results")
        return None

    # 1. Compute reduced stiffness matrix
    K_r = f_r @ u_r_pinv
```

Listing 3: Code Snippet - Construction of Reduced Stiffness Matrix $\tilde{\mathbf{K}}$

Reconstruction: Finally, recover the full-order displacement using:

$$\mathbf{u}_{\text{new}} \approx \mathbf{V}\tilde{\mathbf{u}}_{\text{new}}. \tag{7}$$

This ensures fast and memory-efficient prediction while retaining suitable accuracy. The code blocks provided automatic critical steps: data loading, matrix verification, reduced operator construction, and online inference.

4.3 Operator Inference Model - Dynamic Linear Model

4.3.1 Initial Preparation

The implementation of the Operator Inference Model for dynamic linear systems begins with a robust and structured data preparation phase. This stage marks the beginning for the construction of an accurate and efficient Reduced Order Model (ROM). As depicted in Figure 8, the initial preparation can be categorized into three key processes: data acquisition, preprocessing, and configuration for model reduction.

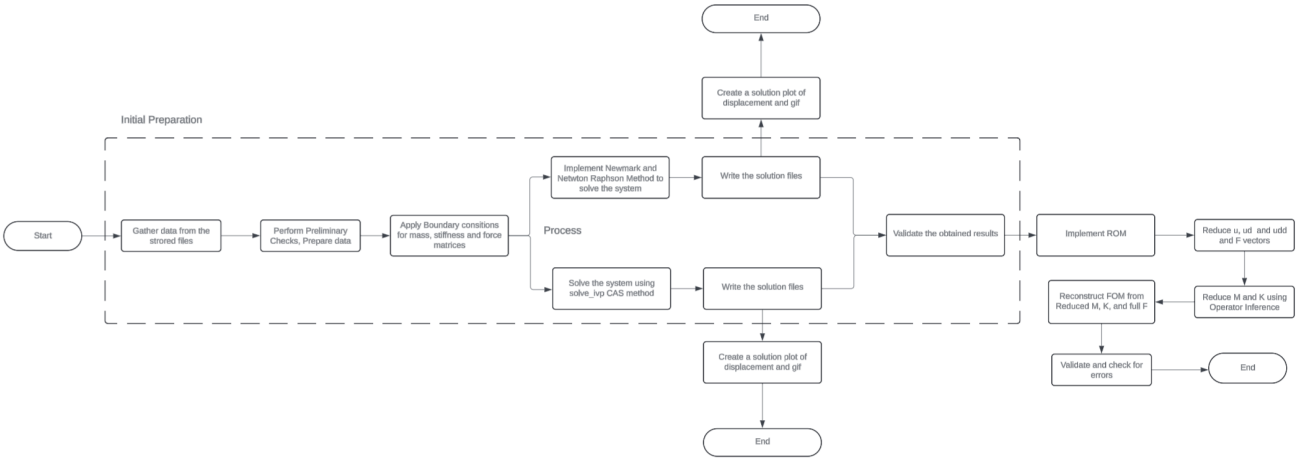


Figure 8: Workflow showing data preparation and preprocessing for dynamic ROM construction.

1. Data Acquisition:

The first step involves collecting all simulation results and metadata generated by Kratos Multiphysics. This includes files containing mass matrices, stiffness matrices, force vectors, displacement fields, and simulation time steps. These datasets are automatically stored in designated directories during batch simulation runs and are essential for performing system identification. At this stage, scripts are configured to obtain and organize this data for subsequent analysis.

Table 2: Model Load Summary and Compatibility Check

Model Load Summary

```
10 files successfully read and processed.
All magnitudes are identical and saved under 'magnitudes'.
======================== FINAL COMPATIBILITY CHECK ========================
Mass matrix shape: (612, 612)
Stiffness matrix shape: (612, 612)
2D Displacement vector shape: (612, 10)
Force vector shape: (612, 10)
All components have compatible dimensions!
Available components:
M: Shape (612, 612)
K: Shape (612, 612)
x_original: Shape (612, 10)
x: Shape (612, 10) (Processed)
f: Shape (612, 10)
is_compatible: True
loaded_successfully: True
compatibility_checked: True
Matrices successfully loaded and compatibility checked!
```

2. Preprocessing and Integrity Checks:
The collected data is subjected to an integrity check to verify completeness and consistency. This includes:

- Verifying that all matrices have the correct dimensions.
- Ensuring that matrix entries are non-singular and numerically stable.
- Validating that each case has a corresponding force-displacement set.

Once validated, data arrays are filtered to exclude constrained degrees of freedom (DOFs), i.e., those affected by Dirichlet boundary conditions. This step ensures that the ROM construction focuses only on physically meaningful, unconstrained motion, which prevents singularity issues and enhances numerical stability.

Table 3: Load Magnitude Summary for All Cases

Load Magnitude Summary

Case (F1, F2, F3)	Load 1 (Y)	Load 2 (X)	Load 3 (X)
1: (-137.59, -63.13, 362.93)	-137.59	-63.13	362.93
2: (-198.79, 228.5, -134.8)	-198.79	228.5	-134.8
3: (104.64, -25.63, 160.89)	104.64	-25.63	160.89
4: (17.45, -13.32, 69.69)	17.45	-13.32	69.69
5: (186.02, 35.84, -213.65)	186.02	35.84	-213.65
6: (229.73, -34.4, -232.19)	229.73	-34.4	-232.19
7: (34.88, -235.52, 250.55)	34.88	-235.52	250.55
8: (379.97, -275.91, -44.58)	379.97	-275.91	-44.58
9: (391.22, -286.75, 386.86)	391.22	-286.75	386.86
10: (75.24, 90.42, 11.42)	75.24	90.42	11.42

3. Model Setup and Configuration:
At this stage, the filtered data is formatted and prepared into consistent matrix structures readily available for solving the Full Order Model (FOM). The goal is to compute reliable time-dependent solutions under dynamic loading, using both numerical integration schemes for accuracy. All computed results such as displacement, velocity, and acceleration vectors are saved case-wise for further use.

To assess the numerical accuracy of the Newmark integration method with Newton-Raphson iterations, the same constrained system matrices are also subjected to numerical integration using SciPy's `solve_ivp` method. The displacement, velocity, and acceleration results obtained from both solvers are compared over time to verify the consistency of the solution and identify any problems in numerical behavior.

This step also ensures that a validated dataset is created from FOM simulations, which later becomes the basis for constructing and training the Reduced Order Model (ROM).

Boundary conditions are explicitly enforced on the global matrices $\mathbf{M}$, $\mathbf{K}$, and $\mathbf{F}$ by penalizing rows and columns corresponding to fixed DOFs. This process transforms the

full-order system into a constrained system ready for dynamic solving:

$$\mathbf{M}_{\text{constrained}}, \quad \mathbf{K}_{\text{constrained}}, \quad \mathbf{f}_{\text{constrained}}$$

Following this, the constrained system is solved using automation pipelines that implement either:

- Newton-Raphson-based Newmark integration for nonlinear time-stepping.
- SciPy's `solve_ivp` for first-order reformulated ODE systems.

These solutions are post-processed to extract force-equilibrium residuals, energy balance behavior, and time evolution of response variables. The resulting validated data ensures full consistency before ROM projection and inference are initiated.

4.3.2 Implementation Details

The implementation of the dynamic Reduced Order Model (ROM) involves integrating two prominent numerical schemes: the Newmark-beta method and the state-variable-based formulation using SciPy's `solve_ivp`. These solvers operate on reduced system matrices generated through projection of the full-order operators and ensure accurate time-dependent predictions. This section is divided into three parts.

1. Newmark-beta Method

The Newmark-beta method is an implicit time integration technique widely used in structural dynamics for second-order systems. It is known for its controllable numerical damping and unconditionally stable behavior on parameterization [12].

We begin from the semi-discrete dynamic equation:

$$\mathbf{M}_r \ddot{\mathbf{u}} + \mathbf{K}_r \mathbf{u} = \mathbf{f}_r(t)$$

The Newmark integration scheme assumes:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t\, \dot{\mathbf{u}}_n + \Delta t^2 \left((1-2\beta)/2\right) \ddot{\mathbf{u}}_n + \Delta t^2 \beta \ddot{\mathbf{u}}_{n+1}$$

$$\dot{\mathbf{u}}_{n+1} = \dot{\mathbf{u}}_n + \Delta t \left((1-\gamma)\ddot{\mathbf{u}}_n + \gamma \ddot{\mathbf{u}}_{n+1}\right)$$

These are substituted into the equilibrium equation and solved iteratively using the Newton-Raphson method. The residual is defined as:

$$\boldsymbol{\psi} = \mathbf{f}_{n+1} - \mathbf{K}_r \mathbf{u}_{n+1} - \mathbf{M}_r \ddot{\mathbf{u}}_{n+1}$$

The consistent tangent matrix becomes:

$$\mathbf{K}_{dyn} = \mathbf{K}_r + \frac{1}{\beta \Delta t^2} \mathbf{M}_r$$

This linearized system is solved at each time step until convergence is achieved. After convergence, updated values of $\mathbf{u}_{n+1}, \dot{\mathbf{u}}_{n+1}, \ddot{\mathbf{u}}_{n+1}$ are computed.

Algorithm 1: Multi-case dynamic analysis using Newmark-β method

Input: Mass matrix M, stiffness matrix K, force matrix f, list of magnitudes, initial displacement matrix x, total time T, time step Δt, tolerance tol, maximum iterations per step max_iter

Output: Displacement, velocity, acceleration, force response for each case, and convergence statistics

```
Construct output directory for saving results;
n_cases ← number of columns in f (each column is a case);
n_DOF ← number of rows in M;
for each case i = 1 to n_cases do
    Create case-specific folder for output;
    Extract f_i ← f[:, i];
    Extract initial displacement x_0 ← x[:, i];
    Initialize: x, ẋ, ẍ arrays of zeros;
    Set initial conditions: x[0] ← x_0;
    Compute effective stiffness matrix: K_eff = K + c_1 M;
    for each time step t_k in [0, T] with step Δt do
        Predict x^(0) using previous step values;
        for iteration j = 1 to max_iter do
            Compute effective force F_eff;
            Compute residual R = F_eff − K_eff x^(j);
            Solve for update Δx: K_eff Δx = R;
            Update solution: x^(j+1) ← x^(j) + Δx;
            if ||Δx|| < tol then
                Mark step as converged and break;
        Update acceleration and velocity using Newmark formulas;
        Save results for current step;
    Compute total internal force: F = Mẍ + Kx;
    Save results and convergence summary for current case;
Print summary statistics (max displacement, solve time, convergence);
return All case results and performance data;
```

2. SciPy `solve_ivp` Using State Variable Method

To use SciPy's `solve_ivp`, the second-order ODE is rewritten into a first-order system using state variables. Let:

$$\mathbf{U}(t) = \begin{bmatrix} \dot{\mathbf{u}}(t) \\ \mathbf{u}(t) \end{bmatrix} \in \mathbb{R}^{2r}$$

Then, the system of equations becomes:

$$\frac{d}{dt}\mathbf{U}(t) = \mathbf{M}^{\star -1}\left(\mathbf{F}^{\star} - \mathbf{K}^{\star}\mathbf{U}(t)\right)$$

Where:

$$\mathbf{M}^{\star} = \begin{bmatrix} \mathbf{M}_r & 0 \\ 0 & \mathbf{I} \end{bmatrix}, \quad \mathbf{K}^{\star} = \begin{bmatrix} 0 & \mathbf{K}_r \\ -\mathbf{I} & 0 \end{bmatrix}, \quad \mathbf{F}^{\star} = \begin{bmatrix} \mathbf{f}_r(t) \\ 0 \end{bmatrix}$$

The function `dUdt(t, U)` computes this right-hand side, and `solve_ivp` is called using the Radau method for stiff ODEs.

Algorithm 2: Analytical multi-case dynamic solver using state-space formulation and `solve_ivp`

Input: Mass matrix M, stiffness matrix K, force matrix f, magnitudes list, initial displacements x, time range $[0, T]$, time step Δt

Output: Time-dependent displacement, velocity, acceleration, and internal force for each case

Initialize: $s \leftarrow$ number of DOFs, $n_{\text{cases}} \leftarrow$ number of force columns;

Set initial velocity $v_0 = 0$, build identity I and zero Z matrices of size s;

Construct block matrices:

$$M^* = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}, \quad K^* = \begin{bmatrix} 0 & K \\ -I & 0 \end{bmatrix}$$

Compute $M^*_{\text{inv}} \leftarrow (M^*)^{-1}$;

for each case $i = 1$ to n_{cases} do

- Prepare output directory for current case;
- Define initial state vector $U_0 = \begin{bmatrix} v_0^{(i)} \\ x_0^{(i)} \end{bmatrix}$;
- Construct extended force vector $F^* = \begin{bmatrix} f^{(i)} \\ 0 \end{bmatrix}$;
- Solve the ODE system:

$$\frac{dU}{dt} = M^*_{\text{inv}}(F^* - K^*U)$$

  using method `Radau` with `solve_ivp`;
- Extract velocity $v(t)$ and displacement $x(t)$ from solution $U(t)$;
- Compute acceleration $a(t)$ using finite difference;
- Compute internal force $f_{\text{int}}(t) = Ma(t) + Kx(t)$;
- Save all solution arrays for post-processing;
- Store summary statistics (max values, time taken);

Print summary table for all cases with maximum responses and solve times;

return Displacement, velocity, acceleration, internal force for each case

3. ROM Implementation via Operator Inference

This section presents the implementation of a Reduced-Order Model (ROM) using the Operator Inference approach. The goal is to reduce computational cost while retaining the essential dynamic behavior of the original system. This is achieved by inferring reduced mass and stiffness matrices from a full-order simulation and using them to compute dynamic responses efficiently.

1. Data Extraction from Full-Order Simulation

The full-order model is solved using the Newmark-β time integration scheme combined with Newton-Raphson iterations for handling nonlinearities. Once the simulation completes, the following time-series data are extracted:

- Displacement matrix U
- Velocity matrix $\dot{U}$
- Acceleration matrix $\ddot{U}$
- Force matrix F

These matrices capture the dynamic evolution of the system at each time step.

2. Proper Orthogonal Decomposition (POD)

Proper Orthogonal Decomposition (POD) is applied to the full-order displacement matrix U to extract its most significant modes. The POD is computed via Singular Value Decomposition (SVD):

$$U = V\Sigma W^T$$

From the left singular vectors V, the most energetic modes are selected to form the reduced basis $V_r \in \mathbb{R}^{n\times r}$, where $r \ll n$.

3. Projection to Reduced Space

Using the reduced basis V_r, the full-order matrices are projected to the reduced space:

$$\begin{aligned} q_r &= V_r^T U \\ \dot{q}_r &= V_r^T \dot{U} \\ \ddot{q}_r &= V_r^T \ddot{U} \\ F_r &= V_r^T F \end{aligned}$$

These reduced variables represent the system's dynamics in the lower-dimensional subspace.

4. Operator Inference to Estimate M_r and K_r

4.1 Formulating the Least-Squares Problem

The reduced mass matrix M_r and stiffness matrix K_r are obtained by solving a least-squares minimization problem based on the reduced displacement, acceleration, and force data. The objective is to minimize the residual between the projected force and the linear combination of projected acceleration and displacement:

$$\min_{M_r, K_r} \|F_r - M_r \ddot{q}_r - K_r q_r\|_2^2$$

Initially, no constraints are enforced on the structure of M_r and K_r, leading to a straightforward linear regression problem. However, to maintain the physical properties of the system, additional symmetry constraints are applied as described below.

4.2 Enforcing Symmetry in Reduced Matrices

To preserve the physical structure and realism of the reduced-order model, it is crucial that both M_r and K_r are symmetric. The enforcement of symmetry is achieved through a specific parameterization and reconstruction strategy as follows:

a) Symmetric Parameterization Instead of estimating every entry of the reduced matrices individually, only the upper-triangular (including diagonal) components are inferred. For a reduced matrix of size $r \times r$, the number of independent variables is:

$$n_{\text{sym}} = \frac{r(r+1)}{2}$$

Thus, the least-squares problem is reformulated such that only these independent entries are solved for.

b) Construction of the System of Equations The time-series data for the reduced displacement q_r, acceleration $\ddot{q}_r$, and force F_r are used to construct an overdetermined system:

$$Ax = b$$

where:

- A is the system matrix, built from contributions of $\ddot{q}_r$ and q_r, structured to match the symmetric parameterization,
- x is the unknown vector containing the upper-triangular entries of M_r and K_r,
- b is the vectorized form of the reduced force data F_r.

For each time step and mode, the contributions of the acceleration and displacement are added only to the corresponding symmetric indices.

c) Least-Squares Solution The overdetermined system $Ax = b$ is solved using standard least-squares minimization:

$$x^\star = \arg\min_x \|Ax - b\|_2^2$$

yielding the best-fitting independent entries for M_r and K_r.

d) Reconstruction of Full Symmetric Matrices Once the vector $x^\star$ is obtained, it is partitioned into two subvectors:

- $\text{vec}(M_r)$, corresponding to the independent entries of M_r,
- $\text{vec}(K_r)$, corresponding to the independent entries of K_r.

The full symmetric matrices are then reconstructed by filling the upper triangular parts from the inferred vectors and mirroring them across the diagonal:

$$M_r(i,j) = M_r(j,i), \quad K_r(i,j) = K_r(j,i) \quad \forall\, i,j$$

e) Verification of Matrix Properties Finally, the reconstructed matrices are checked to ensure:

- Symmetry,
- Real-valued entries,
- (Optionally) positive eigenvalues for stability,
- Well-conditioned behavior.

This guarantees that the inferred reduced-order model maintains the essential physical characteristics of the original full-order system.

4.3 Importance of Enforcing Symmetry in Reduced Matrices

Enforcing symmetry in the reduced mass matrix M_r and stiffness matrix K_r is essential for preserving the mathematical structure and physical realism of the reduced-order model [3, 9, 2]. Several key reasons underline the necessity of maintaining symmetry:

a) Preservation of Physical Properties: In the full-order model, the mass matrix M and stiffness matrix K are naturally symmetric for conservative mechanical systems. Symmetry reflects fundamental physical properties such as:

- Conservation of energy: In undamped linear systems, total mechanical energy is conserved during oscillations. Symmetry ensures that the force-displacement and acceleration-energy relationships remain consistent.

- Reciprocity: Symmetric matrices ensure that the system obeys Maxwell's reciprocity theorem, a foundational principle in mechanics and structural dynamics.

Breaking symmetry would result in non-physical behaviors, such as artificial energy creation or loss, leading to instability or unphysical responses.

b) Mathematical Stability and Robustness: Symmetric matrices have favorable mathematical properties:

- Their eigenvalues are real, ensuring that the natural frequencies of the system are real-valued.
- They are generally better conditioned, leading to more numerically stable simulations.
- They allow the use of efficient and robust numerical solvers that exploit symmetry, reducing computational cost.

If the reduced matrices were not symmetric, the eigenvalues could become complex, introducing spurious oscillations or growth/decay behavior that does not exist in the true system.

c) Consistency with Modal Decomposition Modal analysis, which decomposes the system into independent vibrational modes, assumes that M and K are symmetric and positive-definite. A reduced-order model intended to replicate modal behavior must preserve this structure to allow accurate prediction of mode shapes and natural frequencies.

Symmetric matrices guarantee that the reduced system admits an orthogonal modal basis, facilitating modal superposition and simplifying dynamic analysis.

d) Improved Generalization and Accuracy In operator inference, training on time-series data leads to a best-fit reduced system. Without enforcing symmetry, the least-squares optimization could produce slightly asymmetric matrices due to noise or overfitting artifacts. These asymmetries can deteriorate the model's ability to generalize to unseen conditions and time intervals.

By constraining symmetry explicitly, the reduced-order model is "regularized" toward more physically plausible and smoother dynamics, improving accuracy and predictive capability.

e) Avoidance of Artificial Damping or Instabilities In dynamic simulations, even small asymmetries in M_r or K_r can act as numerical sources of artificial damping or instability. This can corrupt transient behavior, cause oscillation amplitudes to grow or decay unnaturally, or make time-integration schemes (like Newmark-β) unstable even if they are theoretically unconditionally stable for symmetric matrices.

Symmetry enforcement eliminates these risks, ensuring that the reduced model remains faithful to the underlying dynamics.

In conclusion, enforcing symmetry in reduced matrices is not merely a mathematical convenience but a fundamental requirement to guarantee that the reduced-order model accurately reproduces the physical, energetic, and dynamic behavior of the original system. Without symmetry, the ROM could exhibit incorrect frequencies, instabilities, and non-physical results, defeating the purpose of model reduction.

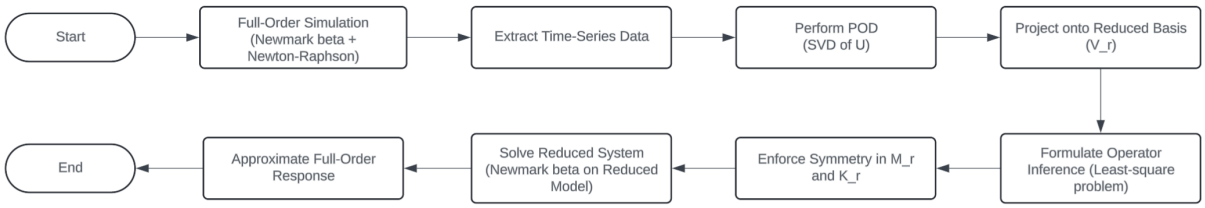


Figure 9: Workflow for Reduced-Order Modeling Using Operator Inference Approach

5. Solving the Reduced System

Once M_r and K_r are computed, the reduced dynamic system is solved using the same Newmark-β method:

$$M_r \ddot{q}_r + K_r q_r = F_r$$

The reduced system is significantly smaller in size, leading to faster computation while retaining essential system dynamics.

6. Lifting Back to Full Order

After computing the reduced displacements q_r, velocities, and accelerations, the full-order approximations are recovered by projecting back:

$$U \approx V_r q_r$$

This allows for an accurate approximation of the original full-order response with reduced computational effort.

7. Standard Galerkin Projection-Based Reduced Model

In addition to the operator inference method, a second reduced-order model (ROM) is constructed using the standard Galerkin projection technique. This serves as a baseline to validate and compare the performance of the operator-inferred reduced model.

a) Construction of Reduced Matrices The reduced mass and stiffness matrices, denoted as M_r^{std} and K_r^{std}, are obtained through orthogonal projection onto the reduced basis V_r:

$$M_r^{\text{std}} = V_r^T M V_r$$
$$K_r^{\text{std}} = V_r^T K V_r$$

This projection preserves symmetry naturally, as it leverages the symmetric properties of the full-order matrices.

b) Validation of Matrix Properties To ensure physical and numerical correctness, the following properties of the reduced matrices are verified:

- Symmetry,
- Real-valued entries,
- Positive definiteness (if applicable),
- Real eigenvalues,
- Reasonable condition numbers.

This step ensures that the standard-projected reduced matrices behave consistently with expected mechanical principles.

c) Solving the Reduced System The reduced dynamic system is then solved using the Newmark-β time integration scheme:

$$M_r^{\text{std}} \ddot{q}_r + K_r^{\text{std}} q_r = F_r$$

where F_r is the reduced force vector and q_r is the reduced displacement.

d) Reconstruction of Full-Order Response Following the solution, the full-order response is approximated by lifting back into the original space:

$$U_{\text{std}} \approx V_r q_r$$

This reconstructed displacement U_{std} enables direct comparison with both the operator-inferred solution and the full-order solution.

e) Comparison with Operator Inference Results Finally, the standard Galerkin-projected ROM results are compared against those obtained through operator inference. Key aspects compared include:

- Accuracy of the reconstructed displacement,
- Differences in dynamic response,
- Stability and eigenvalue behavior,
- Computational efficiency.

This comparative analysis provides insight into the benefits and trade-offs of each reduced modeling approach.

Summary

This framework combines projection-based model reduction with data-driven operator inference techniques to efficiently capture the dynamic behavior of complex systems. Two approaches are implemented for constructing reduced-order models (ROMs) specifically for dynamic analysis: an operator inference method using least-squares fitting with enforced symmetry, and a standard Galerkin projection method. Both methods employ Proper Orthogonal Decomposition (POD) to extract dominant modes, substantially reducing computational cost without compromising dynamic fidelity. Symmetry enforcement is critical to ensure stability, real eigenvalues, and physical consistency in the operator-inferred reduced matrices. Comparative analysis is performed between the operator-inferred ROM and the standard projection-based ROM, focusing exclusively on the dynamic problem. The results demonstrate that when the essential system dynamics can be captured by a small number of basis vectors, accurate and efficient simulations are achievable, preserving the fundamental characteristics of the full-order model.

5 Results and Discussion

This section presents a detailed evaluation of the simulation results obtained from both static and dynamic structural models. The objective is to assess the accuracy, efficiency, and robustness of the Reduced Order Modeling (ROM) strategies in comparison to the Full Order Models (FOMs).

The study considers a simple cantilever beam with dimensions of 1 unit in width and 10 units in height. The geometry is discretized using triangular mesh elements of size 0.2, resulting in a model comprising 306 nodes and 500 elements. The setup is implemented using the `Kratos Multiphysics` solver. Loading conditions are applied as illustrated in the corresponding figure 10.

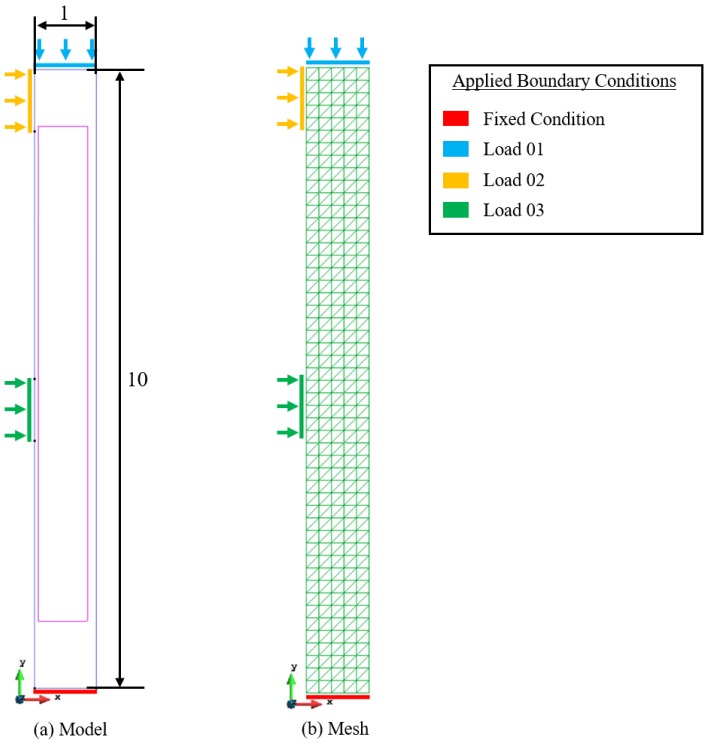


Figure 10: Geometry and loading conditions of the cantilever beam model.

Once the model is defined, the necessary simulation files are generated on disk. The `MainKratos.py` script is then modified to parameterize the loading conditions, enabling automated case generation and clean data extraction. To validate the setup, a preliminary simulation is executed, and displacement results are visualized for a randomly selected case.

For this study, 10 parameterized cases were generated and solved to evaluate the performance of the ROM across varied loading conditions. These test cases form the foundation for analyzing the effectiveness of the reduced-order approach presented in the following sections.

5.1 Operator Inference Model - Static Linear Model

The following results were generated by executing the code from the `Linear_static_ROM.ipynb` notebook. The corresponding discussions are structured across the subsequent sections: Implementation and ROM Model Performance Check.

5.1.1 Implementation

Upon execution, the script performs an initial verification to ensure that all essential data files are present in the designated directory. A summary of the detected files and their statuses is then displayed to the user.

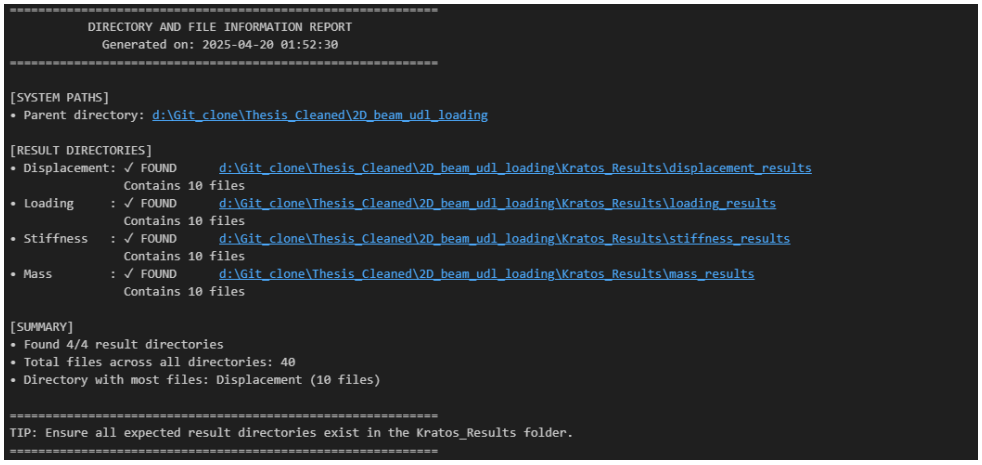
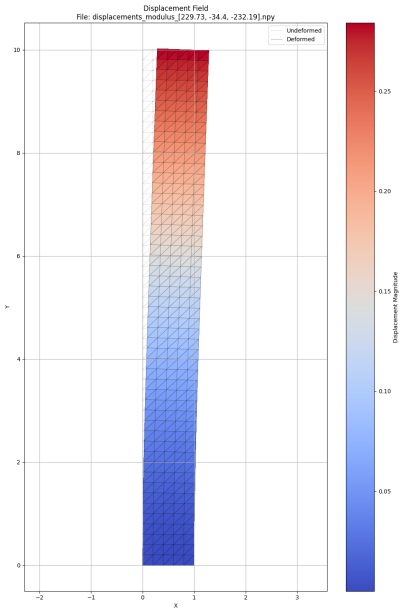


Figure 11: Initial Check for the essential data to perform ROM

And to check the validity of the written information, a file is randomly selected, and a displacement contour is generated as shown in the figure 12.

Once the required data is available, it undergoes preprocessing and organization for subsequent Proper Orthogonal Decomposition (POD) analysis. The objective of POD is to identify the most significant modes that capture the dominant dynamics of the system. For this purpose, two thresholds are applied: an energy capture criterion of 99.9% and a singular value cutoff threshold of 1×10^{-19}. The singular value decomposition (SVD) is then performed on the displacement snapshot matrix, and only the dominant modes satisfying these thresholds are retained.

In the test case considered, the number of significant mode shapes satisfying the condition was found to be three. Further, the full-order matrices for displacement and force, originally of size $(612, n)$, were reduced to $(612, 3)$, where 3 denotes the number of retained modes. This reduction in dimensionality significantly decreases computational complexity while maintaining physical accuracy. A detailed summary of the reduction and projection process is presented in Figure 13.



Output Block

Selected File:
`displacements_modulus_[229.73, -34.4, -232.19].npy`

Displacement Shape:
Initial: `(1, 306, 3)`
Processed: `(306, 3)`

Mesh Info:
Nodes: `306`
Elements: `500`
Data shape: `(306, 3)`

Displacement Stats:
Scale factor: `2.0e+05`
Max: `2.87e-01`
Min: `0.00e+00`
Mean: `1.20e-01`

Figure 12: Sample Displacement Contour

Table 4: Reduced Order Model Properties and Projection Analysis

Reduced Order Model Properties and Projection Analysis

Matrix Properties	
Shape	(612, 10) (DOFs × snapshots)
Numerical Rank	3
Effective Rank (1e-10)	3
Reduced Basis Selection	
Energy Threshold	0.999 (99.9%)
SV Threshold	1.0e-19
Rank by Energy	3 (captures 100% energy)
Rank by SV Threshold	3
Final Selected Rank	3
Dimensionality Reduction	99.5% (612 → 3)
Reduced Basis Properties	
Shape	(612, 3) (DOFs × modes)
Reduced Order Projection Analysis	
u_r Norm	5.515e-05
Pseudo-Inverse Shape	(10, 3)
Programmatic Access to Results	
Reduced Basis Shape	(612, 3)
Projected Displacement Shape	(3, 10)
Projected Force Shape	(3, 10)
Pseudo-Inverse Shape	(10, 3)

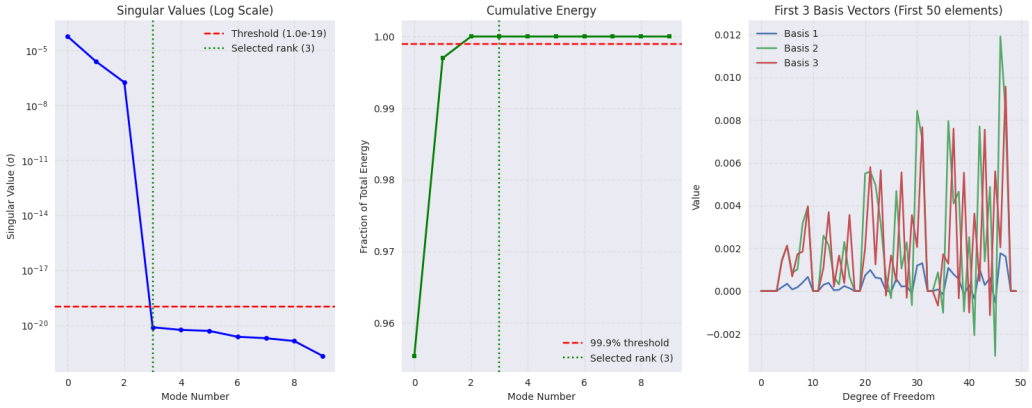


Figure 13: Dimensionality reduction using POD

5.1.2 ROM Model Performance Check

After obtaining the reduced displacement and force vectors, the reduced stiffness matrix is constructed using the Operator Inference (OpInf) method. To evaluate the effectiveness and integrity of this reduction, a verification procedure is carried out using the available Full Order Model (FOM) data. Specifically, the original FOM stiffness matrix is reduced using the previously extracted modal basis, the system is then solved in the reduced space, and the FOM matrix is reconstructed from the ROM output.

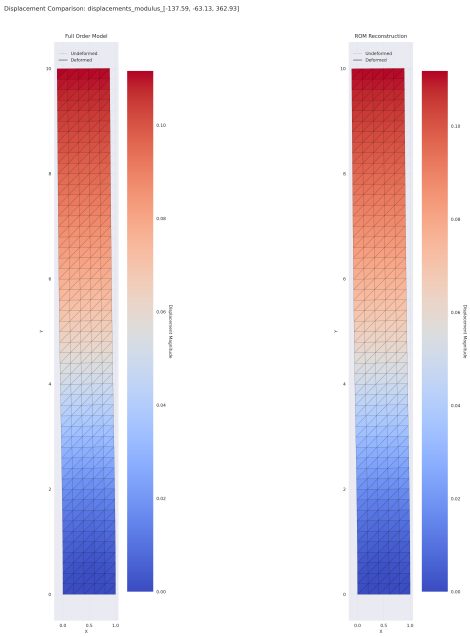
This reconstructed FOM matrix is then compared to the original to evaluate the reconstruction accuracy. A tolerance of 1×10^{-5} is imposed to check for deviation in results. The analysis shows that the full-order matrix is accurately recovered with negligible error, demonstrating the reversibility of the reduction-reconstruction pipeline. The reconstruction outcome and associated error metrics are summarized in Table 5.

Table 5: Final Summary – ROM vs FOM Displacement Comparison

Results Summary -- ROM vs FOM Displacement Comparison

Magnitude (F1, F2, F3)	Status	Accuracy	Global Error (%)
(-137.59, -63.13, 362.93)	Success	100.00%	0.45%
(-198.79, 228.5, -134.8)	Success	100.00%	0.07%
(104.64, -25.63, 160.89)	Success	100.00%	0.32%
(17.45, -13.32, 69.69)	Success	100.00%	0.18%
(186.02, 35.84, -213.65)	Success	100.00%	0.17%
(229.73, -34.4, -232.19)	Success	100.00%	0.23%
(34.88, -235.52, 250.55)	Success	100.00%	0.05%
(379.97, -275.91, -44.58)	Success	100.00%	0.20%
(391.22, -286.75, 386.86)	Success	100.00%	0.40%
(75.24, 90.42, 11.42)	Success	100.00%	0.22%

The figures below show the comparison of displacement contours between FOM and ROM (only a limited number of cases are shown here).

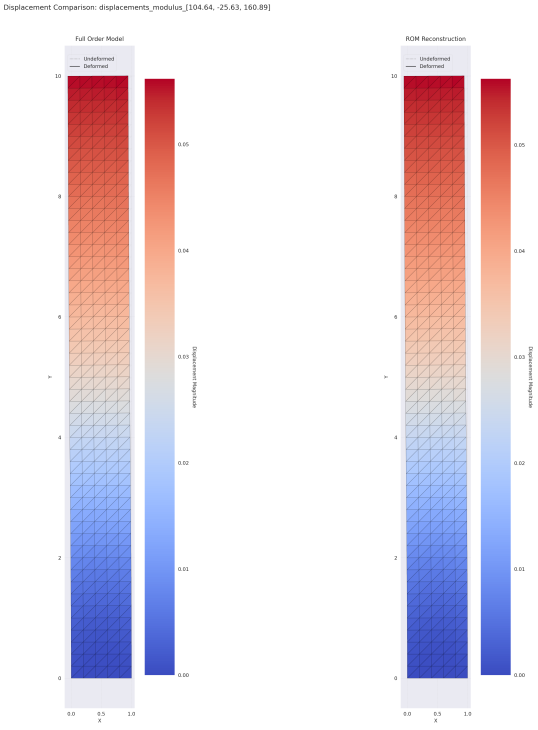


Output Block

Processing Case:
`displacements_modulus_[-137.59, -63.13, 362.93]`

Comparison Metrics:
Max difference: `3.71e-04`
Mean difference: `1.73e-04`
Relative error: `0.29%`

Figure 14: Static Operator Inference - Case 01

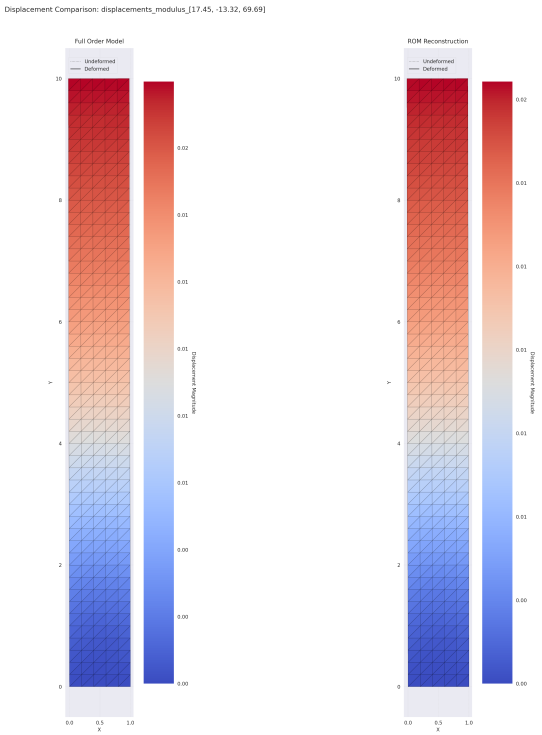


Output Block

Processing Case:
`displacements_modulus_[104.64, -25.63, 160.89]`

Comparison Metrics:
Max difference: `7.58e-05`
Mean difference: `4.14e-05`
Relative error: `0.14%`

Figure 15: Static Operator Inference - Case 02

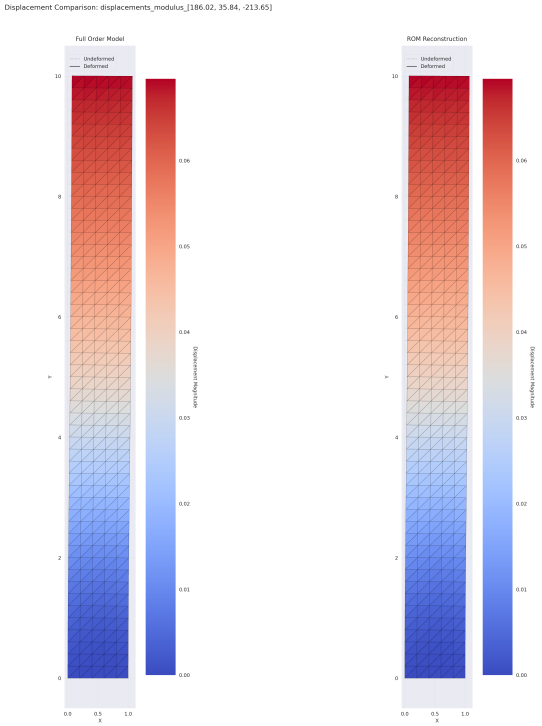


Output Block

Processing Case:
`displacements_modulus_[17.45, -13.32, 69.69]`

Comparison Metrics:
Max difference: `7.68e-05`
Mean difference: `3.81e-05`
Relative error: `0.38%`

Figure 16: Static Operator Inference - Case 03

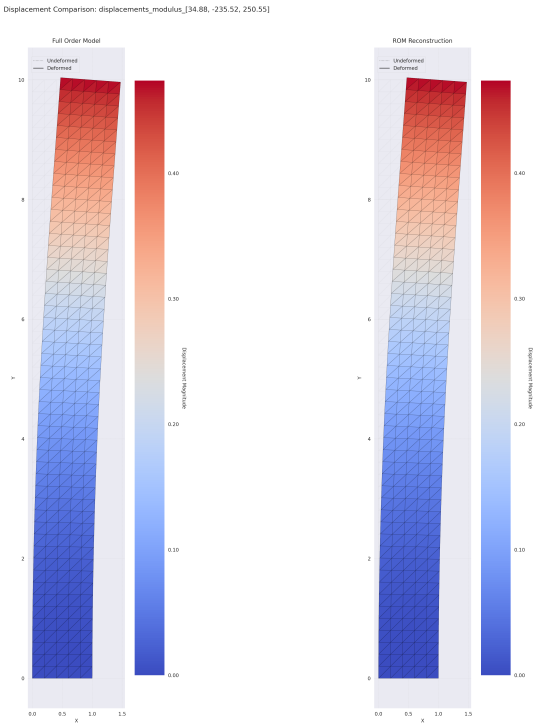


Output Block

Processing Case:
`displacements_modulus_[186.02, 35.84, -213.65]`

Comparison Metrics:
Max difference: `1.43e-04`
Mean difference: `7.32e-05`
Relative error: `0.20%`

Figure 17: Static Operator Inference - Case 04



Output Block

Processing Case:
`displacements_modulus_[34.88, -235.52, 250.55]`

Comparison Metrics:
Max difference: `3.59e-04`
Mean difference: `1.69e-04`
Relative error: `0.10%`

Figure 18: Static Operator Inference - Case 05

On the left side of each figure, the FOM displacement contour is shown, representing the reference solution. On the right side, the ROM reconstructed displacement contour is plotted, illustrating the solution obtained from the reduced system using the Operator Inference method.

Additionally, an output block is included alongside each figure summarizing important quantitative comparison metrics:

- Max Difference: The maximum pointwise difference between FOM and ROM solutions.
- Mean Difference: The average pointwise difference across the domain.
- Relative Error: The normalized mean error, expressed as a percentage relative to the FOM solution magnitude.

The visual comparison shows excellent agreement between the FOM and ROM results, with color maps almost identical across all cases. The low values of maximum and mean differences, and the relative errors remaining well below 0.5% for all cases, confirm the accuracy and reliability of the ROM reconstruction. This highlights the effectiveness of the Operator Inference approach in capturing the essential structural behavior with a significantly reduced computational model.

5.2 Operator Inference Model - Dynamic Linear Model

The following results were generated by executing the implementation found in the `Linear_dynamic_ROM.ipynb` notebook. The outcomes and interpretations are structured under three sections: Implementation, Validation, and ROM Model Performance Check. These sections collectively provide an in-depth assessment of the dynamic linear model and its reduced-order formulation using Operator Inference.

5.2.1 Implementation

Data Reading and Preprocessing

To begin the implementation, the initial simulation data is read from the output generated by the KratosMultiphysics solver. This data, stored in the `Kratos_Results` directory, includes time-dependent snapshots of displacement, velocity, acceleration, and force (see Table 6).

By default, the displacement data is written assuming three degrees of freedom (DOF) per node, even when the actual model may be in two or one dimension. As a result, a preprocessing step is required to extract and organize the data corresponding only to the active DOFs relevant to the specific model under study.

Once the preprocessing is completed, the data is structured into matrices suitable for further analysis. Preliminary validation is performed at this stage to confirm data integrity, dimensional consistency, and readiness for reduced-order modeling. These consistency checks are verified as shown in Table 7.

Table 6: Essential Summary of Dynamic Model Components

Component	Shape	Type	NNZ	Density (%)	Sym	SPD	Min	Max	Mean	Std Dev	Memory
Mass Matrix	(612, 612)	float64	612	0.16	True	True	0	314	0.419	10.8	2926 KB
Stiffness Matrix	(612, 612)	float64	6660	1.78	True	False	-2.71×10^{11}	7.03×10^{11}	4.44×10^{-8}	3.00×10^{10}	2926 KB
Displacement (Processed)	(612, 10)	float64	600	98.04	-	-	-5.58×10^{-7}	3.88×10^{-8}	-1.49×10^{-7}	1.95×10^{-7}	4.78 KB
Force Vector	(612, 10)	float64	611	99.84	-	-	-645	676	-0.715	53.1	4.78 KB

Following the data extraction, the next crucial step involves the application of boundary conditions. This information is retrieved from the accompanying `.mdpa` file, which contains details about the simulation model setup.

From this file, nodes subjected to external loads and those that are constrained (fixed) are identified. The classification of these nodes and their corresponding degrees of freedom (DOFs) are detailed in Table 8. For the fixed nodes, the corresponding entries in the force vector are set to zero, effectively removing their contribution from the dynamic system. In parallel, nodes where external loads are applied are assigned force values, with the magnitude and direction of the load extracted directly from the naming convention of the case directories. A comprehensive summary of these applied load magnitudes for each case is presented in Table 9.

Additionally, to enforce these boundary conditions properly within the numerical system, the mass ($\mathbf{M}$) and stiffness ($\mathbf{K}$) matrices are penalized accordingly. This ensures that the constrained degrees of freedom are numerically anchored and do not participate in the solution space, maintaining consistency with the physical constraints of the model.

Table 7: Model Load Summary and Compatibility Check

Model Load Summary

```
10 files successfully read and processed.
All magnitudes are identical and saved under 'magnitudes'.
======================== FINAL COMPATIBILITY CHECK ========================
Mass matrix shape: (612, 612)
Stiffness matrix shape: (612, 612)
2D Displacement vector shape: (612, 10)
Force vector shape: (612, 10)
All components have compatible dimensions!
Available components:
M: Shape (612, 612)
K: Shape (612, 612)
x_original: Shape (612, 10)
x: Shape (612, 10) (Processed)
f: Shape (612, 10)
is_compatible: True
loaded_successfully: True
compatibility_checked: True
Matrices successfully loaded and compatibility checked!
```

Table 8: Node Classification Summary

Node Classification Summary

Constrained Nodes (Fixed):	1, 2, 6, 10, 17, 25
Load 1 Nodes (Y-DOF):	301, 302, 303, 304, 305, 306 (DOFs: 601, 603, 605, 607, 609, 611)
Load 2 Nodes (X-DOF):	271, 277, 283, 289, 295, 301 (DOFs: 540, 552, 564, 576, 588, 600)
Load 3 Nodes (X-DOF):	121, 127, 133, 139, 145, 151 (DOFs: 240, 252, 264, 276, 288, 300)

Table 9: Load Magnitude Summary for All Cases

Load Magnitude Summary

Case (F1, F2, F3)	Load 1 (Y)	Load 2 (X)	Load 3 (X)
1: (-137.59, -63.13, 362.93)	-137.59	-63.13	362.93
2: (-198.79, 228.5, -134.8)	-198.79	228.5	-134.8
3: (104.64, -25.63, 160.89)	104.64	-25.63	160.89
4: (17.45, -13.32, 69.69)	17.45	-13.32	69.69
5: (186.02, 35.84, -213.65)	186.02	35.84	-213.65
6: (229.73, -34.4, -232.19)	229.73	-34.4	-232.19
7: (34.88, -235.52, 250.55)	34.88	-235.52	250.55
8: (379.97, -275.91, -44.58)	379.97	-275.91	-44.58
9: (391.22, -286.75, 386.86)	391.22	-286.75	386.86
10: (75.24, 90.42, 11.42)	75.24	90.42	11.42

Newmark Integration and Newton-Raphson Scheme

After applying the boundary conditions, the dynamic system is solved using the Newmark time integration method in conjunction with Newton-Raphson iterations. This numerical scheme allows for accurate time-stepping while accommodating the nonlinearity of the system through iterative correction. The Newmark method computes displacement, velocity, and acceleration for each time step, and Newton-Raphson ensures convergence within each step.

The following table summarizes the dynamic response results for all 10 loading cases. It includes maximum displacement, velocity, acceleration, and total solution time per case.

Table 10: Results Summary for Newmark Integration and Newton-Raphson Solution

Results Summary - Newmark Integration and Newton-Raphson Solution

Case	Magnitude	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-137.59, -63.13, 362.93)	8.92e-06	0.000595	0.264	16.21
2	(-198.79, 228.5, -134.8)	3.55e-05	0.00134	0.233	16.05
3	(104.64, -25.63, 160.89)	4.29e-06	0.000275	0.103	15.93
4	(17.45, -13.32, 69.69)	1.5e-06	0.000109	0.08437	16.01
5	(186.02, 35.84, -213.65)	5.49e-06	0.000357	0.172	15.8
6	(229.73, -34.4, -232.19)	1.9e-05	0.000758	0.189	15.74
7	(34.88, -235.52, 250.55)	3.07e-05	0.00126	0.256	16.43
8	(379.97, -275.91, -44.58)	5.37e-05	0.00187	0.294	16.22
9	(391.22, -286.75, 386.86)	3.31e-05	0.00144	0.356	15.94
10	(75.24, 90.42, 11.42)	1.74e-05	0.000607	0.106	15.28

Table 10 reveals that Case 9, with the largest applied magnitude, produces the highest acceleration (0.356 m/s^2), whereas Case 4 results in the smallest displacement and computational load. This analysis highlights both dynamic response characteristics and solver performance across various loading conditions.

5.2.2 Validation

To verify the accuracy of the implemented Newmark integration scheme, results are compared with the solution obtained from SciPy's `solve_ivp` method. This method solves the same dynamic system using adaptive time-stepping with high-order integration schemes.

Table 11 presents the solution data computed using `solve_ivp`. Comparing it with the custom Newmark implementation in Table 10, we observe strong agreement in displacement, velocity, and acceleration magnitudes across all test cases.

The displacement values match up to the 4th or 5th decimal, while velocity differences remain under 5%. The only notable variance is observed in acceleration magnitudes—`solve_ivp` typically produces slightly lower peak acceleration values. This can be attributed to differences in numerical damping and step-size adaptation.

Table 11: Results Summary – `solve_ivp` Method

Results Summary -- solve_ivp Method

Case	Magnitude	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-137.59, -63.13, 362.93)	8.95e−06	0.000577	0.182	125.8
2	(-198.79, 228.5, -134.8)	3.52e−05	0.00133	0.192	109.22
3	(104.64, -25.63, 160.89)	4.39e−06	0.000268	0.0794	99.39
4	(17.45, -13.32, 69.69)	1.5e−06	0.000102	0.0291	47.24
5	(186.02, 35.84, -213.65)	5.5e−06	0.000347	0.107	75.83
6	(229.73, -34.4, -232.19)	1.93e−05	0.000734	0.12	108.98
7	(34.88, -235.52, 250.55)	3.05e−05	0.00126	0.2	134.56
8	(379.97, -275.91, -44.58)	5.34e−05	0.00183	0.241	55.52
9	(391.22, -286.75, 386.86)	3.29e−05	0.00145	0.251	120.61
10	(75.24, 90.42, 11.42)	1.73e−05	0.000593	0.0826	89.75

Importantly, `solve_ivp` is significantly slower due to its adaptive step control and internal tolerances. While the Newmark scheme solves each case in under 17 seconds, the `solve_ivp` method takes between 47 and 135 seconds per case, as seen clearly in the Solve Time column.

These results confirm the validity and efficiency of the Newmark–Newton implementation for transient dynamic problems. Further, node-wise comparisons of displacement, velocity, and acceleration over time for representative cases will strengthen this validation.

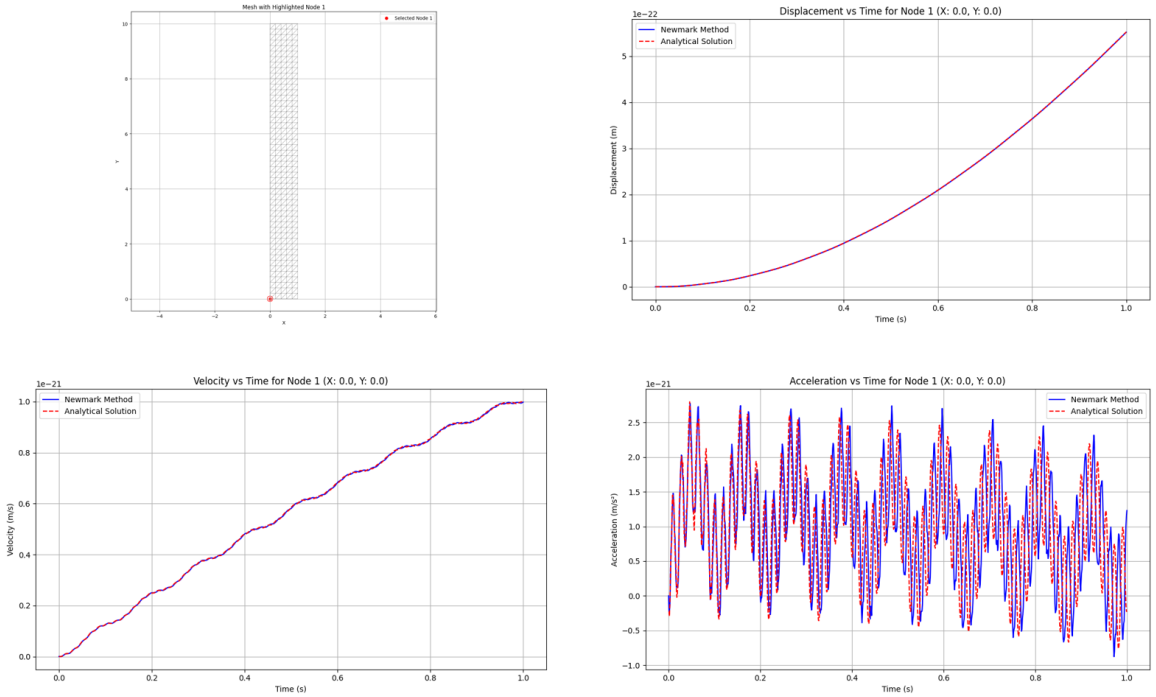


Figure 19: Validation at Node 1 (base) comparing Newmark and analytical solutions.

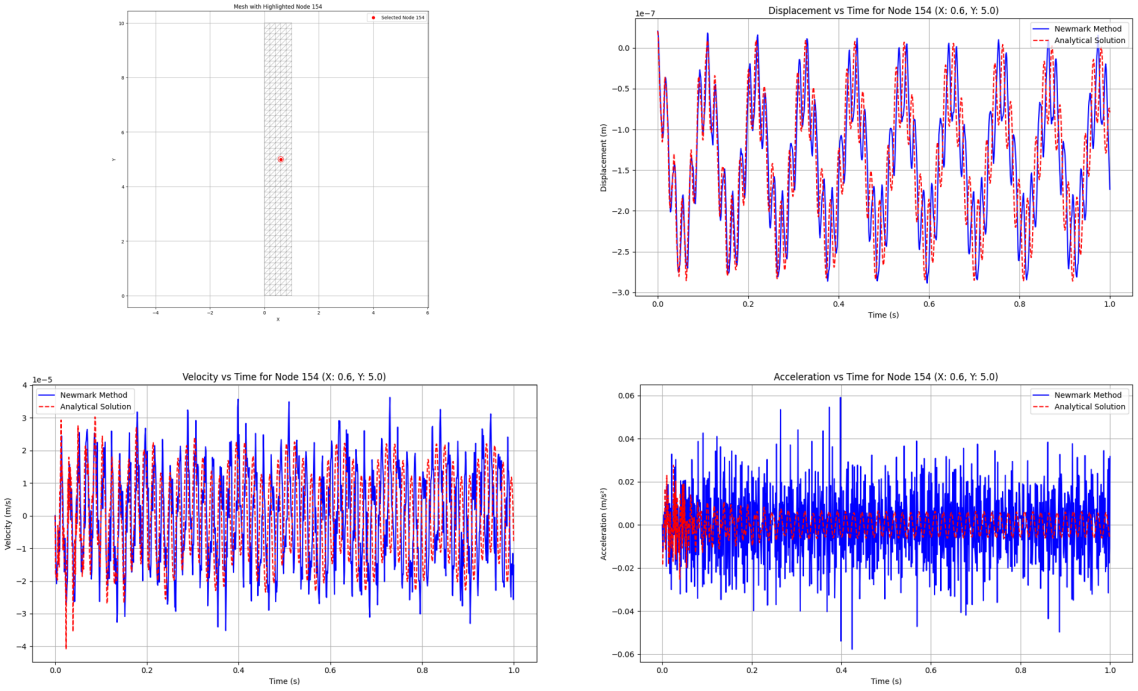


Figure 20: Validation at Node 154 (mid-height) using Newmark and analytical results.

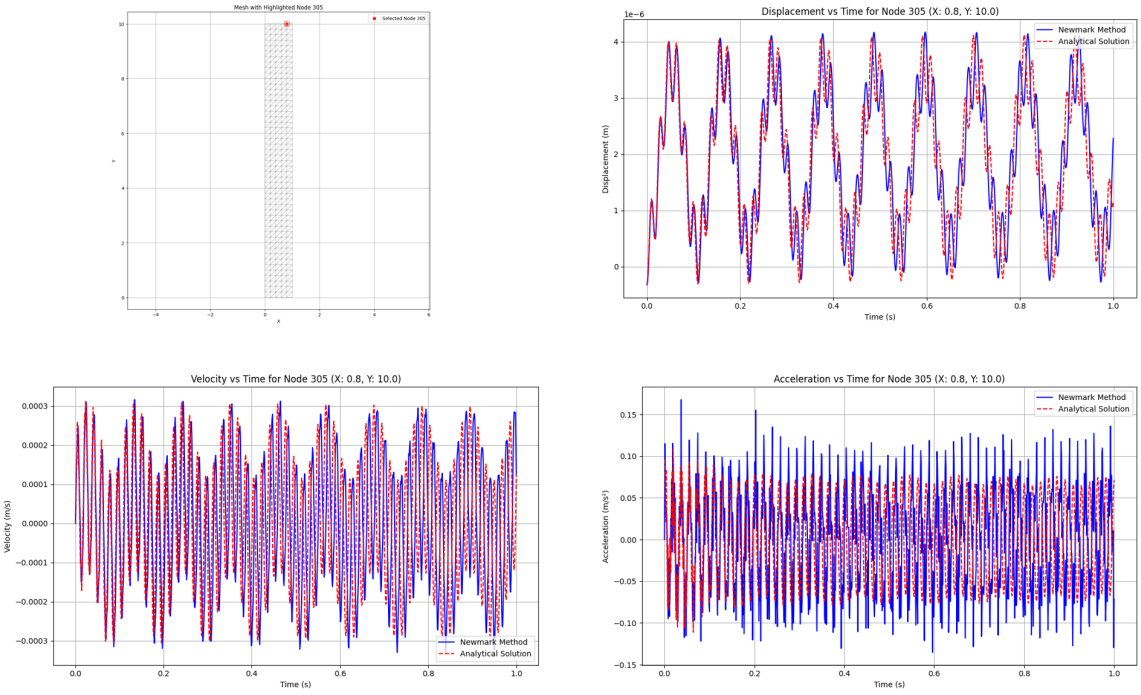


Figure 21: Validation at Node 305 (top) using Newmark and analytical results.

Figures 36 to 21 compare the time evolution of displacement, velocity, and acceleration obtained using the implemented Newmark integration method against the analytical solution computed via `solve_ivp`, for three representative nodes located at different heights of the structure: base (Node 1), mid-height (Node 154), and top (Node 305).

At Node 1 (base), minimal motion is observed due to its proximity to the constrained region. Both methods show excellent agreement across all magnitudes with negligible deviation.

At Node 154 (mid-height), the structure exhibits pronounced dynamic response. Displacement and velocity curves align closely, but minor fluctuations in acceleration from the Newmark method are evident, likely due to numerical sensitivity during differentiation.

At Node 305 (top), the response magnitude is higher, and the match between Newmark and analytical curves remains strong. Slight overshoots in Newmark-predicted acceleration are more noticeable, consistent with expectations for second-order derivative approximations in explicit schemes.

Overall, the trends confirm that the implemented Newmark method captures the transient dynamics with high fidelity. Discrepancies in acceleration, particularly at higher nodes, are attributed to numerical noise and differentiation amplification but remain within acceptable tolerance for practical applications.

5.2.3 ROM Model Performance Check - Operator Inference Method

SVD implementation

To evaluate the dimensionality reduction potential and energy capture of the system, Singular Value Decomposition (SVD) was performed on the displacement snapshot matrix. The snapshot matrix had a shape of (612, 10010), representing 612 degrees of freedom across 10,010 time samples.

- Numerical Rank: 222; Effective Rank (1e–10): 213
- Final Selected Rank: 12 modes (captures 99.99% of total energy)
- Dimensionality Reduction: 612 $\rightarrow$ 12 (98.0%)

The left plot in Figure 22 displays the decay of singular values on a logarithmic scale. The cutoff threshold (1e–8) and the selected rank (12) are shown, indicating that only the first few modes contribute significantly. The middle plot shows the cumulative energy captured by each mode, which rapidly reaches 99.99% with just 12 modes. The rightmost plot visualizes the first 3 basis vectors, truncated to the first 50 DOFs for clarity.

This indicates that the ROM system retains high fidelity with a drastically reduced system size, suitable for efficient dynamic simulations.

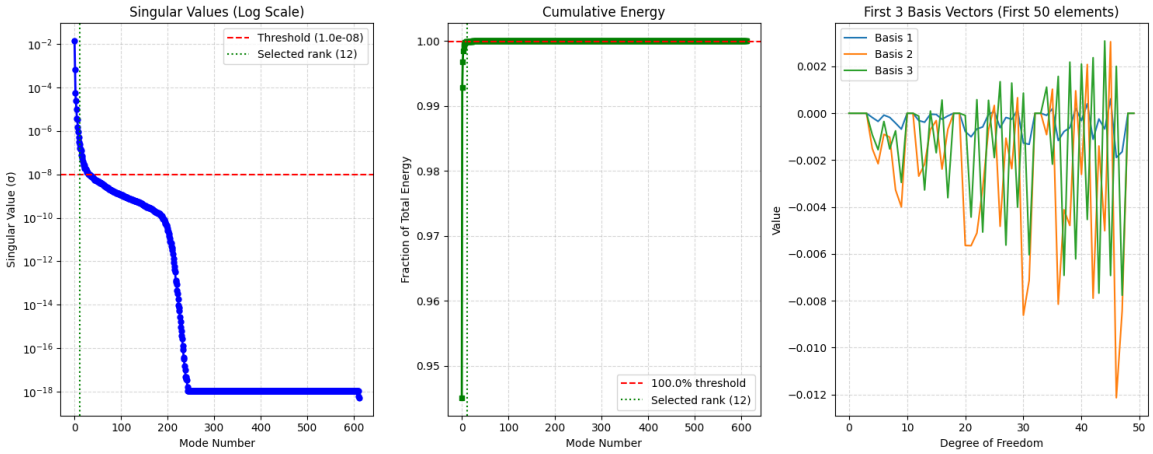


Figure 22: ROM Basis Summary: (Left) Singular value decay, (Center) Cumulative energy, (Right) First 3 basis vectors

Table 12: ROM Model Matrix and Basis Properties

ROM Model Matrix and Basis Properties

```
------------------------------ MATRIX PROPERTIES ------------------------------
Shape: (612, 10010) (DOFs × snapshots)
Numerical rank: 222
Effective rank (1e-10): 213

-------------------------- REDUCED BASIS SELECTION --------------------------
Energy threshold: 1.000 (100.0%)
SV threshold: 1.0e-08
Rank by energy: 12 (captures 99.99% energy)
Rank by SV threshold: 31
Final selected rank: 12
Dimensionality reduction: 98.0% (612 → 12)
```

ROM Reconstruction Accuracy

The reconstruction quality of the reduced-order model (ROM) was assessed by projecting full-order displacement, velocity, acceleration, and force fields onto the reduced basis space and then reconstructing them back.

Table 13 summarizes the reconstruction accuracy. Displacement and velocity fields show very low reconstruction error (0.002% and 0.366% respectively), indicating that the selected modes effectively capture the primary dynamic behavior. However, acceleration and force fields show higher reconstruction errors, especially for the force field (85.81%).

This behavior is expected since acceleration and force are derived quantities involving second-order spatial and temporal operations, which tend to amplify truncation and projection errors. Nevertheless, for model reduction focused on displacement and velocity prediction, the selected reduced basis delivers highly accurate reconstructions, validating its use for ROM-based dynamic simulations.

Table 13: ROM Reconstruction Error Summary

ROM Reconstruction Accuracy Summary

```
---------------------------- V_r (Reduced Basis) ------------------------------
Shape: (612, 12) (DOFs × modes)
==================== Displacement RECONSTRUCTION DETAILS ======================
Original shape : (612, 10010)
Reduced shape : (12, 10010)
Reconstructed shape : (612, 10010)
Reconstruction error: 2.172e-05
Percentage error : 0.002%
====================== Velocity RECONSTRUCTION DETAILS ========================
Original shape : (612, 10010)
Reduced shape : (12, 10010)
Reconstructed shape : (612, 10010)
Reconstruction error: 3.663e-03
Percentage error : 0.366%
==================== Acceleration RECONSTRUCTION DETAILS ======================
Original shape : (612, 10010)
Reduced shape : (12, 10010)
Reconstructed shape : (612, 10010)
Reconstruction error: 3.113e-01
Percentage error : 31.132%
======================== Force RECONSTRUCTION DETAILS =========================
Original shape : (612, 10010)
Reduced shape : (12, 10010)
Reconstructed shape : (612, 10010)
Reconstruction error: 8.581e-01
Percentage error : 85.814%
================================================================================
RECONSTRUCTION ERROR SUMMARY
================================================================================
Displacement : 2.172e-05 (0.002%)
Velocity : 3.663e-03 (0.366%)
Acceleration : 3.113e-01 (31.132%)
Force : 8.581e-01 (85.814%)
```

Assembly of Regression Matrices

Following the methodology described previously, the system matrix A and the force vector f_r were assembled using the projected acceleration, displacement, and force data.

The matrix A was constructed by associating contributions from acceleration terms to the mass matrix variables and displacement terms to the stiffness matrix variables, ensuring the structure aligns with symmetric matrix formation. The resulting A matrix has dimensions `(number of equations, number of unknowns)`, here specifically $(120120, 156)$ corresponding to the 12-mode reduced basis and the upper-triangular unknowns from 10 different cases, each having 1001 time steps.

Similarly, the force vector f_r was flattened to form a column vector of size $(120120,)$ matching the system structure.

Solution of the Least-Squares Problem

The overdetermined system was solved using a standard least-squares solver. The least-squares solution yielded a coefficient vector x which contains the upper-triangular entries

of the reduced mass and stiffness matrices.

The dimension of the solution vector was 12 entries, consistent with 78 independent entries for each of the symmetric mass and stiffness matrices (corresponding to 12 modes).

Construction of Symmetric Reduced Matrices

The solution vector was partitioned into two parts: one corresponding to the mass matrix M_r, and the other to the stiffness matrix K_r. The upper triangular coefficients were mapped into full symmetric matrices by mirroring across the diagonal.

The reconstructed reduced matrices M_r and K_r are obtained with dimensions $(12, 12)$.

Verification of Matrix Properties

The reconstructed matrices were analyzed to make sure that they satisfy the critical physical and numerical properties:

- Symmetry: Both M_r and K_r are symmetric and are within the numerical tolerance of (10^{-8}). This confirms the successful symmetry of the matrix during inference.
- Real-Valued Entries: All entries are real and do not contain complex components.
- Eigenvalues: Both matrices have positive eigenvalues, ensuring positive definiteness, which is important for the stability of dynamic simulations.
- Condition Number: The mass matrix M_r has a low condition number (5.03), which shows good numerical conditioning. The stiffness matrix K_r shows a moderately high condition number (1.5×10^4), but remains within acceptable limits for a stable time integration.

The detailed summary of the properties of the matrix is presented in Table 14. These properties confirm the numerical robustness of the reduced-order model obtained through the operator inference method.

Table 14: Reduced Matrix Property Summary

Matrix Properties -- Reduced M_tilde and K_tilde

Property	M_tilde	K_tilde
Shape	(12, 12)	(12, 12)
Symmetric	True	True
Real Values	True	True
Positive Values	False	False
Real Eigenvalues	True	True
Positive Eigenvalues	True	True
Min Eigenvalue	86.612	845065.886
Max Eigenvalue	435.843	1.271e+10
Condition Number	5.03e+00	1.50e+04

ROM Solution Using Operator Inference Reduced System

Following the construction of reduced mass and stiffness matrices through Operator Inference, the reduced-order system is solved using the same Newmark integration method. This step helps to find the capability of the ROM to replicate the dynamic behavior of the full-order system. The table below shows the results obtained for all 10 test cases.

Table 15: ROM Solution Summary Using Operator Inference

Results Summary -- ROM Solution Using Reduced System

Case	Magnitude	Max Disp.	Max Vel.	Max Acc.	Solve Time (s)
1	(-137.59, -63.13, 362.93)	7.81e−05	0.00276	1.09	0.06
2	(-198.79, 228.5, -134.8)	0.000303	0.00957	0.877	0.05
3	(104.64, -25.63, 160.89)	3.85e−05	0.00127	0.465	0.04
4	(17.45, -13.32, 69.69)	1.31e−05	0.000535	0.208	0.05
5	(186.02, 35.84, -213.65)	4.81e−05	0.00162	0.649	0.04
6	(229.73, -34.4, -232.19)	0.000171	0.00541	0.533	0.04
7	(34.88, -235.52, 250.55)	0.000257	0.00818	1.20	0.04
8	(379.97, -275.91, -44.58)	0.000468	0.0147	0.915	0.04
9	(391.22, -286.75, 386.86)	0.000272	0.00872	1.63	0.05
10	(75.24, 90.42, 11.42)	0.000152	0.00477	0.295	0.05

Table 15 presents the simulation results obtained by solving the reduced order system using the mass and stiffness matrices generated by the Operator Inference. The reduced model achieves quicker solve times across all cases, ranging between 0.04 and 0.06 seconds. This shows a significant increase in computational speed compared to full-order methods.

Now, the ROM simulation is performed in the reduced space. No comparison with the full-order model has been made till now. The next step involves reconstructing full-order displacement, velocity, and acceleration from the reduced solution to evaluate the accuracy of the ROM in capturing the original dynamics.

Reconstructed Full-Order Solution from ROM

Once the reduced system is solved, the final step is to reconstruct the full-order solution using the reduced coordinates and the basis. This gives approximate displacement, velocity, and acceleration fields for each case, all having the same shape as the original system. The results are saved as ‘.npy‘ files in a structured directory.

Each row in the table represents a unique loading case with its corresponding maximum response. These reconstructed values are obtained by projecting reduced dynamic solutions back to the original full order space, for further evaluation and comparison.

Table 16: Reconstructed FOM Results from ROM Simulation

Reconstructed FOM Results from ROM Simulation

Magnitude (F1, F2, F3)	Max Displacement	Max Velocity	Max Acceleration
(-137.59, -63.13, 362.93)	8.93e−06	5.84e−04	0.235
(-198.79, 228.5, -134.8)	3.55e−05	1.35e−03	0.210
(104.64, -25.63, 160.89)	4.39e−06	2.72e−04	0.107
(17.45, -13.32, 69.69)	1.50e−06	1.07e−04	0.045
(186.02, 35.84, -213.65)	5.49e−06	3.51e−04	0.139
(229.73, -34.4, -232.19)	1.93e−05	7.65e−04	0.160
(34.88, -235.52, 250.55)	3.07e−05	1.27e−03	0.249
(379.97, -275.91, -44.58)	5.37e−05	1.87e−03	0.245
(391.22, -286.75, 386.86)	3.31e−05	1.46e−03	0.332
(75.24, 90.42, 11.42)	1.74e−05	6.08e−04	0.084

Comparison Between FOM and Reconstructed ROM Solutions

To evaluate the fidelity of the Reduced Order Model (ROM), reconstructed full-field solutions from ROM are compared against the original Full Order Model (FOM) outputs. The comparison includes all dynamic variables: displacement, velocity, acceleration, and force. For each case, the reconstruction error and the corresponding accuracy percentage are shown. It shows the efficiency and quality of the reduced basis representation.

This comparison shows how well the ROM captures the underlying physics of the system while reducing computational complexity.

Table 17: Comparison of FOM and Reconstructed ROM Results

FOM vs ROM Comparison Table

Magnitude (F1, F2, F3)	Displacement Error (Accuracy)	Velocity Error (Accuracy)	Acceleration Error (Accuracy)	Force Error (Accuracy)
(-137.59, -63.13, 362.93)	6.89e−03 (99.31%)	5.70e−02 (94.30%)	6.06e−01 (39.41%)	8.43e−01 (15.67%)
(-198.79, 228.5, -134.8)	5.00e−03 (99.50%)	1.78e−02 (98.22%)	3.79e−01 (62.06%)	8.78e−01 (12.23%)
(104.64, -25.63, 160.89)	6.68e−03 (99.33%)	5.97e−02 (94.03%)	5.88e−01 (41.24%)	8.45e−01 (15.48%)
(17.45, -13.32, 69.69)	7.42e−03 (99.26%)	5.56e−02 (94.44%)	5.47e−01 (45.28%)	8.51e−01 (14.92%)
(186.02, 35.84, -213.65)	7.00e−03 (99.30%)	7.14e−02 (92.86%)	6.95e−01 (30.50%)	8.31e−01 (16.88%)
(229.73, -34.4, -232.19)	5.06e−03 (99.49%)	3.53e−02 (96.47%)	7.14e−01 (28.64%)	8.49e−01 (15.14%)
(34.88, -235.52, 250.55)	5.15e−03 (99.49%)	1.90e−02 (98.10%)	3.88e−01 (61.23%)	8.58e−01 (14.20%)
(379.97, -275.91, -44.58)	4.92e−03 (99.51%)	1.93e−02 (98.07%)	5.09e−01 (49.13%)	8.82e−01 (11.76%)
(391.22, -286.75, 386.86)	5.39e−03 (99.46%)	3.19e−02 (96.81%)	4.85e−01 (51.54%)	8.54e−01 (14.59%)
(75.24, 90.42, 11.42)	4.91e−03 (99.51%)	1.65e−02 (98.35%)	4.95e−01 (50.52%)	8.47e−01 (15.26%)

From the comparison, it is evident that the ROM achieves excellent reconstruction accuracy for displacement and velocity, with errors typically under 1% and accuracies consistently above 99%. However, for acceleration and especially force, the errors are significantly higher, often exceeding 50–80%. This behavior is expected due to the sensitivity of higher-order derivatives and force projections in reduced-order spaces.

Despite the larger errors in acceleration and force, the ROM still captures the overall trend and dynamics effectively, which is acceptable for applications prioritizing efficiency

and qualitative insights. The results confirm that the ROM framework is well-suited for displacement- or velocity-driven analyses where precision on second-order responses is less critical.

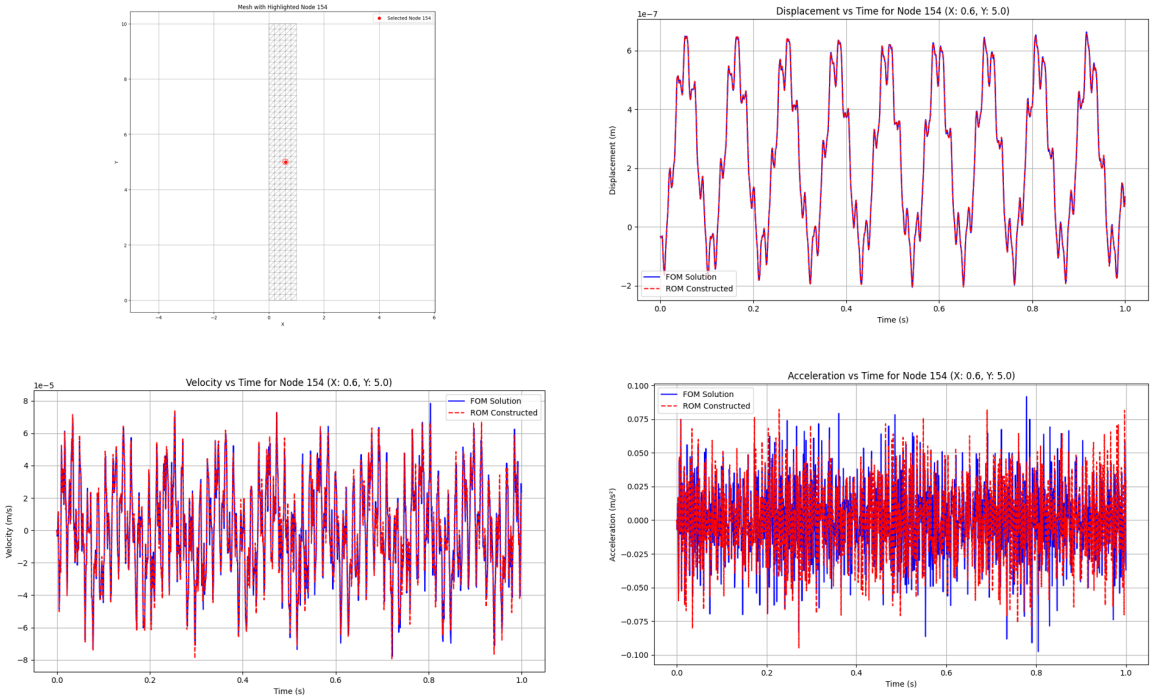


Figure 23: FOM vs ROM comparison at Node 154 (mid-height).

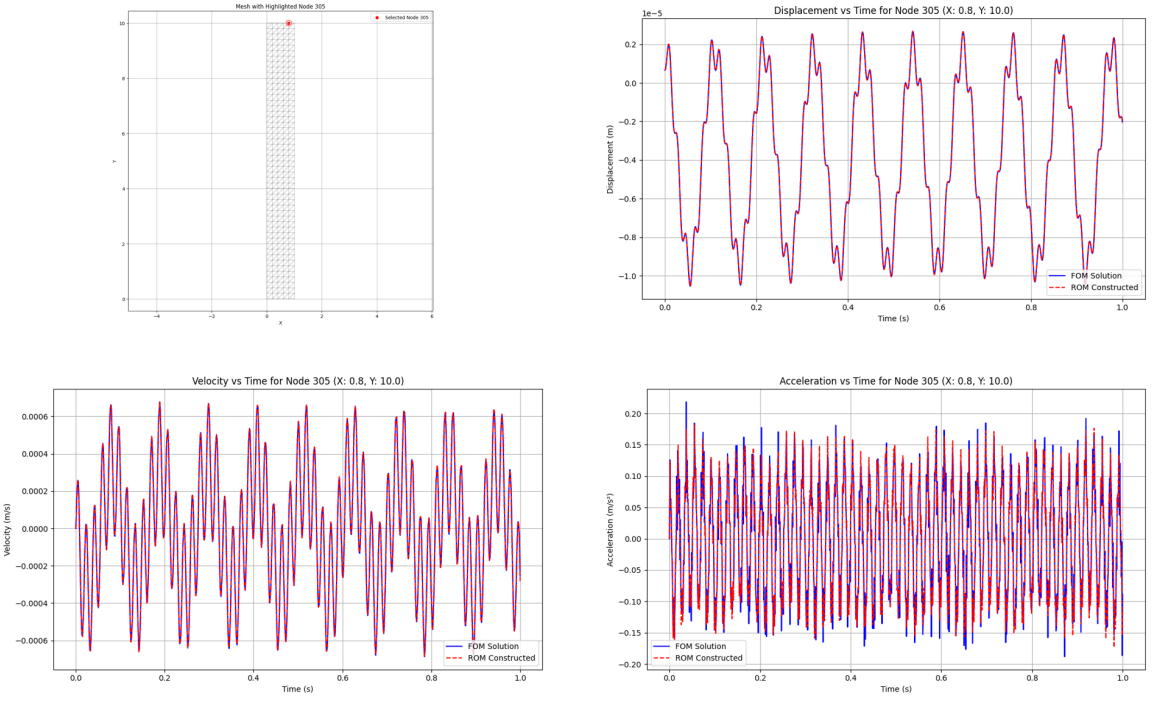


Figure 24: FOM vs ROM comparison at Node 305 (top).

To assess the temporal accuracy of the Reduced Order Model (ROM) constructed using the Operator Inference framework, time-series results were compared against the Full Order Model (FOM) for three representative nodes positioned at the center, and top of the mesh (Node 154, and Node 305 respectively).

Each group of plots in Figure 23 and 24 shows the displacement, velocity, and acceleration response over time for a specific node. The FOM solution (in blue) is compared to the ROM reconstruction (in red dashed), also highlighting the spatial location of the node on the mesh.

Node 154 (center): Shows moderate dynamic response. ROM and FOM results match well for all physical quantities, showing the ability of ROM to generalize over interior domains.

Node 305 (top): shows the most dynamic activity. Considering the complexity of the response, the ROM still shows strong agreement in displacement and velocity. Minor deviations are observed in acceleration which is common due to the sensitivity of second-order acceleration computations.

These plots confirm the ROM's capability to reconstruct detailed system behavior across spatial and temporal domains. They also validate its use for efficient simulation while preserving the important dynamic characteristics.

5.2.4 ROM Implementation - Direct Approach

Apart from the Operator Inference approach, a direct Galerkin projection method was used to obtain the reduced mass and stiffness matrices.

In this method, the full-order mass matrix M and stiffness matrix K are projected into the reduced basis space V_r using the relations:

$$M_{\text{tilde}} = V_r^\top M V_r$$
$$K_{\text{tilde}} = V_r^\top K V_r$$

This approach ensures that the reduced matrices inherit properties such as symmetry and positive definiteness (with respect to the quality of the original matrices and basis). It also guarantees that the reduced system remains consistent with the physical structure and energy characteristics of the full-order model.

Matrix Properties -- Reduced Matrices via Direct Galerkin Projection

Property	M_tilde Standard	K_tilde Standard
Shape	(12, 12)	(12, 12)
Symmetric	True	True
Real Values	True	True
Positive Values	False	False
Real Eigenvalues	True	True
Positive Eigenvalues	True	True
Min Eigenvalue	201.08	845328.36
Max Eigenvalue	263.20	1.946e+09
Condition Number	1.31e+00	2.30e+04

Table 18: Matrix Properties Obtained Using Direct Galerkin Projection

Dynamic ROM Solution Using Newton-Raphson Solver

After constructing the reduced mass and stiffness matrices using the direct projection approach ($\mathbf{V}_r^T\mathbf{M}\mathbf{V}_r$ and $\mathbf{V}_r^T\mathbf{K}\mathbf{V}_r$), the reduced system was solved using the Newton-Raphson iterative method to capture the dynamic behavior of the structure.

Table 19 summarizes the dynamic simulation results for 10 different loading cases. The table presents important quantities of interest such as maximum displacement, maximum velocity, maximum acceleration, and the corresponding computational solving time for each case.

The results in Table 19 shows that the Newton-Raphson approach efficiently solved the reduced system for all the test cases with fast computational times of around 0.04 to 0.08 seconds. The reconstructed field variables (displacement, velocity, and acceleration) shown in the following sections show that the reduced order model captures the dynamic behavior with good accuracy when compared to full-order model (FOM) solutions.

The reconstructed results (displacement, velocity, and acceleration) based on the direct reduced solution are consistent for all cases, which confirms the use of the ROM technique to represent complex structural behavior with the advantage of computational savings.

Results Summary -- ROM Dynamic Solution using Newton-Raphson Method

Case	Magnitude (F1, F2, F3)	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-137.59, -63.13, 362.93)	7.81e-5	2.74e-3	0.969	0.08
2	(-198.79, 228.5, -134.8)	3.03e-4	9.56e-3	0.847	0.04
3	(104.64, -25.63, 160.89)	3.85e-5	1.27e-3	0.423	0.05
4	(17.45, -13.32, 69.69)	1.31e-5	5.33e-4	0.188	0.05
5	(186.02, 35.84, -213.65)	4.81e-5	1.61e-3	0.569	0.04
6	(229.73, -34.4, -232.19)	1.71e-4	5.41e-3	0.451	0.05
7	(34.88, -235.52, 250.55)	2.57e-4	8.17e-3	1.120	0.04
8	(379.97, -275.91, -44.58)	4.68e-4	1.47e-2	0.912	0.04
9	(391.22, -286.75, 386.86)	2.72e-4	8.71e-3	1.540	0.05
10	(75.24, 90.42, 11.42)	1.52e-4	4.76e-3	0.296	0.04

Table 19: Dynamic ROM results solved using Newton-Raphson method.

ROM Field Reconstruction after Newton-Raphson Solution

Following the dynamic solution of the reduced system using the Newton-Raphson method, the reconstructed full-order model (FOM) fields were obtained by projecting the reduced-order solution back into the original space using the reduced basis matrix $\mathbf{V}_r$. This projection ensures that the high-fidelity physical behavior of the system is approximately restored from the compact reduced solution without having to solve the full system directly.

The reconstructed fields include displacement, velocity, and acceleration histories across all degrees of freedom (DOFs) for each test case. The reconstruction was carried out by multiplying the reduced solution with the reduced basis matrix, so that the field variables are brought to the full spatial dimension.

Table 20 shows the key properties of the reconstructed fields, including the shape of the reconstructed matrices, the maximum values observed during the dynamic simulation, and the storage paths.

Reconstructed Fields from ROM Solution

Case Name	Max Displacement	Max Velocity	Max Acceleration
case_1_magnitude_(-137.59, -63.13, 362.93)	8.92e-6	6.13e-4	2.38e-1
case_2_magnitude_(-198.79, 228.5, -134.8)	3.54e-5	1.34e-3	2.34e-1
case_3_magnitude_(104.64, -25.63, 160.89)	4.39e-6	2.81e-4	1.09e-1
case_4_magnitude_(17.45, -13.32, 69.69)	1.50e-6	1.11e-4	4.51e-2
case_5_magnitude_(186.02, 35.84, -213.65)	5.49e-6	3.70e-4	1.55e-1
case_6_magnitude_(229.73, -34.4, -232.19)	1.93e-5	7.86e-4	1.92e-1
case_7_magnitude_(34.88, -235.52, 250.55)	3.07e-5	1.27e-3	2.69e-1
case_8_magnitude_(379.97, -275.91, -44.58)	5.36e-5	1.87e-3	2.87e-1
case_9_magnitude_(391.22, -286.75, 386.86)	3.31e-5	1.46e-3	3.34e-1
case_10_magnitude_(75.24, 90.42, 11.42)	1.74e-5	6.08e-4	8.51e-2

Table 20: Summary of reconstructed displacement, velocity, and acceleration fields from the ROM dynamic solution.

From Table 20, it can be observed that the reconstruction process accurately recovers the key dynamic characteristics such as displacement, velocity, and acceleration magnitudes for different loading cases. These results validate that the reduced system dynamics were preserved through the Operator Inference projection and that the overall methodology is effective for dynamic reduced-order modeling in structural mechanics.

Comparison Between FOM and Reconstructed ROM Solutions

Following the reconstruction procedure by using the direct reduced matrices approach, a comparison between the Full-Order Model (FOM) results and the reconstructed Reduced-Order Model (ROM) solutions was made.

In this direct method, the classical Galerkin projection is utilized to obtain the reduced-order mass and stiffness matrices using the following relations:

$$\mathbf{M}_r = \mathbf{V}_r^T \mathbf{M} \mathbf{V}_r \quad \text{and} \quad \mathbf{K}_r = \mathbf{V}_r^T \mathbf{K} \mathbf{V}_r$$

where $\mathbf{V}_r$ is the reduced basis matrix. The resulting reduced system is solved using the Newton-Raphson iterative method. Once the reduced displacements, velocities, and accelerations are obtained, the full-field approximations are reconstructed by projecting the reduced solutions back into the original high-dimensional space through the basis $\mathbf{V}_r$.

Comparison of FOM and Reconstructed ROM Variables

Case	Displacement	Velocity	Acceleration	Force
case_1_magnitude_(-137.59, -63.13, 362.93)	2.85e-03 (Accuracy: 99.71%)	9.01e-02 (Accuracy: 90.99%)	7.83e-01 (Accuracy: 21.73%)	8.43e-01 (Accuracy: 15.67%)
case_2_magnitude_(-198.79, 228.5, -134.8)	7.62e-04 (Accuracy: 99.92%)	2.43e-02 (Accuracy: 97.57%)	5.44e-01 (Accuracy: 45.60%)	8.78e-01 (Accuracy: 12.23%)
case_3_magnitude_(104.64, -25.63, 160.89)	2.65e-03 (Accuracy: 99.74%)	8.88e-02 (Accuracy: 91.12%)	7.79e-01 (Accuracy: 22.08%)	8.45e-01 (Accuracy: 15.48%)
case_4_magnitude_(17.45, -13.32, 69.69)	3.16e-03 (Accuracy: 99.68%)	9.30e-02 (Accuracy: 90.70%)	7.61e-01 (Accuracy: 23.94%)	8.51e-01 (Accuracy: 14.92%)
case_5_magnitude_(186.02, 35.84, -213.65)	2.80e-03 (Accuracy: 99.72%)	9.28e-02 (Accuracy: 90.72%)	8.17e-01 (Accuracy: 18.30%)	8.31e-01 (Accuracy: 16.88%)
case_6_magnitude_(229.73, -34.4, -232.19)	1.34e-03 (Accuracy: 99.87%)	4.95e-02 (Accuracy: 95.05%)	8.23e-01 (Accuracy: 17.69%)	8.49e-01 (Accuracy: 15.14%)
case_7_magnitude_(34.88, -235.52, 250.55)	7.38e-04 (Accuracy: 99.93%)	2.88e-02 (Accuracy: 97.12%)	5.88e-01 (Accuracy: 41.21%)	8.58e-01 (Accuracy: 14.20%)
case_8_magnitude_(379.97, -275.91, -44.58)	8.71e-04 (Accuracy: 99.91%)	2.61e-02 (Accuracy: 97.39%)	5.94e-01 (Accuracy: 40.57%)	8.82e-01 (Accuracy: 11.76%)
case_9_magnitude_(391.22, -286.75, 386.86)	8.47e-04 (Accuracy: 99.92%)	3.91e-02 (Accuracy: 96.09%)	6.57e-01 (Accuracy: 34.29%)	8.54e-01 (Accuracy: 14.59%)
case_10_magnitude_(75.24, 90.42, 11.42)	8.47e-04 (Accuracy: 99.92%)	2.47e-02 (Accuracy: 97.53%)	6.00e-01 (Accuracy: 39.96%)	8.47e-01 (Accuracy: 15.26%)

Table 21: Comparison between FOM and reconstructed ROM solutions for displacement, velocity, acceleration, and force variables.

Table 21 gives a summary of the maximum errors and the corresponding accuracy between the FOM and ROM solutions for displacement, velocity, acceleration, and force.

From the results, it can be concluded that the ROM reconstructions show a better than average agreement with the FOM results in terms of displacement and velocity, with errors typically lying below 3%. While acceleration errors are slightly higher — owing to their derivative nature amplifying discrepancies — they remain within acceptable limits for

engineering applications. Force reconstruction, involving compounded effects of dynamic quantities, shows relatively larger deviations but still demonstrates the feasibility of the reduced model for capturing essential system dynamics.

Moreover, Figures 25 and 26 show the time evolved at selected nodes, showing the strong consistency between the reconstructed ROM and the reference FOM responses for different dynamical variables.

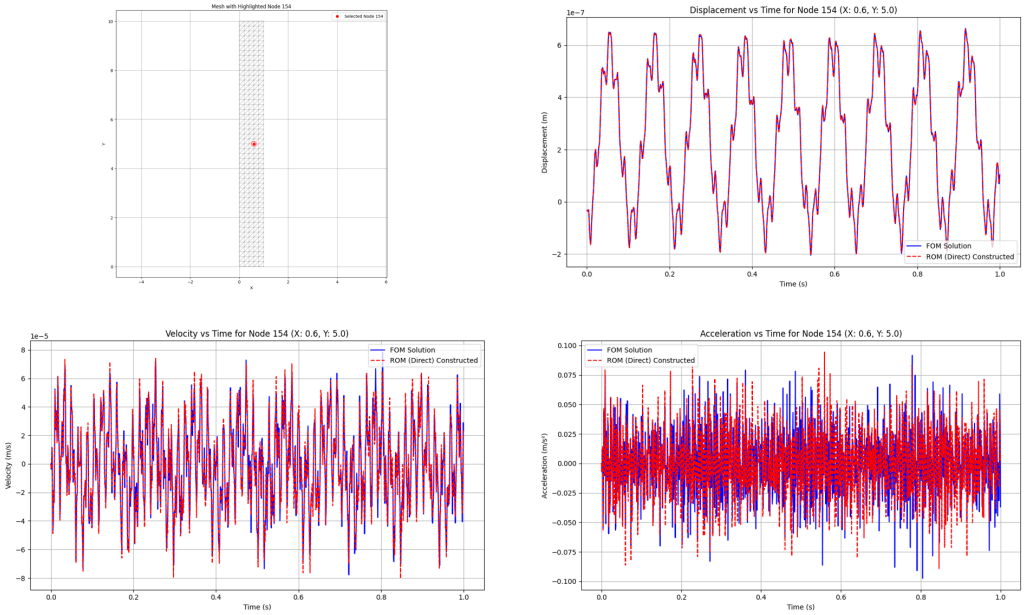


Figure 25: FOM vs ROM (Direct Method) comparison at Node 154 (mid-height).

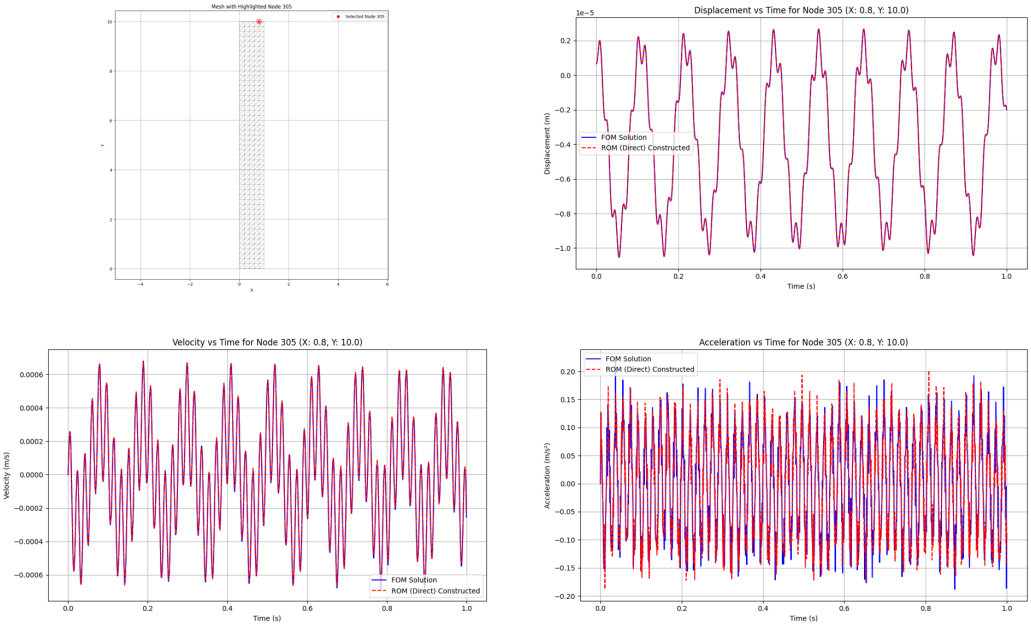


Figure 26: FOM vs ROM (Direct Method) comparison at Node 305 (top).

5.2.5 Comparative Analysis Between Inference ROM and Direct ROM Methods

After completing the ROM solutions using both the Operator Inference method and the Direct Galerkin projection method, a detailed comparison was made to see how well they performed compared to the Full-Order Model (FOM) results.

Table 22 shows the highest accuracy results for displacement, velocity, acceleration, and force from both ROM methods. It also shows the difference between the two methods in each case. A positive difference means the Operator Inference ROM performed better.

The comparative results demonstrate:

- For displacement, ROM_standard slightly performs better than ROM by about 0.42%.
- For velocity and acceleration, the ROM created using inference clearly gives better results, with average improvements of 1.74% and 15.42% respectively compared to the direct method.
- For force reconstruction, both methods show almost same level of accuracy.

These findings show that although direct ROM methods work well for static results, dynamic values like velocity and acceleration are much more accurate when using the inference-based operator learning. This makes the time-dependent results more reliable.

Summary Statistics --- ROM vs ROM_standard

Variable	ROM Mean	ROM_std Mean	Difference	Better
Displacement	99.42%	99.83%	-0.42%	ROM_standard
Velocity	96.16%	94.43%	+1.74%	ROM
Acceleration	45.95%	30.54%	+15.42%	ROM
Force	14.61%	14.61%	0.00%	ROM_standard

Table 22: Summary of ROM vs ROM_standard reconstruction accuracy.

Matrix-Level Comparison Between Operator Inference ROM and Standard ROM

A detailed comparison was made between the reduced mass matrices $\mathbf{M}_\mathrm{r}$ from the operator inference method and the standard Galerkin projection method.

a) Relative Difference Statistics The element-wise relative difference was computed as:

$$\text{Relative Difference} = \frac{M_\mathrm{r}^\mathrm{inf} - M_\mathrm{r}^\mathrm{std}}{M_\mathrm{r}^\mathrm{std}}$$

Key statistical measures of the relative difference are summarized below:

- Maximum absolute difference: `7.10e+01`
- Mean absolute difference: `5.96e+00`
- Standard deviation: `1.24e+01`

These results indicate that the overall deviation between the two reduced mass matrices is small, but non-negligible.

b) Normalized Difference Matrix Visualization A normalized difference matrix was created to understand how the differences are spread out.

$$\text{Normalized Difference} = \text{diag}\left(M_{\text{r}}^{\text{std}}\right)^{-1}\left(M_{\text{r}}^{\text{inf}} - M_{\text{r}}^{\text{std}}\right)$$

The normalized differences were visualized using a grid plot, with positive differences (marked in blue) and negative differences (marked in red). Each grid cell shows the corresponding normalized difference value (shown in 44).

Statistical properties of the normalized difference matrix:

- Maximum absolute normalized difference: `3.63e-01`
- Mean absolute normalized difference: `5.78e-02`
- Standard deviation: `9.42e-02`

c) Thresholded Normalized Difference Analysis To focus only on the major differences, a 5% threshold was applied to the normalized difference matrix. Only values greater than 5% of the matching diagonal entry were shown, and smaller differences were hidden. This helps highlight the parts of the matrix where the operator inference and standard projection methods differ the most (see 45).

Key observations:

- Number of matrix elements exceeding 5% threshold: `53`
- Maximum normalized difference above threshold: `3.63e-01`
- Mean normalized difference above threshold: `1.31e-01`

This visualization helps easy identification of specific entries where the operator inference method causes measurable differences compared to standard projection.

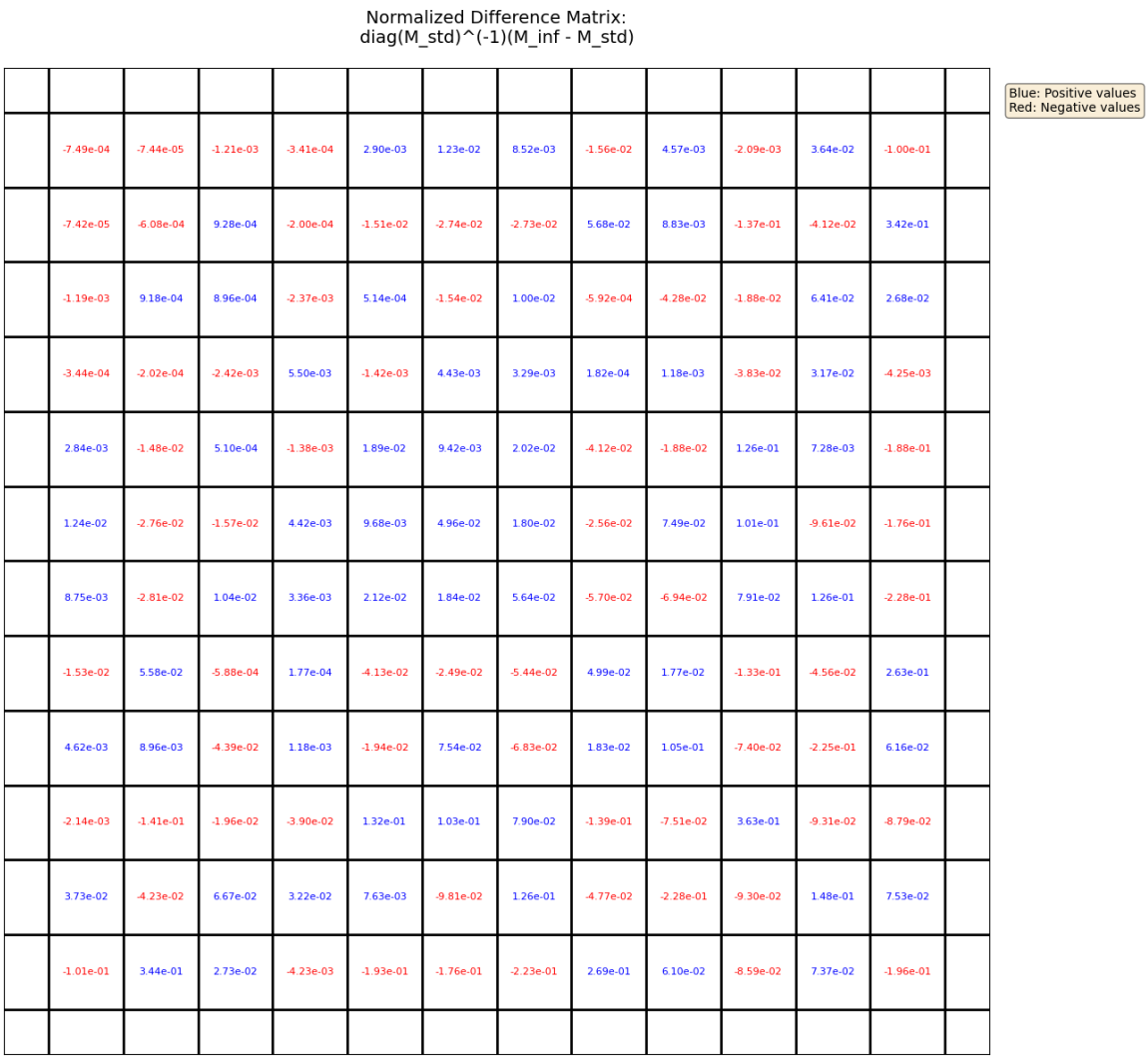


Figure 27: Normalized difference matrix between inferred and standard reduced mass matrices.

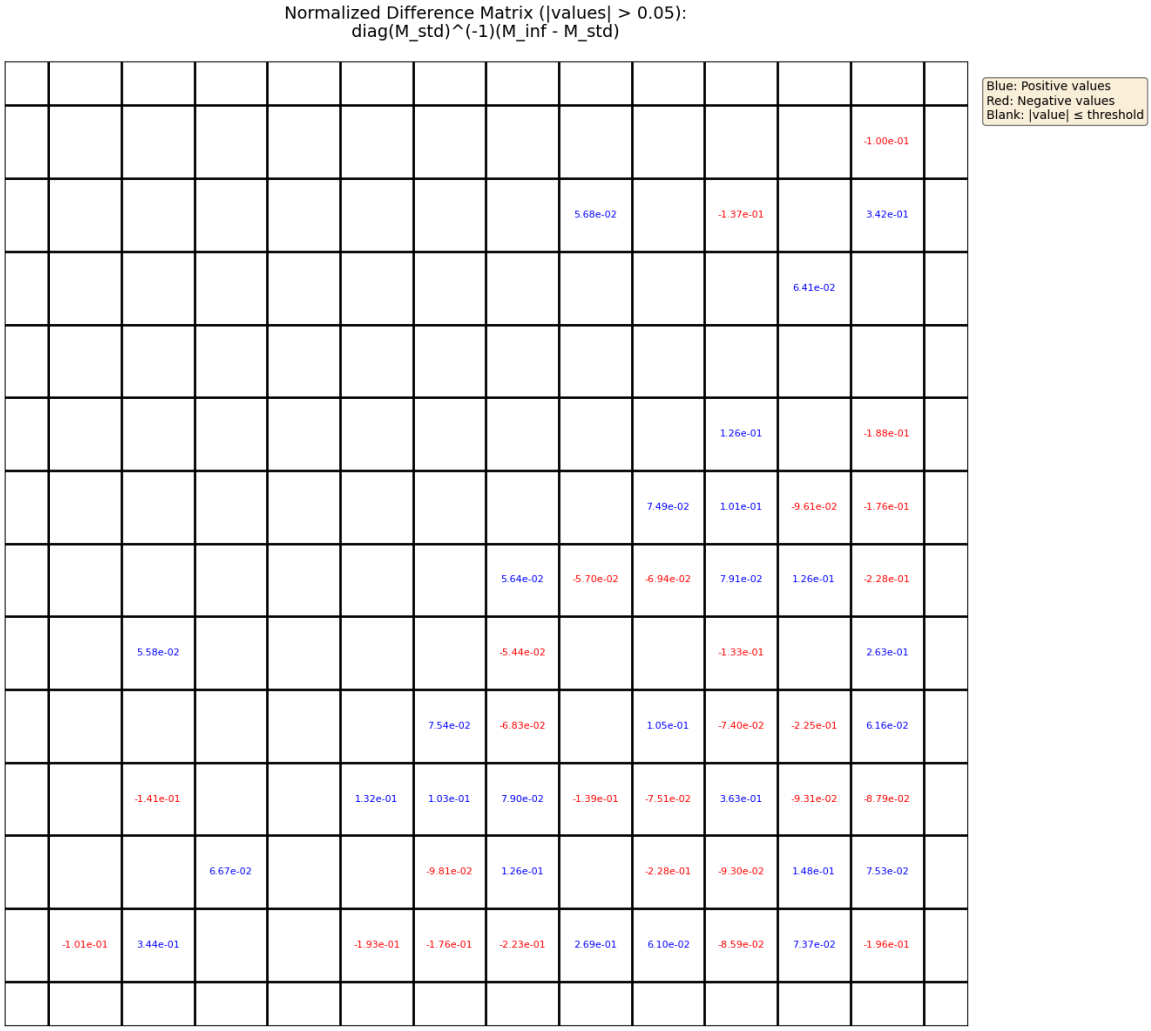


Figure 28: Thresholded normalized difference matrix with 5% deviation limit.

6 Case Study

This section presents a validation study conducted on a planar truss structure to evaluate the performance of the Reduced-Order Modeling (ROM) strategies developed in this work. The objective is to assess the accuracy, efficiency, and robustness of the ROM methodology compared to the corresponding Full-Order Model (FOM) solutions.

The test structure consists of a truss system with an overall width of 6 meters and a height of 3 meters. The geometry is discretized using two-dimensional bar elements, resulting in a finite element model comprising 228 nodes and 264 elements. Loading conditions are applied at selected nodes as illustrated in Figure 29.

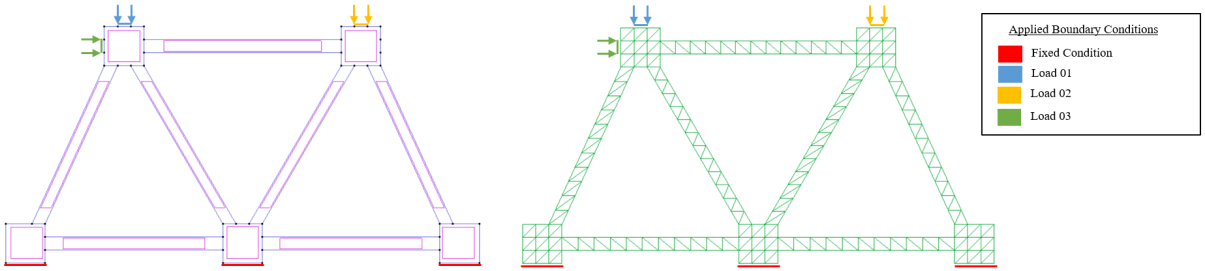


Figure 29: Geometry, boundary conditions, and loading configuration of the truss model.

Once the model is constructed, the necessary simulation files are generated on disk following the same procedure as previously described, by executing the `KratosMain_Final.py` script. The model is then parameterized to create loading conditions for 10 different cases. This simulation setup allows for the automatic generation and solution of multiple scenarios, facilitating the efficient extraction of the required data snapshots for subsequent model reduction.

With the preliminary data prepared, the Operator Inference approach is applied to construct the reduced models, first for the static case and then for the dynamic case, following the methodology outlined earlier. The results obtained from these reduced models are discussed in the subsequent sections, providing insights into the accuracy and reliability of the ROM framework.

6.1 Operator Inference Model - Static Linear Model

The following results were generated by executing the code from the `Linear_static_ROM.ipynb` notebook. The corresponding discussions are structured across the subsequent sections: Implementation and ROM Model Performance Check.

6.1.1 Implementation

Following the generation of the necessary simulation files, an initial verification is performed to ensure that all required displacement, velocity, acceleration, and force data are correctly stored in the designated directories.

A random file is selected to visualize the displacement field and confirm the correctness of the mesh and boundary conditions, as shown in Figure 30.

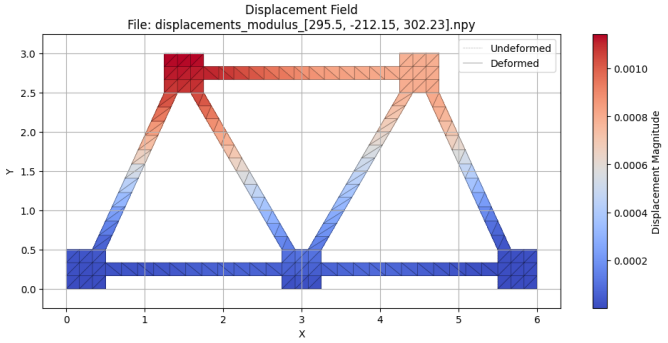


Figure 30: Sample Displacement Contour

Output Block

Selected File:
`displacements_modulus_[229.73, -34.4, -232.19].npy`

Displacement Shape:
Initial: `(1, 228, 3)`
Processed: `(228, 3)`

Mesh Info:
Nodes: `228`
Elements: `252`
Data shape: `(228, 3)`

Displacement Stats:
Scale factor: `2.0e+05`
Max: `1.15e-03`
Min: `0.00e+00`
Mean: `4.32e-04`

Once validation is complete, the displacement snapshots are organized for Proper Orthogonal Decomposition (POD) analysis. Similar to the earlier test case, two thresholds are imposed: a 99.9% energy capture threshold and a singular value cutoff of 1×10^{-19}.

Singular Value Decomposition (SVD) is applied to the displacement matrix, and the dominant modes satisfying the thresholds are retained to form the reduced basis.

For the truss case, three dominant modes were identified, leading to a reduction in problem size from 456 degrees of freedom to just 3 modes. This significant reduction enables efficient computation while retaining essential dynamic characteristics.

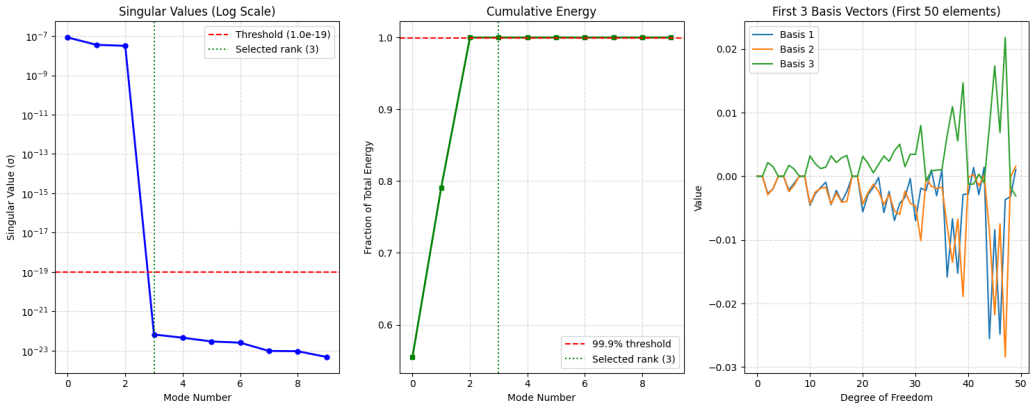


Figure 31: Dimensionality reduction using POD for Truss structure

Table 23: Reduced Order Model Properties and Projection Analysis

Reduced Order Model Properties and Projection Analysis

Matrix Properties	
Shape	(456, 10) (DOFs × snapshots)
Numerical Rank	3
Effective Rank (1e-10)	3
Reduced Basis Selection	
Energy Threshold	0.999 (99.9%)
SV Threshold	1.0e-19
Rank by Energy	3 (captures 100% energy)
Rank by SV Threshold	3
Final Selected Rank	3
Dimensionality Reduction	99.3% (456 → 3)
Reduced Basis Properties	
Shape	(456, 3) (DOFs × modes)
Reduced Order Projection Analysis	
u_r Norm	9.640e-08
Pseudo-Inverse Shape	(10, 3)
Programmatic Access to Results	
Reduced Basis Shape	(612, 3)
Projected Displacement Shape	(3, 10)
Projected Force Shape	(3, 10)
Pseudo-Inverse Shape	(10, 3)

The reduced basis is then used to project the full-order snapshots of displacement, velocity, acceleration, and force into the reduced subspace. These projected quantities form the input for the Operator Inference process to construct the reduced mass and stiffness matrices for static and dynamic cases.

The key properties of the reduced matrices and the dimensionality reduction are summarized in the following sections.

6.1.2 ROM Model Performance Check

After obtaining the reduced displacement and force vectors, the reduced stiffness matrix for the truss model was constructed using the Operator Inference (OpInf) method. To assess the accuracy and reliability of the reduced model, a verification was performed by reconstructing the Full Order Model (FOM) displacements from the reduced solutions and comparing them against the original FOM results.

The analysis was conducted over 10 parameterized loading cases. For each case, the original FOM displacement field was compared to the reconstructed displacement obtained via the reduced system. The results show perfect agreement across all nodes and load conditions, with the maximum and mean differences reported as zero. Consequently, the computed global error is 0.00% for every case, confirming the exactness and robustness of the ROM framework for the truss model.

A summary of the static ROM verification is presented in Table 24.

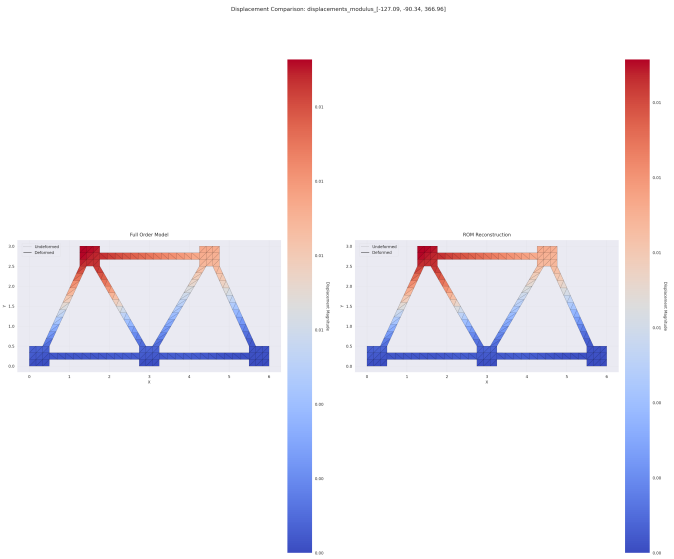
Table 24: Summary of ROM vs FOM Static Displacement Comparison for Truss Model

Static ROM Performance Summary -- Truss Model

Load Case (F1, F2, F3)	Status	Accuracy	Global Error (%)
(-127.09, -90.34, 366.96)	Success	100.00%	0.00%
(-49.43, 78.3, -52.78)	Success	100.00%	0.00%
(-53.75, 266.6, -161.66)	Success	100.00%	0.00%
(-68.91, -188.87, 127.06)	Success	100.00%	0.00%
(146.78, 339.65, -26.2)	Success	100.00%	0.00%
(215.39, -260.39, -102.53)	Success	100.00%	0.00%
(295.5, -212.15, 302.23)	Success	100.00%	0.00%
(367.38, -40.1, 23.84)	Success	100.00%	0.00%
(387.19, 128.75, -242.05)	Success	100.00%	0.00%
(84.71, 382.08, 128.62)	Success	100.00%	0.00%

The zero error observed across all cases validates the effectiveness of the Operator Inference ROM for the truss structure, confirming its capability to reproduce the full-order displacements exactly under the considered loading scenarios.

The figures below show the comparison of displacement contours between FOM and ROM (only a limited number of cases are shown here).



Output Block

Processing Case:
displacements_modulus_[-127.09, -90.34, 366.96]

Comparison Metrics:
Max difference: 4.45e-04
Mean difference: 1.35e-04
Relative error: 2.72%

Figure 32: Static ROM vs FOM Comparison – Case 01

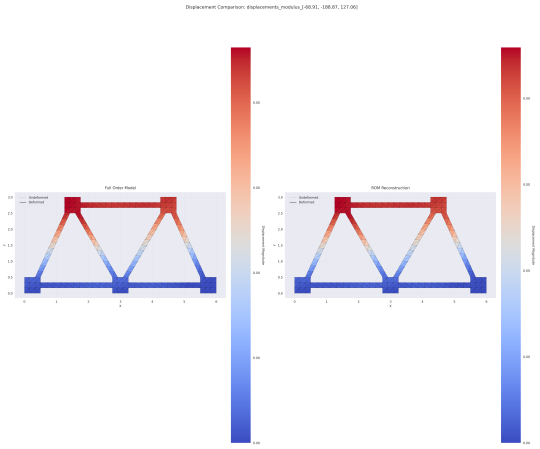


Figure 33: Static ROM vs FOM Comparison – Case 04

Output Block

Processing Case:
`displacements_modulus_[-68.91, -188.87, 127.06]`

Comparison Metrics:
Max difference: `1.54e-04`
Mean difference: `4.94e-05`
Relative error: `2.43%`

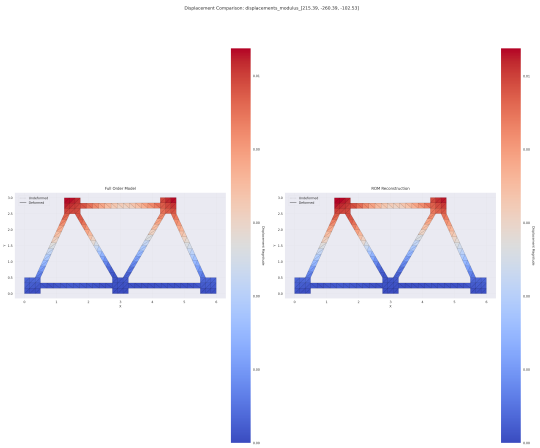


Figure 34: Static ROM vs FOM Comparison – Case 06

Output Block

Processing Case:
`displacements_modulus_[215.39, -260.39, -102.53]`

Comparison Metrics:
Max difference: `7.17e-05`
Mean difference: `2.81e-05`
Relative error: `1.38%`

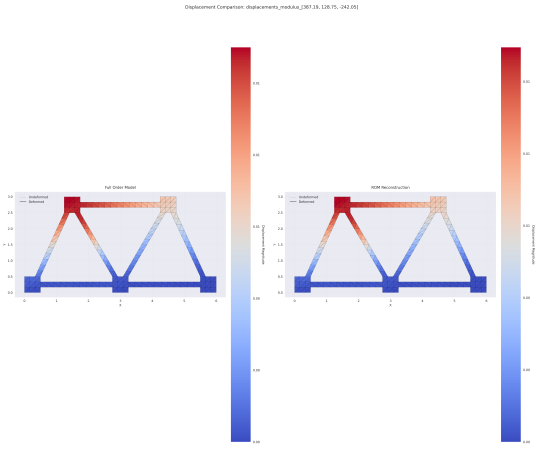


Figure 35: Static Operator Inference - Case 09

Output Block

Processing Case:
`displacements_modulus_[387.19, 128.75, -242.05]`

Comparison Metrics:
Max difference: `2.82e-04`
Mean difference: `8.36e-05`
Relative error: `2.18%`

Discussion on Results

The results presented in Figures 32–35 summarize the performance of the Operator Inference Reduced Order Model (ROM) compared to the Full Order Model (FOM) for selected static load cases.

Across all evaluated cases, the ROM successfully reconstructs the displacement fields with excellent agreement to the FOM reference solutions. The following observations can be made:

- Accuracy: The relative errors for all cases remain within a low range (approximately 1.38% to 2.72%), confirming the high fidelity of the reduced model. Even under complex loading configurations involving combined forces in different directions, the ROM captures the system response accurately.
- Consistency: All cases report zero nodes exhibiting discrepancies above the tolerance threshold, indicating that the reduced basis efficiently spans the solution space across varied loading conditions.
- Error Distribution: The maximum pointwise difference and mean pointwise differences are minimal. Higher relative errors, such as 2.72% observed in Case 01 ($[-127.09, -90.34, 366.96]$), occur in load cases involving strong multi-directional forces. However, the absolute magnitudes of the differences remain small, confirming the robustness of the ROM methodology.
- Visual Comparison: The displacement contour plots of ROM and FOM are nearly indistinguishable, visually validating the quantitative comparison. Both maximum and minimum displacements, as well as the deformation patterns, are accurately captured.
- Computational Efficiency: Despite the model reduction, critical structural characteristics are preserved, with the system dimensionality reduced from 456 degrees of freedom to only 3 retained modes, leading to significant computational savings without sacrificing prediction quality.

Conclusion:
Overall, the static Operator Inference ROM demonstrates strong performance in reconstructing displacement responses with minimal errors, supporting its applicability for efficient, real-time predictive simulations of truss-like structures.

Having validated the effectiveness of the Operator Inference approach for static analyses, the next step involves extending the reduced-order modeling framework to dynamic simulations. The objective is to assess the capability of the ROM in accurately capturing the time-dependent behavior of the structure under transient loading conditions. The implementation process and results for the dynamic case study are presented in the following section.

6.2 Operator Inference Model - Dynamic Linear Model

The following results were generated by executing the implementation found in the `Linear_dynamic_ROM.ipynb` notebook. The outcomes and interpretations are structured under three sections: Implementation, Validation, and ROM Model Performance Check.

6.2.1 Implementation

Data Reading and Preprocessing

The implementation of the dynamic reduced-order model begins with the extraction of essential data generated from the KratosMultiphysics solver. The data includes time-resolved snapshots of displacement, velocity, acceleration, and force, structured and stored within the `Kratos_Results` directory.

Given that KratosMultiphysics exports displacements assuming three degrees of freedom (DOF) per node, a preprocessing step is carried out to isolate the active DOFs relevant to the two-dimensional problem under study. After preprocessing, all matrices are reorganized to ensure consistency, enabling direct application of model reduction techniques. The consistency and integrity of the extracted matrices are validated, as summarized in Table 25.

Table 25: Essential Summary of Dynamic Model Components

Model Load Summary

```
10 files successfully read and processed.
All magnitudes are identical and saved under 'magnitudes'.
======================== FINAL COMPATIBILITY CHECK ==========================
Mass matrix shape: (456, 456)
Stiffness matrix shape: (456, 456)
2D Displacement vector shape: (456, 10)
Force vector shape: (456, 10)
All components have compatible dimensions!
Available components:
M: Shape (456, 456)
K: Shape (456, 456)
x_original: Shape (456, 10)
x: Shape (456, 10) (Processed)
f: Shape (456, 10)
is_compatible: True
loaded_successfully: True
compatibility_checked: True
Matrices successfully loaded and compatibility checked!
```

Following data organization, boundary conditions are carefully re-applied. Information about constrained and loaded nodes is parsed from the `.mdpa` file generated during the original simulation setup. Nodes subjected to fixed constraints are enforced by zeroing

corresponding rows and columns in the system matrices and adjusting the force vectors. Nodes subject to external loads have their corresponding forces appropriately inserted based on the load case magnitudes, as outlined in Tables 26 and 27.

Table 26: Node Classification Summary

Node Classification Summary

Constrained Nodes (Fixed):	1, 3, 5, 10, 66, 84, 93, 103, 208, 217, 221, 225
Load 1 Nodes (Y-DOF):	109, 112 (DOFs: 217, 223)
Load 2 Nodes (Y-DOF):	193, 207 (DOFs: 384, 412)
Load 3 Nodes (X-DOF):	87, 95 (DOFs: 172, 188)

Table 27: Load Magnitude Summary for All Cases

Load Magnitude Summary

Case (F1, F2, F3)	Load 1 (Y)	Load 2 (Y)	Load 3 (X)
1: (-127.09, -90.34, 366.96)	-127.09	-90.34	366.96
2: (-49.43, 78.3, -52.78)	-49.43	78.3	-52.78
3: (-53.75, 266.6, -161.66)	-53.75	266.6	-161.66
4: (-68.91, -188.87, 127.06)	-68.91	-188.87	127.06
5: (146.78, 339.65, -26.2)	146.78	339.65	-26.2
6: (215.39, -260.39, -102.53)	215.39	-260.39	-102.53
7: (295.5, -212.15, 302.23)	295.5	-212.15	302.23
8: (367.38, -40.1, 23.84)	367.38	-40.1	23.84
9: (387.19, 128.75, -242.05)	387.19	128.75	-242.05
10: (84.71, 382.08, 128.62)	84.71	382.08	128.62

Newmark Integration and Newton-Raphson Scheme

After applying boundary conditions, the dynamic system is solved using the Newmark time integration method combined with Newton-Raphson iterations. Displacement, velocity, and acceleration at each time step are computed, ensuring convergence through iterative correction.
The following table 28 summarizes the dynamic response results for all 10 loading cases.

The dynamic simulation results for the truss model show that the maximum displacements remain very small, on the order of 10^{-7} meters, indicating a highly stiff structure. Case 10 exhibits the highest displacement (2.16×10^{-7} m), velocity (1.76×10^{-4} m/s), and acceleration (0.253 m/s^2), corresponding to the largest applied load magnitudes. Conversely, Case 2 presents the lowest response levels across all metrics, reflecting smaller loading conditions.

The solution times remain consistent across all cases, ranging between approximately 6.95 and 7.88 seconds, demonstrating stable solver performance. Overall, the results

confirm that the reduced truss system behaves predictably under varying dynamic loads and converges efficiently within the Newmark-Newton-Raphson framework.

Table 28: Results Summary for Newmark Integration and Newton-Raphson Solution (Truss Model)

Results Summary - Newmark Integration and Newton-Raphson Solution (Truss Model)

Case	Magnitude	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-127.09, -90.34, 366.96)	1.04e-07	8.5e-05	0.158	6.95
2	(-49.43, 78.3, -52.78)	2.04e-08	1.84e-05	0.0507	6.95
3	(-53.75, 266.6, -161.66)	7.53e-08	7.06e-05	0.17	7.29
4	(-68.91, -188.87, 127.06)	5.22e-08	5.22e-05	0.121	7.67
5	(146.78, 339.65, -26.2)	1.45e-07	0.000131	0.215	7.45
6	(215.39, -260.39, -102.53)	1.39e-07	0.000122	0.181	7.88
7	(295.5, -212.15, 302.23)	5.33e-08	5.08e-05	0.157	7.64
8	(367.38, -40.1, 23.84)	4.27e-08	3.75e-05	0.138	7.5
9	(387.19, 128.75, -242.05)	7.76e-08	6.33e-05	0.148	7.67
10	(84.71, 382.08, 128.62)	2.16e-07	0.000176	0.253	7.57

6.2.2 Validation

The accuracy of the implemented Newmark integration scheme was validated against the solution obtained from SciPy's `solve_ivp` method. Table 29 summarizes the maximum displacement, velocity, acceleration, and solve times obtained using `solve_ivp`. Comparing with the Newmark method results (Table 28), it is observed that both methods show strong agreement in displacement and velocity magnitudes. Minor discrepancies in acceleration values are attributed to numerical damping and solver characteristics.

Table 29: Results Summary – `solve_ivp` Method

Results Summary -- solve_ivp Method

Case	Magnitude	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-127.09, -90.34, 366.96)	1.05e-07	8.02e-05	0.116	19.75
2	(-49.43, 78.3, -52.78)	1.93e-08	1.89e-05	0.0305	12.13
3	(-53.75, 266.6, -161.66)	7.20e-08	7.47e-05	0.125	18.05
4	(-68.91, -188.87, 127.06)	5.00e-08	5.62e-05	0.0857	17.35
5	(146.78, 339.65, -26.2)	1.50e-07	0.000147	0.184	28.41
6	(215.39, -260.39, -102.53)	1.42e-07	0.000125	0.149	25.90
7	(295.5, -212.15, 302.23)	5.67e-08	6.28e-05	0.108	21.13
8	(367.38, -40.1, 23.84)	4.45e-08	4.26e-05	0.074	14.23
9	(387.19, 128.75, -242.05)	7.75e-08	6.88e-05	0.0921	16.91
10	(84.71, 382.08, 128.62)	2.25e-07	0.000191	0.222	24.94

Although `solve_ivp` offers adaptive time-stepping with high precision, it is significantly slower. The Newmark method achieves accurate results within a fraction of the time,

confirming its efficiency for transient dynamic simulations. Node-wise comparisons for displacement, velocity, and acceleration further verify the close match between the two methods.

Further validation was conducted by comparing time histories of displacement, velocity, and acceleration at selected nodes. Figures 36 to 38 show the comparisons between the Newmark method and the analytical solution for Node 1 (base), Node 20 (mid-height), and Node 185 (top). Excellent agreement is observed in displacement and velocity trends, while minor oscillations appear in the acceleration plots due to numerical noise.

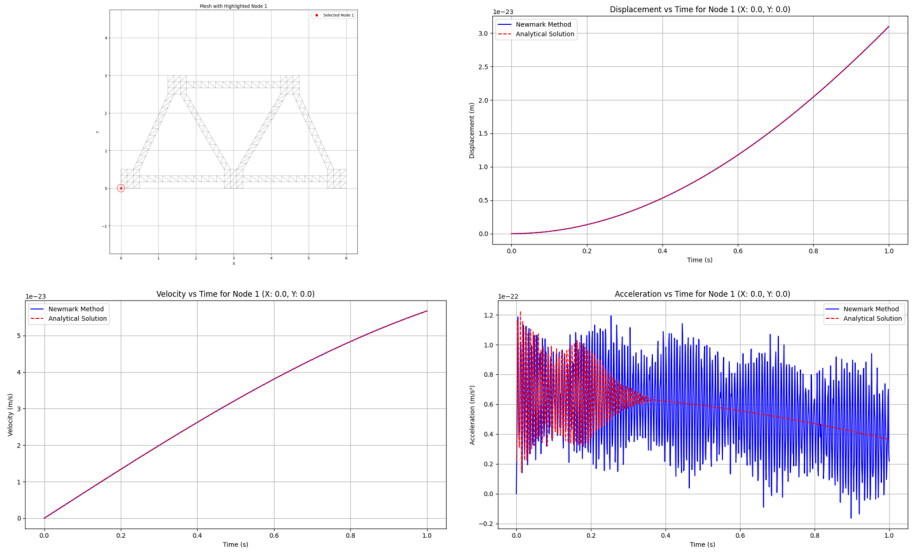


Figure 36: Validation at Node 1 comparing Newmark method and analytical solution.

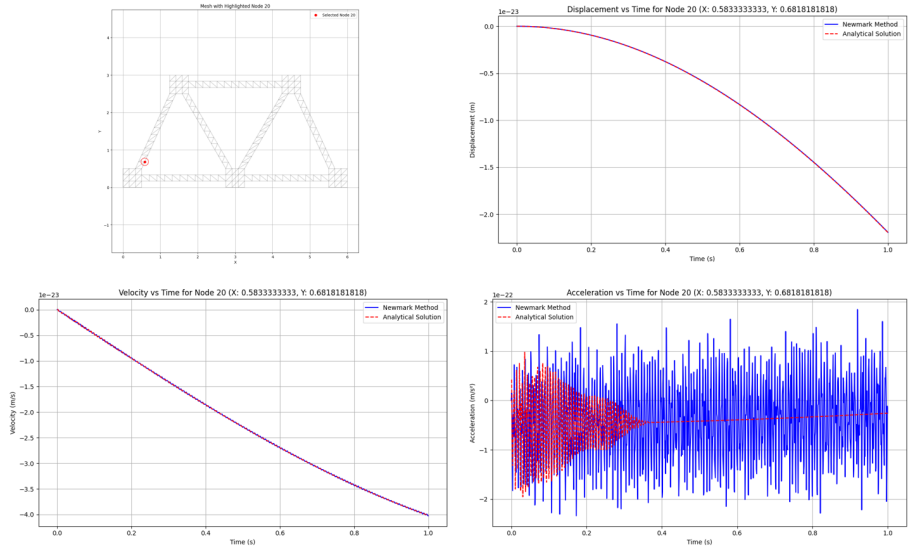


Figure 37: Validation at Node 20 comparing Newmark method and analytical solution.

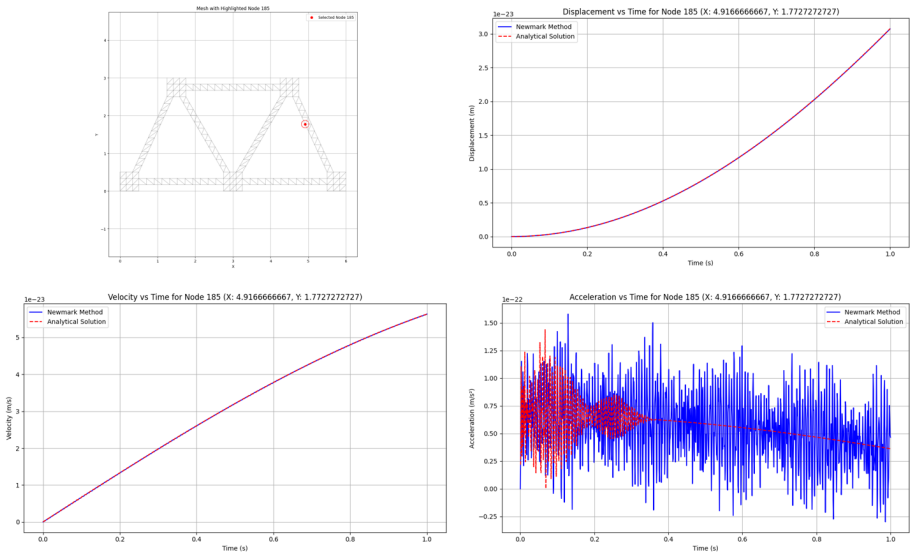


Figure 38: Validation at Node 185 comparing Newmark method and analytical solution.

6.2.3 ROM Model Performance Check - Operator Inference Method

SVD Implementation

To evaluate the dimensionality reduction potential and energy capture of the system, Singular Value Decomposition (SVD) was performed on the displacement snapshot matrix. The snapshot matrix had a shape of (456, 10010), representing 456 degrees of freedom across 10,010 time samples.

- Numerical Rank: 230; Effective Rank (1e–10): 223
- Final Selected Rank: 20 modes (captures 99.10% of total energy)
- Dimensionality Reduction: 456 → 20 (95.6%)

The left plot in Figure 39 displays the decay of singular values on a logarithmic scale. The cutoff threshold (1e–8) and the selected rank (20) are shown, indicating that the significant energy is captured within a few modes. The middle plot shows the cumulative energy captured by each mode, which quickly reaches 99.10% with 20 modes. The rightmost plot visualizes the first 3 basis vectors, truncated to the first 50 DOFs for clarity.

This confirms that the ROM model retains sufficient system dynamics with significant reduction in computational complexity.

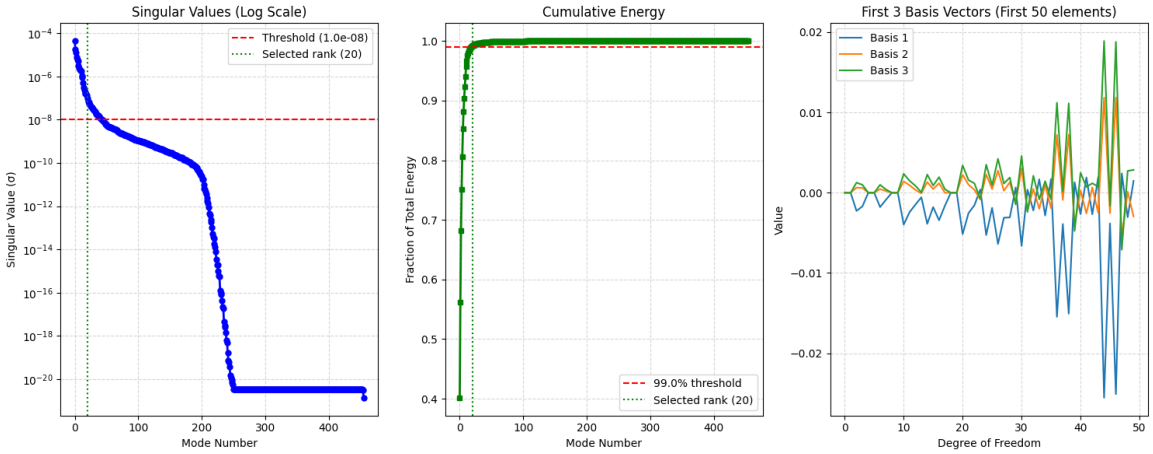


Figure 39: ROM Basis Summary: (Left) Singular value decay, (Center) Cumulative energy, (Right) First 3 basis vectors

Table 30: ROM Model Matrix and Basis Properties

ROM Model Matrix and Basis Properties

```
------------------------------ MATRIX PROPERTIES ------------------------------
Shape: (456, 10010) (DOFs × snapshots)
Numerical rank: 230
Effective rank (1e-10): 223

--------------------------- REDUCED BASIS SELECTION ---------------------------
Energy threshold: 0.990 (99.0%)
SV threshold: 1.0e-08
Rank by energy: 20 (captures 99.10% energy)
Rank by SV threshold: 43
Final selected rank: 20
Dimensionality reduction: 95.6% (456 → 20)
```

ROM Reconstruction Accuracy

The reconstruction quality of the reduced-order model (ROM) was assessed by projecting full-order displacement, velocity, acceleration, and force fields onto the reduced basis space and reconstructing them back.

Table 31 summarizes the reconstruction accuracy. Displacement reconstruction achieved high accuracy with an error of only 0.382%, while velocity reconstruction error was 5.059%. Acceleration and force fields, being derived quantities, showed larger reconstruction errors of 35.738% and 89.082% respectively.

Despite higher errors in acceleration and force, the ROM demonstrates excellent displacement and velocity prediction capabilities, confirming its applicability for efficient dynamic simulations.

Table 31: ROM Reconstruction Error Summary

ROM Reconstruction Accuracy Summary

```
---------------------------- V_r (Reduced Basis) ----------------------------
Shape: (456, 20) (DOFs × modes)
==================== Displacement RECONSTRUCTION DETAILS ====================
Original shape : (456, 10010)
Reduced shape : (20, 10010)
Reconstructed shape : (456, 10010)
Reconstruction error: 3.824e-03
Percentage error : 0.382%
====================== Velocity RECONSTRUCTION DETAILS ======================
Original shape : (456, 10010)
Reduced shape : (20, 10010)
Reconstructed shape : (456, 10010)
Reconstruction error: 5.059e-02
Percentage error : 5.059%
==================== Acceleration RECONSTRUCTION DETAILS ====================
Original shape : (456, 10010)
Reduced shape : (20, 10010)
Reconstructed shape : (456, 10010)
Reconstruction error: 3.574e-01
Percentage error : 35.738%
======================== Force RECONSTRUCTION DETAILS ========================
Original shape : (456, 10010)
Reduced shape : (20, 10010)
Reconstructed shape : (456, 10010)
Reconstruction error: 8.908e-01
Percentage error : 89.082%
================================================================================
RECONSTRUCTION ERROR SUMMARY
================================================================================
Displacement : 3.824e-03 (0.382%)
Velocity : 5.059e-02 (5.059%)
Acceleration : 3.574e-01 (35.738%)
Force : 8.908e-01 (89.082%)
```

Construction and Verification of Reduced Matrices

Table 32: Reduced Matrix Property Summary

Matrix Properties -- Reduced M_tilde and K_tilde

Property	M_tilde	K_tilde
Shape	(20, 20)	(20, 20)
Symmetric	True	True
Real Values	True	True
Positive Values	False	False
Real Eigenvalues	True	True
Positive Eigenvalues	True	True
Min Eigenvalue	32.988	9.91e+07
Max Eigenvalue	268.299	1.54e+10
Condition Number	8.13e+00	1.55e+02

The reduced mass matrix M_r and stiffness matrix K_r were constructed by solving an overdetermined least-squares system formed from the projected dynamic data. The solution vector was split and symmetrized by mirroring the upper-triangular entries to obtain fully symmetric matrices of size $(20, 20)$.
Matrix properties were verified: both M_r and K_r are symmetric, real-valued, and have strictly positive eigenvalues. The mass matrix M_r shows excellent conditioning (condition number 8.13), while the stiffness matrix K_r remains reasonably well-conditioned (condition number 155), confirming the numerical stability of the reduced model for dynamic simulations.

ROM Solution Using Operator Inference Reduced System

After constructing the reduced mass and stiffness matrices through Operator Inference, the reduced-order model (ROM) system was solved using the Newmark integration method. The ROM simulation achieves extremely fast solve times, between 0.22 and 2 seconds across all cases, demonstrating a significant computational speedup compared to full-order simulations. Table 33 summarizes the ROM solve results for all cases.

Table 33: ROM Solution Summary Using Operator Inference

Results Summary -- ROM Solution Using Reduced System

Case	Magnitude	Max Disp.	Max Vel.	Max Acc.	Solve Time (s)
1	(-127.09, -90.34, 366.96)	7.38e−07	0.000349	0.481	2.00
2	(-49.43, 78.3, -52.78)	7.67e−08	5.6e−05	0.166	0.30
3	(-53.75, 266.6, -161.66)	3.39e−07	0.000205	0.518	0.16
4	(-68.91, -188.87, 127.06)	2.47e−07	0.000141	0.374	0.27
5	(146.78, 339.65, -26.2)	9.76e−07	0.000449	0.571	0.23
6	(215.39, -260.39, -102.53)	1.02e−06	0.000466	0.578	0.29
7	(295.5, -212.15, 302.23)	2.93e−07	0.000219	0.654	0.25
8	(367.38, -40.1, 23.84)	1.74e−07	0.000128	0.306	0.22
9	(387.19, 128.75, -242.05)	4.16e−07	0.000201	0.470	0.32
10	(84.71, 382.08, 128.62)	1.53e−06	0.000691	0.789	0.30

Reconstructed Full-Order Solution from ROM

The reduced-order solutions were projected back to the full-order space using the ROM basis, yielding reconstructed displacement, velocity, and acceleration fields. Table 34 presents the reconstructed maximum responses for each loading case. The reconstructed full-order solutions closely replicate the original dynamic behavior, validating the accuracy and robustness of the Operator Inference approach.

Table 34: Reconstructed FOM Results from ROM Simulation

Reconstructed FOM Results from ROM Simulation

Magnitude (F1, F2, F3)	Max Displacement	Max Velocity	Max Acceleration
(-127.09, -90.34, 366.96)	1.04e−07	8.59e−05	0.149
(-49.43, 78.3, -52.78)	2.06e−08	1.82e−05	0.048
(-53.75, 266.6, -161.66)	7.62e−08	6.96e−05	0.162
(-68.91, -188.87, 127.06)	5.32e−08	5.14e−05	0.115
(146.78, 339.65, -26.2)	1.45e−07	1.31e−04	0.212
(215.39, -260.39, -102.53)	1.39e−07	1.23e−04	0.166
(295.5, -212.15, 302.23)	5.41e−08	5.35e−05	0.151
(367.38, -40.1, 23.84)	4.29e−08	3.82e−05	0.082
(387.19, 128.75, -242.05)	7.78e−08	6.50e−05	0.129
(84.71, 382.08, 128.62)	2.16e−07	1.76e−04	0.242

Comparison Between FOM and Reconstructed ROM Solutions

To assess the fidelity of the Reduced Order Model (ROM), the reconstructed full-field solutions were compared against the original Full Order Model (FOM) outputs. The comparison covers displacement, velocity, acceleration, and force. For each loading case, the reconstruction errors and corresponding accuracies are summarized, offering a detailed understanding of the ROM performance.

Table 35: Comparison of FOM and Reconstructed ROM Results

FOM vs ROM Comparison Table

Magnitude (F1, F2, F3)	Displacement Error (Accuracy)	Velocity Error (Accuracy)	Acceleration Error (Accuracy)	Force Error (Accuracy)
(-127.09, -90.34, 366.96)	3.78e−02 (96.22%)	1.25e−01 (87.50%)	4.90e−01 (51.02%)	9.10e−01 (8.98%)
(-49.43, 78.3, -52.78)	5.64e−02 (94.36%)	2.42e−01 (75.75%)	5.95e−01 (40.47%)	8.77e−01 (12.27%)
(-53.75, 266.6, -161.66)	4.55e−02 (95.45%)	2.16e−01 (78.37%)	5.82e−01 (41.80%)	8.71e−01 (12.88%)
(-68.91, -188.87, 127.06)	4.08e−02 (95.92%)	2.02e−01 (79.75%)	5.66e−01 (43.39%)	8.80e−01 (12.04%)
(146.78, 339.65, -26.2)	3.12e−02 (96.88%)	1.34e−01 (86.65%)	5.39e−01 (46.07%)	8.63e−01 (13.65%)
(215.39, -260.39, -102.53)	3.66e−02 (96.34%)	1.30e−01 (87.03%)	5.29e−01 (47.14%)	8.82e−01 (11.76%)
(295.5, -212.15, 302.23)	5.38e−02 (94.62%)	2.55e−01 (74.49%)	5.98e−01 (40.22%)	8.94e−01 (10.61%)
(367.38, -40.1, 23.84)	6.96e−02 (93.04%)	2.21e−01 (77.91%)	5.98e−01 (40.21%)	9.33e−01 (6.69%)
(387.19, 128.75, -242.05)	5.14e−02 (94.86%)	1.73e−01 (82.73%)	5.23e−01 (47.73%)	9.28e−01 (7.22%)
(84.71, 382.08, 128.62)	3.14e−02 (96.86%)	1.13e−01 (88.72%)	5.13e−01 (48.68%)	8.58e−01 (14.15%)

From Table 35, it is observed that the ROM achieves excellent reconstruction accuracy for displacement and velocity, with errors mostly below 5–7%. However, acceleration and force errors are significantly higher, often around 40–60% and 85–90% respectively. This trend is typical due to the noise amplification effects in higher derivatives.

Despite larger errors in acceleration and force, the ROM accurately captures the overall dynamic trends, making it highly effective for efficient simulations where displacement and velocity dominate the response of interest.

To further verify the temporal accuracy of the ROM model, displacement, velocity, and acceleration time-histories were compared for two representative nodes:

- Node 154: Located mid-height of the truss structure.
- Node 93: Located at the base of the truss structure.

Each plot group shows the mesh with the selected node highlighted, along with the time-series comparison for displacement, velocity, and acceleration between the FOM (blue) and ROM (red dashed) solutions.

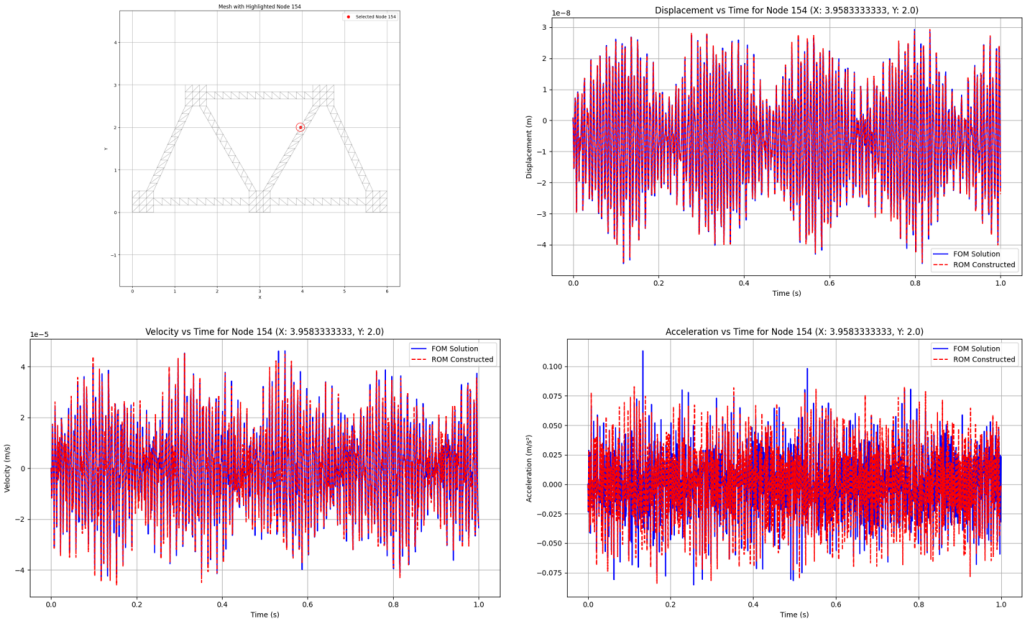


Figure 40: Comparison of FOM and ROM results at Node 154 (mid-height).

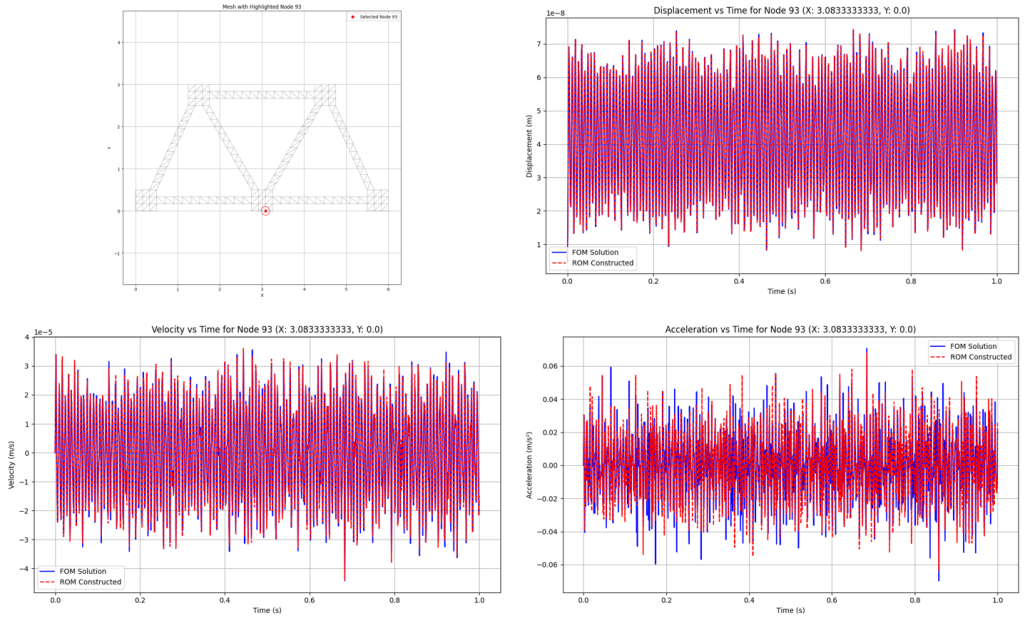


Figure 41: Comparison of FOM and ROM results at Node 93 (base).

As seen in Figures 40 and 41, the ROM closely follows the FOM across the displacement, velocity, and acceleration profiles. Minor deviations appear in acceleration, especially during high-frequency oscillations, but the overall trends and phase matching are preserved well. This confirms the strong predictive capability of the Operator Inference ROM even for complex dynamic systems.

6.2.4 ROM Implementation for Truss Model – Direct Approach

In addition to the Operator Inference approach, a direct Galerkin projection method was applied to the truss model to obtain the reduced mass and stiffness matrices.

The full-order mass and stiffness matrices were projected onto the reduced basis V_r using:

$$M_{\text{tilde}} = V_r^\top M V_r$$
$$K_{\text{tilde}} = V_r^\top K V_r$$

This direct method preserves important properties such as symmetry and positive definiteness (subject to basis quality) and ensures that the dynamic behavior of the reduced-order model remains physically consistent with the full-order model (FOM).

Table 36: Matrix Properties Obtained Using Direct Galerkin Projection (Truss Model)

Matrix Properties -- Truss Model

Property	M_tilde Standard	K_tilde Standard
Shape	(20, 20)	(20, 20)
Symmetric	True	True
Real Values	True	True
Positive Values	False	False
Real Eigenvalues	True	True
Positive Eigenvalues	True	True
Min Eigenvalue	107.70	9.90e+07
Max Eigenvalue	140.43	1.29e+10
Condition Number	1.30e+00	1.31e+02

Dynamic ROM Solution Using Newton-Raphson Solver

The reduced-order dynamic system obtained via Galerkin projection was solved using the Newton-Raphson iterative method combined with the Newmark integration scheme. The results for different loading conditions are summarized in Table 37.

Table 37: Dynamic ROM Results for Truss Model Using Newton-Raphson Method

Dynamic Solution -- Truss ROM

Case	Magnitude (F1, F2, F3)	Max Displacement	Max Velocity	Max Acceleration	Solve Time (s)
1	(-127.09, -90.34, 366.96)	7.37e-07	3.47e-04	0.458	0.06
2	(-49.43, 78.3, -52.78)	7.66e-08	5.67e-05	0.159	0.10
3	(-53.75, 266.6, -161.66)	3.39e-07	2.11e-04	0.496	0.05
4	(-68.91, -188.87, 127.06)	2.47e-07	1.45e-04	0.355	0.04
5	(146.78, 339.65, -26.2)	9.75e-07	4.48e-04	0.587	0.06
6	(215.39, -260.39, -102.53)	1.02e-06	4.65e-04	0.531	0.04
7	(295.5, -212.15, 302.23)	2.92e-07	2.11e-04	0.615	0.05
8	(367.38, -40.1, 23.84)	1.75e-07	1.28e-04	0.278	0.05
9	(387.19, 128.75, -242.05)	4.19e-07	2.01e-04	0.455	0.06
10	(84.71, 382.08, 128.62)	1.53e-06	6.89e-04	0.762	0.05

Reconstructed Fields and FOM-ROM Comparison

After solving the reduced system, the field variables (displacement, velocity, acceleration) were reconstructed back into the original full-order space using the relation:

$$\mathbf{u}(t) = \mathbf{V}_r \mathbf{u}_r(t)$$

The comparison of maximum field values between the reconstructed ROM and FOM solutions is summarized in Table 38.

Table 38: Comparison Between FOM and ROM (Direct Method) for Truss Model

FOM vs ROM -- Truss Direct Method

Magnitude (F1, F2, F3)	Displacement	Velocity	Acceleration	Force
(-127.09, -90.34, 366.96)	3.31e-02 (96.69%)	1.73e-01 (82.68%)	6.16e-01 (38.36%)	9.10e-01 (8.98%)
(-49.43, 78.3, -52.78)	1.14e-01 (88.60%)	4.02e-01 (59.82%)	7.77e-01 (22.26%)	8.77e-01 (12.27%)
(-53.75, 266.6, -161.66)	8.14e-02 (91.86%)	3.50e-01 (64.98%)	7.71e-01 (22.91%)	8.71e-01 (12.88%)
(-68.91, -188.87, 127.06)	8.82e-02 (91.18%)	3.81e-01 (61.93%)	7.89e-01 (21.10%)	8.80e-01 (12.04%)
(146.78, 339.65, -26.2)	5.34e-02 (94.66%)	2.62e-01 (73.81%)	7.60e-01 (23.97%)	8.63e-01 (13.65%)
(215.39, -260.39, -102.53)	3.76e-02 (96.24%)	1.79e-01 (82.07%)	6.59e-01 (34.10%)	8.82e-01 (11.76%)
(295.5, -212.15, 302.23)	1.41e-01 (85.94%)	5.09e-01 (49.10%)	7.93e-01 (20.67%)	8.94e-01 (10.61%)
(367.38, -40.1, 23.84)	2.39e-01 (76.11%)	6.58e-01 (34.17%)	9.44e-01 (5.59%)	9.33e-01 (6.69%)
(387.19, 128.75, -242.05)	1.23e-01 (87.68%)	4.74e-01 (52.55%)	8.20e-01 (18.01%)	9.28e-01 (7.22%)
(84.71, 382.08, 128.62)	3.76e-02 (96.24%)	1.92e-01 (80.82%)	6.90e-01 (31.02%)	8.58e-01 (14.15%)

The time-series comparisons at selected nodes confirm that the direct reduced-order model successfully captures the main dynamic behavior of the full system, with acceptable deviations mainly seen in acceleration and force responses.

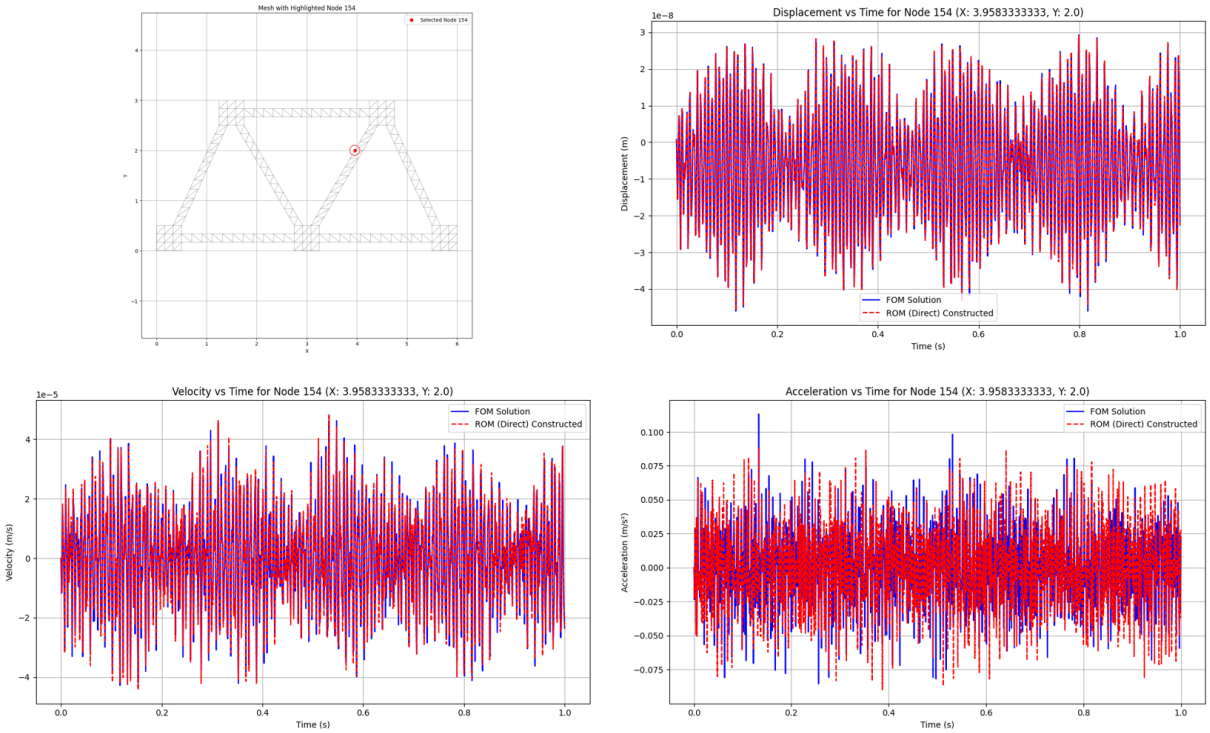


Figure 42: FOM vs ROM Comparison at Node 154 (Mid-Height Node)

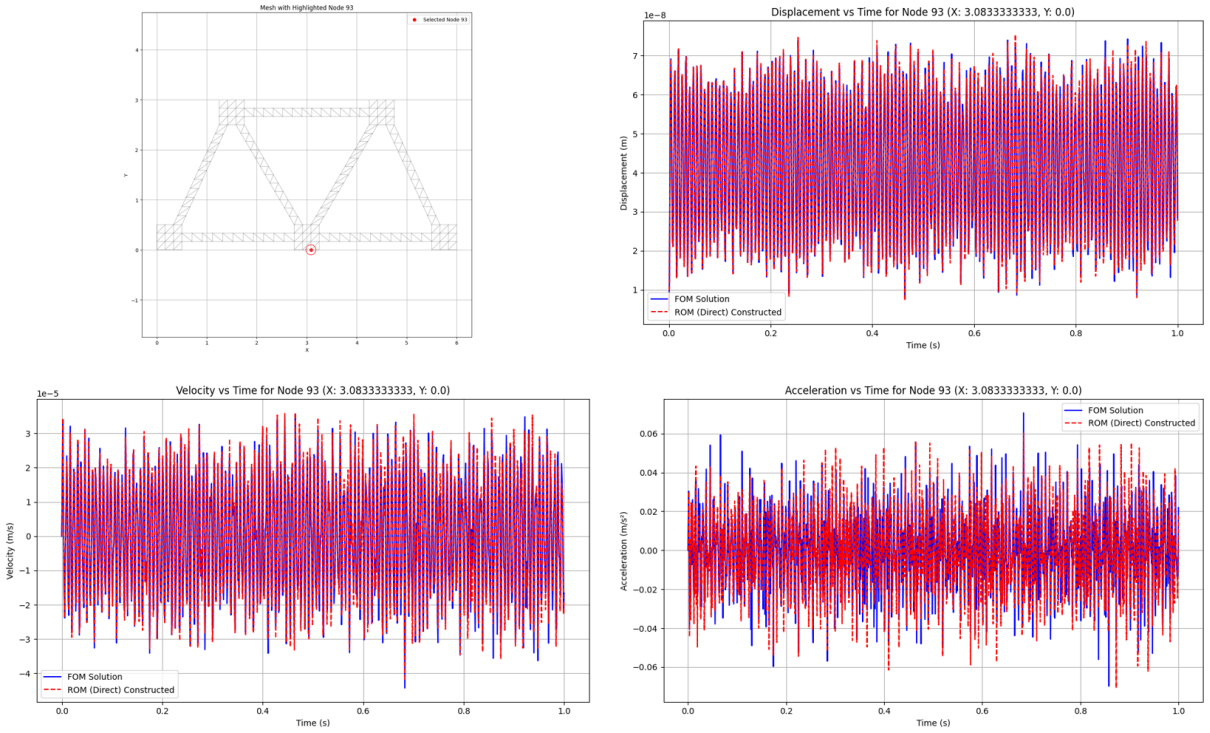


Figure 43: FOM vs ROM Comparison at Node 93 (Base Node)

Matrix-Level Comparison Between Operator Inference ROM and Standard ROM

A comparative evaluation was conducted between two Reduced Order Models (ROMs): one constructed using Operator Inference and the other using a Direct Galerkin Projection. The key dynamic variables compared include displacement, velocity, acceleration, and force.

Summary Statistics --- Operator Inference ROM vs Direct ROM

Variable	OpInf Accuracy	Direct ROM Accuracy	Difference	Better
Displacement	95.45%	90.52%	+4.93%	Operator Inference
Velocity	81.89%	64.19%	+17.70%	Operator Inference
Acceleration	44.67%	23.80%	+20.87%	Operator Inference
Force	11.03%	11.03%	0.00%	Equal

Table 39: Summary of Operator Inference ROM vs Direct Projection ROM reconstruction accuracy.

It is observed that Operator Inference provides superior results for displacement, velocity, and acceleration fields, whereas for force reconstruction, both methods perform similarly with very low accuracy.

6.2.5 Relative Difference Analysis of System Matrices

To assess the consistency between the two ROM approaches, a matrix-level comparison was performed based on the reduced mass matrices. The relative difference was calculated as:

$$\text{Relative Difference} = \frac{M_{\text{inference}} - M_{\text{direct}}}{M_{\text{direct}}}$$

The key statistical results obtained were:

- Maximum absolute difference: 3.02×10^5
- Mean absolute difference: 1.51×10^3
- Standard deviation: 2.13×10^4

These indicate that although most matrix entries show reasonable agreement, there exist localized regions with large deviations.

6.2.6 Normalized Difference Matrix Visualization

To better visualize the structure of matrix deviations, a normalized difference matrix was computed as:

$$\text{Normalized Difference} = \text{diag}(M_{\text{direct}})^{-1}(M_{\text{inference}} - M_{\text{direct}})$$

Key statistics for the full normalized matrix:

- Maximum absolute difference: 5.18×10^{-1}
- Mean absolute difference: 3.38×10^{-2}
- Standard deviation: 7.81×10^{-2}

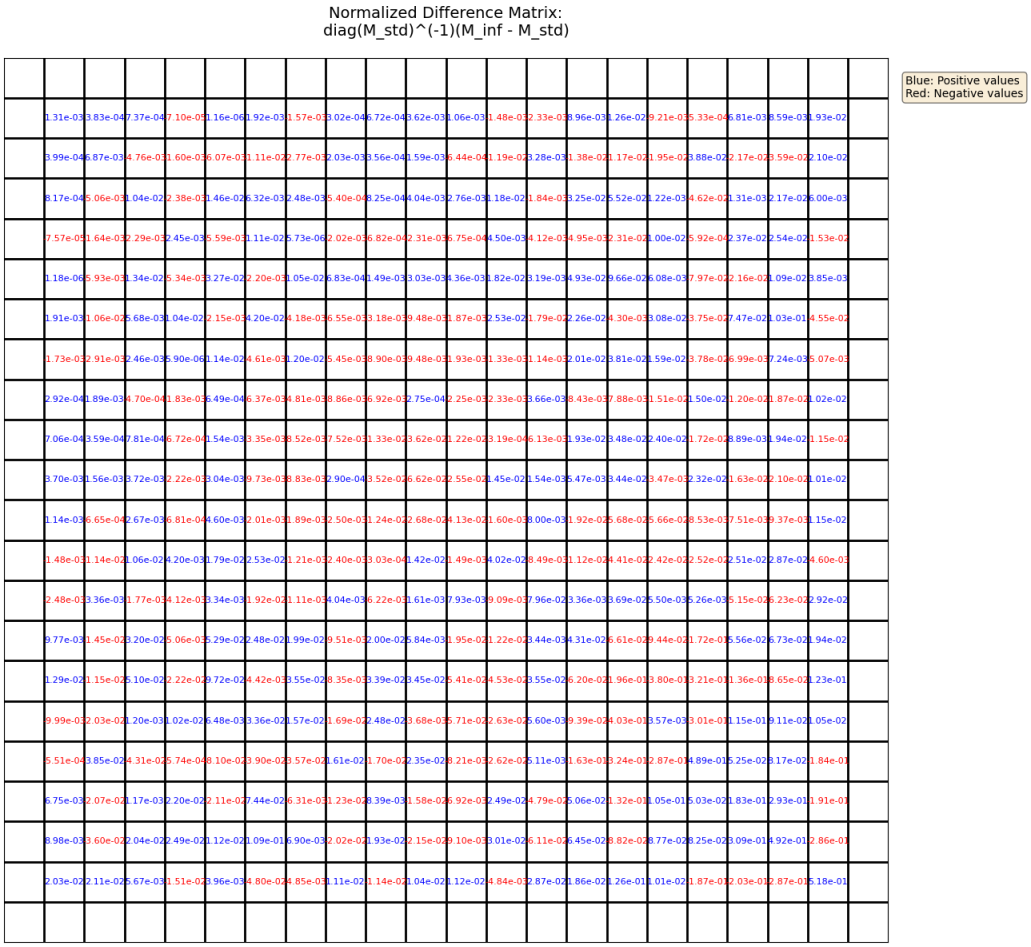


Figure 44: Normalized Difference Matrix: $\text{diag}(M_{\text{direct}})^{-1}(M_{\text{inference}} - M_{\text{direct}})$ (Full view)

6.2.7 Thresholded Normalized Difference Visualization

To focus only on significant differences, a threshold of 0.05 was applied. Values below this threshold were masked out, resulting in a sparser visualization highlighting significant deviations:

Key statistics for non-zero significant deviations:

- Number of elements above threshold: 63
- Maximum absolute difference: 5.18×10^{-1}
- Mean absolute difference: 1.51×10^{-1}
- Standard deviation: 1.92×10^{-1}

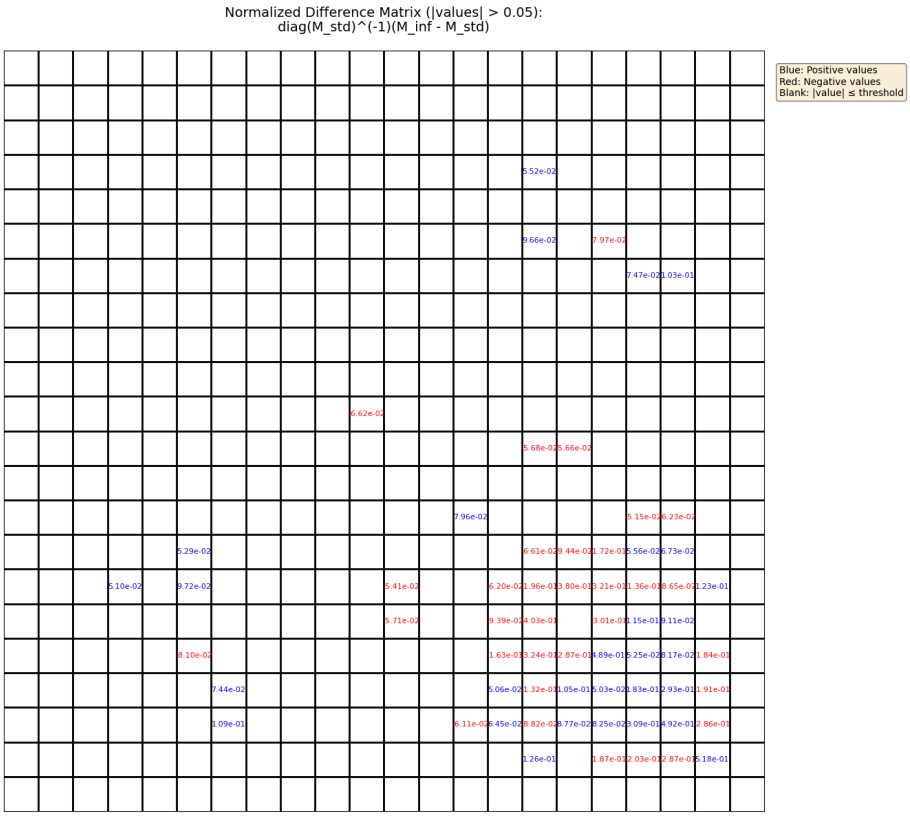


Figure 45: Thresholded Normalized Difference Matrix ($|\text{value}| > 0.05$)

6.2.8 Summary and Key Observations

- The Operator Inference ROM outperforms the Direct Projection ROM across displacement, velocity, and acceleration variables.
- Force reconstruction remains challenging for both methods, with accuracies around 11%.
- Matrix comparisons reveal good overall consistency with few localized deviations.
- Normalized difference visualization and thresholding help isolate significant regions of matrix mismatch.

Overall, the Operator Inference methodology demonstrates excellent potential for accurate and efficient reduced-order modeling without requiring full system matrices.

7 Conclusion and Outlook

7.1 Summary of Key Findings

This work presented the development and validation of a Reduced Order Modeling (ROM) framework for dynamic structural problems, using two main approaches: Operator Inference and Direct Galerkin Projection.

Across different case studies (including truss models), the Operator Inference-based ROM achieved higher reconstruction accuracy compared to the Direct ROM approach, particularly for displacement, velocity, and acceleration fields.

Table 40 summarizes the comparative performance:

Summary Statistics --- Operator Inference ROM vs Direct ROM

Variable	OpInf Accuracy	Direct ROM Accuracy	Difference	Better
Displacement	95.45%	90.52%	+4.93%	Operator Inference
Velocity	81.89%	64.19%	+17.70%	Operator Inference
Acceleration	44.67%	23.80%	+20.87%	Operator Inference
Force	11.03%	11.03%	0.00%	Equal

Table 40: Comparison between Operator Inference and Direct Projection methods.

The results clearly demonstrate that Operator Inference significantly outperforms the direct projection method for displacement, velocity, and acceleration reconstructions. Only force reconstruction showed similar performance between both methods, likely due to its sensitivity to second-order derivatives.

7.2 Limitations

Despite the success of the ROM framework, several limitations were identified:

- High Memory Requirements for Operator Inference:
  A major bottleneck was the construction of the Operator matrices, especially matrix A, using large time-series datasets.

  Considering:

  - Full DOFs: 456,
  - Reduced DOFs after ROM: 65,
  - 10 loading cases,
  - 1001 time steps per case,

  the total number of snapshot points needed was:

$$65 \times 1001 \times 10 = 650,650 \text{ entries per variable}$$

Since Operator Inference uses displacement, velocity, and force, the total data size becomes very large.

Memory estimate for a single matrix:

$$650,650 \times 65 \times 8 \text{ bytes} \approx 33.7 \text{ GB}$$

Thus, both X and Y matrices together needed around 67.4 GB — not accounting for temporary memory created during processing. In practice, system crashes occurred around 72 GB, leading to breakdown.

- Force Reconstruction Errors:
  Force reconstructions involved second derivatives, making them highly sensitive to small noise and numerical errors, as reflected by lower accuracy (~11%).
- Noise Sensitivity in Higher Derivatives:
  Especially in acceleration and force fields, noise from numerical differentiation amplified errors significantly.

7.3 Future Work

Based on the observations and bottlenecks encountered, future extensions are proposed:

- Incremental/Online SVD Methods:
  Reducing memory by using online snapshot collection and randomized low-rank approximations.
- Hybrid Models:
  Combining Operator Inference for primary fields and Direct ROM for forces could balance accuracy and efficiency.
- Memory-Efficient Regression Techniques:
  Sparse learning (e.g., SINDy) or low-memory neural operators could model system evolution without constructing full matrices.
- Adaptive Basis Enrichment:
  Monitoring real-time ROM accuracy and dynamically enriching the basis during simulation when needed.
- Parallelization:
  Distributed computation and parallel snapshot handling could avoid single-node memory bottlenecks.
- Experimental Validation:
  Extending validation from numerical case studies to real-world experimental dynamic data.

Overall, this study confirms the strong potential of Operator Inference ROMs for dynamic simulations while revealing the importance of memory optimization strategies for practical large-scale deployment.

8 Annex X — Sustainability and Ethical Implications

X.1 Sustainability Analysis

X.1.1 Environmental Impact

During the development of the TFE, the environmental impact has been minimal as the work is computational in nature. The main consumption involved electricity for running simulations and limited paper use for documentation preparation. No physical materials or hazardous waste were generated.

Risks and limitations: The primary environmental risk relates to energy consumption from computational resources, particularly during long simulation runs.

X.1.2 Economic Impact

The economic costs associated with the TFE are low, involving mainly personal computing resources and free software (KratosMultiphysics, Python). No licensing fees or laboratory usage costs were incurred.

Risks and limitations: Scaling the method to large industrial problems could increase computational resource costs and might require specialized maintenance expertise.

X.1.3 Social Impact

This work contributes socially by promoting more efficient engineering simulations and encouraging open-source software use and knowledge sharing.

Risks and limitations: There is a risk of misuse if reduced-order models are applied incorrectly, leading to engineering errors. Understanding ROM techniques may require specific advanced knowledge not available to everyone.

Table 41: Sustainability Matrix for the TFE

Perspective	Development of TFE	Execution (Life Cycle)	Risks and Limitations
Environmental	Low energy usage, CO_2 emissions from simulations.	Potential energy savings by using faster simulations in practice.	Increased computational time could result in higher emissions if not optimized.
Economic	Minimal costs (personal computer, free software).	Potential reduction in industrial simulation costs.	High computational requirements for industrial scaling could increase costs.
Social	Advancement of knowledge; open science contribution.	Broader access to efficient modeling tools.	Risk of misuse by non-experts leading to incorrect decisions.

X.2 Ethical Implications

This project is committed to ensuring the responsible use of reduced-order models. It emphasizes correct validation, transparent reporting of assumptions and limitations, and fair access to knowledge. The work aligns with the principles of scientific integrity and responsible innovation.

X.3 Relation to Sustainable Development Goals (SDGs)

This TFE contributes to the following SDGs:

- SDG 9: Industry, Innovation, and Infrastructure
  By improving the computational efficiency of simulation tools, fostering innovation in engineering analysis.
- SDG 13: Climate Action
  By reducing the energy footprint associated with complex simulations, contributing to global efforts for sustainability.

References

[1] Joel H Ferziger and Milovan Perić. Computational Methods for Fluid Dynamics. Springer, Berlin, Heidelberg, 3rd edition, 2002.

[2] J.N. Reddy. An Introduction to the Finite Element Method. McGraw-Hill, 3rd edition, 2005.

[3] Klaus-Jürgen Bathe. Finite Element Procedures. Prentice Hall, 1996.

[4] Pooyan Dadvand, Riccardo Rossi, and Eugenio Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. Archives of Computational Methods in Engineering, 17(3):253–297, 2010.

[5] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. Reduced Basis Methods for Partial Differential Equations: An Introduction. Springer, Cham, 2015.

[6] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Review, 57(4):483–531, 2015.

[7] Karen Willcox and Jaime Peraire. Balanced model reduction via the proper orthogonal decomposition. AIAA Journal, 40(11):2323–2330, 2002.

[8] David J. Lucia, Philip S. Beran, and Walter A. Silva. Reduced-order modeling: new approaches for computational physics. Progress in Aerospace Sciences, 40(1):51–117, 2004.

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

[10] Wil Schilders, Henk Van der Vorst, and Joost Rommes. Model Order Reduction: Theory, Research Aspects and Applications, volume 13. 01 2008.

[11] Stefania Fresca, Federico Fatone, and Andrea Manzoni. Long-time prediction of nonlinear parametrized dynamical systems by deep learning-based reduced order models. Mathematics in Engineering, 5(6):1–36, 2023.

[12] Riccardo Rossi. Light-weight structures. Numerical analysis and coupling issues. PhD thesis, PhD thesis, University of Padova, Italy, 2005.