# Study of compression techniques for partial differential equation solvers 

## Document:

Report

## Author:

Alex Fustagueras Gómez


## Director:

Joaquín Hernández Ortega

## Degree:

Bachelor's degree in Aerospace Technology Engineering

Examination session:
Spring 2023


## Acknowledgements

This thesis brings together the research carried out over months of hard work, where the interests of the author have been reflected in an attempt to face new challenges. The author extends his gratitude to Joaquín Hernández, the professor and tutor, for the freedom and trust provided, which have fostered creativity in an unfamiliar environment.

Additionally, it is worth saying that without the assistance of my family, this thesis would not have been feasible. They have consistently provided unconditional support in the pursuit of my academic career, and it is something that shall never be forgotten.

# Study of compression techniques for partial differential equation solvers 


#### Abstract

: Partial Differential Equations (PDEs) are widely applied in many branches of science, and solving them efficiently, from a computational point of view, is one of the cornerstones of modern computational science. The finite element (FE) method is a popular numerical technique for calculating approximate solutions to PDEs.

A not necessarily complex finite element analysis containing substructures can easily generate enormous quantities of elements that hinder and slow down simulations. Therefore, compression methods are required to decrease the amount of computational effort while retaining the significant dynamics of the problem.

In this study, it was decided to apply a purely algebraic approach. Various methods will be included and discussed, ranging from research-level techniques to other apparently unrelated fields like image compression, via the discrete Fourier transform (DFT) and the Wavelet transform or the Singular Value Decomposition (SVD).


Keywords: finite element method; reduced order model; partial differential equations; singular value decomposition; fourier transform

## Summary

In the fields of science and engineering, Partial Differential Equations (PDEs) play a fundamental role as a powerful tool for modeling diverse systems. They provide a means to describe the evolution of physical quantities in relation to independent variables such as space and time. However, the analytical solutions for PDEs are often elusive and challenging to obtain. As a result, the application of numerical approximation methods, particularly the finite element (FE) approach, becomes vital.

During FE simulations, significant amounts of data are generated, making dimensionality reduction techniques, whose implementation is known as reduced-order models (ROMs), very valuable for compressing the obtained information. This thesis explores several techniques for achieving such a reduction, and as its main goal, it sets out to implement at least one of the described techniques inside a code of FE and perform the analysis of one problem with the assistance of the professor. The initial focus is on the Singular Value Decomposition (SVD) an the Proper Orthogonal Decomposition (POD), widely utilized methods for effectively reducing data dimensions. Additionally, transform-based compression methods like the discrete Fourier transform (DFT) and wavelets, commonly employed in fields such as image compression, are explored as potential approaches for compressing equations derived from FE methods.

Furthermore, subsequent sections of the thesis provide insights into the application of these techniques in parametrized FE models. The particularities of employing reduced-order models and hyper-reduced-order models using techniques like the Empirical Cubature Method (ECM) are also discussed. Finally, a numerical assessment of a simple yet impactful problem is presented, demonstrating the effectiveness of these methods and opening paths for further exploration within the thesis. To conclude the thesis, the need for these methods is justified as a relevant contribution to advancements in the engineering community.

## Resum

En els camps de la ciència i de l'enginyeria, les Equacions en Derivades Parcials (EDPs) juguen un paper fonamental com una eina potent per a la modelització de sistemes. En aquest sentit, proporcionen un mitjà per descriure l'evolució de les magnituds físiques en relació amb variables independents com l'espai i el temps. No obstant això, les solucions analítiques de les EDPs sovint són difícils d'obtenir, el que requereix l'ús de mètodes d'aproximació numèrica com enfocaments amb elements finits (EF).

Durant simulacions d'EF, es generen grans quantitats de dades, cosa que fa que tècniques de reducció de dimensionalitat, conegudes com a models de reducció d'ordre (MRO), siguin molt valuoses per comprimir la informació obtinguda. Aquesta tesi explora diverses tècniques per aconseguir aquesta reducció i, com a objectiu principal, es proposa implementar almenys una de les tècniques descrites dins d'un codi amb elements finits i realitzar l'anàlisi d'un problema amb l'assistència del professor. L'enfocament inicial es centra en la Descomposició de Valors Singulars (DVS) i la Descomposició Ortogonal Pròpia (DOP), mètodes àmpliament utilitzats per reduir eficaçment les dimensions de les dades. A més, també s'exploren mètodes de compressió basats en transformacions com la transformada discreta de Fourier (TDF) i els wavelets, que s'utilitzen habitualment en àrees com la compressió d'imatges, però que son presentas com nous enfocaments per comprimir les equacions dels mètodes d'EF.

Les seccions següents de la tesi proporcionen una visió de l'aplicació d'aquestes tècniques en models parametritzats d'EF. També es discuteixen les particularitats de l'ús de models de reducció d'ordre i models d'ordre hiper-reduïts utilitzant tècniques com el Mètode de Cubatura Empírica (ECM). Finalment, es presenta una avaluació numèrica d'un problema senzill però de gran impacte, que demostra l'efectivitat d'aquests mètodes i obre camins per a investigacions futures de més profunditat. Per concloure la tesi, es justifica la necessitat d'aquests mètodes com una contribució rellevant als avenços de la comunitat d'enginyeria.

## Contents

List of Tables ..... II
List of Figures ..... III
List of Algorithms ..... IV

1. Introduction ..... 1
1.1. Aim ..... 1
1.2. Scope ..... 1
1.3. Requirements ..... 2
1.4. Justification ..... 2
1.5. Schedule ..... 4
1.5.1. Description of the tasks to be developed ..... 4
1.5.2. Interdependency relationships among tasks ..... 5
1.5.3. Work breakdown structure (WBS) ..... 6
1.5.4. Gantt chart ..... 7
2. State of the art ..... 8
2.1. Lossless and lossy compression ..... 9
2.2. Measures of performance ..... 9
2.3. Dimensionality reduction ..... 10
3. Compression algorithms ..... 12
3.1. Singular Value Decomposition ..... 12
3.1.1. General definition ..... 13
3.1.2. Intuitive interpretation ..... 14
3.1.3. Method computation ..... 16
3.1.4. Eigenfaces example ..... 19
3.2. Proper Orthogonal Decomposition ..... 23
3.2.1. Approximation basis functions ..... 23
3.2.2. General definition ..... 24
3.2.3. Approximation of surface example ..... 26
3.3. Fourier and Wavelet Transform ..... 28
3.3.1. Brief introduction ..... 28
3.3.2. General definition ..... 30
3.3.3. Discrete Fourier Transform ..... 31
3.3.4. Fast Fourier Transform ..... 33
3.3.5. Wavelet Transform ..... 34
4. Compression of equations ..... 37
4.1. General Overview ..... 37
4.1.1. Nodal Vector Approximation Approaches ..... 38
4.1.2. Integral Approximation Approaches ..... 38
4.2. Parametrized Finite Element Model ..... 39
4.3. First Reduction Stage ..... 40
4.3.1. Unknown Nodal Displacements ..... 40
4.3.2. Prescribed Nodal Displacements and External Forces ..... 40
4.3.3. Projection onto the Reduced-Order Space ..... 41
4.3.4. Internal Forces ..... 42
4.4. Second Reduction Stage ..... 42
4.4.1. Empirical Cubature Method ..... 43
4.5. Summary ..... 47
5. Numerical Assessment ..... 48
5.1. Performance Analysis ..... 55
6. Future research ..... 56
7. Concluding remarks ..... 57
8. Environmental implications ..... 59
9. References ..... 60
A. Code Implementation ..... 65
B. QR Factorization ..... 67
B.1. Classical Gram-Schmidt ..... 68
B.2. Modified Gram-Schmidt ..... 69
B.3. Householder Triangularization ..... 70
C. Empirical Cubature Method ..... 72
List of Tables
1.1. Table summary of all tasks to carry out along with their description ..... 4
1.2. Table summary of all tasks dependencies relationships ..... 5
1.3. Task activity duration for Figure 1.3. ..... 7
5.1. Comparison of the dimensions and computation time of the FE problem and the HROM, along with the corresponding compression ratios. ..... 55

## List of Figures

1.1. Average queue time versus the number of CPU cores available for the jobs on supercomputer PARAM Yuva II [3] ..... 3
1.2. Work breakdown structure of the project (Source: Own). ..... 6
1.3. Gantt chart of the project (Source: Own). ..... 7
2.1. Generalized schematic of compression and reconstruction algorithm, extracted from [7] ..... 8
2.2. Domain and subdomains interface in the commonly known first-order homog- enization [12, 13], extracted from [14] ..... 10
3.1. Decomposition of input matrix $\mathbf{A}$ into diagonal matrix of singular values $\Sigma$ and matrices of left and right singular vectors. Orange color illustrates low-rank approximation (Source: Own) ..... 13
3.2. Linear transformation of a vector in $\mathbb{R}^{m}$ to $\mathbb{R}^{n}$ via a rectangular matrix $\mathbf{A}$ (Source: Own) ..... 14
3.3. Left and right singular values of a rectangular matrix $\mathbf{A}, m=3$ and $n=2$ (Source: Own) ..... 15
3.4. Singular Value Decomposition formula dissection of a rectangular matrix $\mathbf{A}$, $m=3$ and $n=2$ (Source: Own). ..... 15
3.5. Geometric interpretation of the singular value decomposition of a 2 by 2 matrix $\mathbf{A}$, extracted from [37] ..... 16
3.6. Images obtained from the Extended Yale Face Database B in grey colormap (Source: Own) ..... 19
3.7. Schematic procedure to obtain eigenfaces from library of faces, extracted from [4] ..... 20
3.8. Images obtained from the Extended Yale Face Database B in grey colormap (Source: Own) ..... 21
3.9. Singular values of the data matrix $\mathbf{X}$ (Source: Own) ..... 21
3.10. Approximate representation of test image using eigenfaces basis of various order r. (Source: Own) ..... 22
3.11. (top-left) Representation of the actual surface $\mathbf{z}(x, t)$ ). (others) Rank approx- imation of the surface extracted using SVD (Source: own). ..... 26
3.12. SVD and POD analysis of the approximation performed (Source: Own) ..... 27
3.13. Change of coordinate of a vector in two dimensions, extracted from [4]. ..... 29
3.14. Discrete data sampled for the discrete Fourier transform, extracted from [4]. ..... 31
3.15. Illustration of resolution limitations and uncertainty in time-frequency analysis, extracted from [4] ..... 34
3.16. Three Haar wavelets for the first two levels of the multi-resolution in Figure 3.15 (d), extracted from [4]. ..... 36
5.1. Mesh characterization of the stress-plane beam in GID. (Source: Own). ..... 48
5.2. Boundary conditions applied to the geometry of the problem (Source: own) ..... 49
5.3. (right) SVD truncation error versus number of basis vectors employed in theapproximation $(n)$. (left) The portion between modes 4 and 8 is shown inmagnified form in logarithmic scale (Source: own)51
5.4. Truncation error versus number of modes (matrix of internal work snapshots) (Source: own) ..... 52
5.5. Deformed shapes corresponding to the first 4 dominant displacement modes $\left[\boldsymbol{\Phi}_{\mathbf{1}}, \boldsymbol{\Phi}_{\mathbf{2}}, \boldsymbol{\Phi}_{\mathbf{3}}, \boldsymbol{\Phi}_{\mathbf{4}}\right]$ (Source: own) ..... 52
5.6. Characterization of the ECM integration points (for $m=18$ ) (Source: Own). ..... 53
5.7. Location of the $m=18$ integration points chosen by the ECM algorithm (Source: own). ..... 53
5.8. Evolution over time of the variables of the 1st DOF. Results computed with the FE model and the ROM using $m=18$ and the full set of FE integration points (Source: own) ..... 54
5.9. Contour plot of normal 2nd Piola-Kirchhoff stress in the x-direction by the FE model (top) and the HROM (bottom) (Source: own). ..... 54
A.1. Organization of the repository (Source: own). ..... 65
A.2. Organization of the repository (Source: own). ..... 65

## List of Algorithms

1. Golub-Kahan Bidiagonalization . . . . . . . . . . . . . . . . . . . . . . . . . . 18
2. POD basis of rank $r$. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
3. First Reduction Stage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
4. Second Reduction Stage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
5. Classical Gram-Schmidt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
6. Modified Gram-Schmidt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
7. Householder QR Factorization . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
8. Empirical Cubature Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
9. Least-squares via one-rank update . . . . . . . . . . . . . . . . . . . . . . . . 73
10. Update of the inverse Hermitian Matrix when several columns are removed . 74
11. Update of the inverse of an Hermitian Matrix when one columns is removed . 74

## 1. Introduction

### 1.1. Aim

The goal of this project is to develop an in-house code containing at least one of the data methods and/or compression techniques usually applied to approximate complex nonlinear partial differential equations (PDEs) arising from real-world engineering problems.

Therefore, several methods will be explored throughout the thesis, ranging from basic introductory material to research-level strategies, combining techniques from other fields (e.g., image compression) to tackle previously unachievable problems brought by the new computational era.

### 1.2. Scope

The development of the project will include the study and implementation of the listed following tasks:

- Overview of several data-driven methods, applied optimization, and classical techniques approaches, with a quantitative and qualitative assessment of all available underlying algorithms for solving engineering problems.
- Study and development of an algorithm capable of using Principal Orthogonal Decomposition (POD) as one of the uses of Singular Value Decomposition (SVD).
- Study and comprehension of the parameterized finite element (FE) model and the current reduced-order modeling panorama, especially through the ECM, since it is the professor's main field of study.
- Study and validation of a finite element simulation code and its related reduction model with the assistance of the professor.
- Performing a compression performance comparison of the finite element simulation using the key ideas acquired in the study phase.

Moreover, the development of the project will NOT include:

- Development of a finite element simulation. The simulation data is being provided by the tutor's research group. The treatment will be carried out by the student.
- Development of SVD, Fourier and Wavelet transforms libraries and/or related internal functionalities of MATLAB's framework.
- Simulation tests with intricate meshes, containing many integration points, as the primary goal of the thesis is proving applicability, extension fields are outside the scope.


### 1.3. Requirements

The requirements of the project are divided into two categories: those concerning the constraints for the final solution and those concerning the procedure requirements that must be followed during the development of the thesis.

- Constraints for the final solution.
- Application of at least one of the compression techniques reviewed using finite element simulation data provided by the tutor's research group.
- The code must be developed using MATLAB software.
- Standard MATLAB libraries and functionalities will be implemented.
- All generated code must be easily readable and, if necessary, extendable.
- Use of the specialized software (GID) developed by the professor's research group for post-processing purposes.
- Procedure requirements.
- The bachelor final thesis project dedication limit is 300 hours.
- All academic documents developed during the entire thesis must be written in English.
- The GitHub online repository will be used to commit the code modifications. This tool provides easy access to the code and its various versions for the programmer, and it is good practice for future projects.
- All academic deliverables must be completed by June 21st.


### 1.4. Justification

Partial Differential Equations (PDEs) have been enormously successful as a tool for modeling processes both in science and engineering. PDEs explain how physical quantities or variables alter in relation to a number of independent variables, including space and time. By formulating PDEs that describe the underlying physics or dynamics of a system, one can gain insights into its behavior, make predictions, and analyze its properties. However, for the vast majority of geometries and problems, and due to the limited availability of analytical solutions, these PDEs cannot be solved using analytical methods [1].

Instead, a numerical approximation for solving the equations is constructed, generally based on several discretization methods, to appropriately capture the essence of the continuous PDE problem. These discretization approaches approximate the PDEs with numerical model equations that can be solved numerically. The numerical model equation solutions are, in turn, an approximation of the genuine PDE solutions. Such approximations are computed using methodologies such as the finite element method (FEM), which uses an element-based approach for the computation of the variables of the problem, as later sections of the work will discuss in more detail.

The issue arises from the fact that in order to capture the true dynamics and properties of a model, one must build very condensed meshes (dividing the domain into smaller regions) and/or a high number of elements, which in turn contain a large number of integration points that the approximation must address. In the field of computation, this is referred to as the "curse of dimensionality", where high-dimensional PDEs often require sophisticated numerical techniques, efficient algorithms, and powerful computational resources to obtain accurate solutions in a reasonable time frame. As stated in [2], the execution of such high-dimensional simulations is nearly impossible due to restricted available resources as the management of a vast amount of data is required.

In this manner, High-Performance Computing (HPC) is one strategy for overcoming these restrictions. It thrives on the power of aggregated computing power to handle data-intensive tasks that regular workstations are unable to handle. Nevertheless, access to such exclusive high-end devices is granted to only a limited set of users, especially those whose research and development work is already being adopted by large HPC facilities [3]. Furthermore, and what is more concerning, the average waiting time for the jobs (uses of such devices) becomes substantial when the simulation is too expensive to perform (see Figure 1.1).


Figure 1.1.: Average queue time versus the number of CPU cores available for the jobs on supercomputer PARAM Yuva II [3].

Therefore, the development of techniques that speed up the simulation, modeling, and rendering of finite element intensive tasks would allow researchers to acquire the findings of sophisticated analyses more quickly and efficiently. These techniques are known as compression techniques since the end result is a reduced model of the original. This pursuit is what engineers and scientists from all disciplines are attempting to uncover in what has been referred to by many authors as the fourth paradigm of scientific discovery [4].

As a result, the aim of this thesis is to harness the power of applying data-driven methodologies in finite element simulations to the availability of vast and increasing quantities of data as a way of integrating compression techniques that accelerate the study and optimization of nonlinear systems with complex multiscale physics.

### 1.5. Schedule

Designing a planification strategy is one of the primary jobs in the early stages of development. There are various tools available to aid in this pursuit. In this scenario, a work breakdown structure was initially constructed to identify the tasks to be accomplished, followed by a Gantt chart to demonstrate their sequencing in the timeline.

### 1.5.1. Description of the tasks to be developed

The tasks to be carried out have been collected and ordered in the following Table 1.1, where a brief description of each activity to be developed is provided.

| ID | Task | Description |
| :--- | :--- | :--- |
| 0 | Introduction to data compression | Beginning of the study with some basic notions of the field of <br> interest |
| 1 | Overview of data-driven methods | Get an overview of the methods available for performing data <br> compression. |
| 1.1 | Introduction to SVD | Inquire into high-dimensional reduction models using SVD. |
| 1.2 | Introduction to POD | Inquire into high-dimensional reduction models using POD. |
| 1.3 | Introduction to Fourier Transform | Inquire into the Fourier series' application to data compression. |
| 1.4 | Introduction to Wavelet Transform | Inquire into the Wavelet application to data compression. |
| 2 | Review of FE methdology | A comprehensive understanding of the finite element methodology. |
| 2.1 | Parametrization of the FE model | Defining and assigning variables that control the behavior of the <br> finite element model. |
| 2.2 | Overview of reduced order modelling | Investigation of the panorama of the reduced-order modelling. |
| 2.3 | Approaches for the reduction stages | Investigation of the applications of the reduced-order modelling. |
| 2.4 | Introduction to ECM algorithm | Inquire into the Empirical Cubature Method algorithm details. |
| 3 | Performance comparison and valida- | Performing a comparison of the reduced-order model developed. |
| tion of the model | Algorithm applicability to the code | Assessing the applicability of the algorithm to the provided code. |
| 3.1 | Evaluation of an strucutural simple case of study. |  |
| 3.2 | Numerical assessment | Performing a comparison between the FE entire model and the <br> reduced HROM. |
| 3.3 | Performance analysis | Analyze the results obtained by using the algorithms developed by <br> the code provided. |
| 3.4 | Conclusions and Results | Development of the deliverables (report, budget, project charter). |
| 4 | Project Management | Drafting and realization of the project charter delivery. |
| 4.1 | Project Charter | Drafting and realization of the report delivery. |
| 4.2 | Report | Drafting and realization of the budget delivery. |
| 4.3 | Budget |  |

Table 1.1.: Table summary of all tasks to carry out along with their description.

### 1.5.2. Interdependency relationships among tasks

The task dependencies are shown in the Table 1.2 below, which was created to help organize the schedule later shown in the Gantt chart in Figure 1.3.

| ID | Task | Preceding tasks |
| :---: | :--- | :---: |
| 0 | Introduction to data compression |  |
| 1 | Overview of data-driven methods | 0 |
| 1.1 | Introduction to SVD | 1 |
| 1.2 | Introduction to POD | 1.1 |
| 1.3 | Introduction to Fourier Transform | 1.2 |
| 1.4 | Introduction to Wavelet Transform | 1.3 |
| 2 | Review of FE methdology | 1.4 |
| 2.1 | Parametrization of the FE model | 2 |
| 2.2 | Overview of reduced order modelling | 2.1 |
| 2.3 | Approaches for the reduction stages | 2.2 |
| 2.4 | Introduction to ECM algorithm | 2.3 |
| 3 | Performance comparison and validation of the model | 2.4 |
| 3.1 | Algorithm applicability to the code | 3 |
| 3.2 | Numerical assessment | 3.1 |
| 3.3 | Performance analysis | 3.2 |
| 3.4 | Conclusions and Results | 3.3 |
| 4 | Project Management | 1 |
| 4.1 | Project Charter | 1 |
| 4.2 | Report | 1.4 |
| 4.3 | Budget | 3.4 |

Table 1.2.: Table summary of all tasks dependencies relationships.

### 1.5.3. Work breakdown structure (WBS)

The scheme of all the activities to be developed is shown in the next figure, where the structure of the work can be seen clearly.


Figure 1.2.: Work breakdown structure of the project (Source: Own).

### 1.5.4. Gantt chart

Finally, the Gantt chart shows the duration of each task, their start and end throughout the course of the thesis. It is worth mentioning that the Gantt chart is planned for a regular TFE semester, that is, between the start in February and the end in June. The design of the planification has been done in a cascade mode, as it can be seen in Table 1.2.


Figure 1.3.: Gantt chart of the project (Source: Own).

| Task | Start Date | End Date | Duration [days] |
| :---: | :---: | :---: | :---: |
| 0. Introduction to data compression | 13/Feb | 4/Mar | 19 |
| 1. Overview of data-driven methods | 5/Mar | 25/Mar | 20 |
| 1.1. Introduction to SVD | 26/Mar | 4/Apr | 9 |
| 1.2 Introduction to POD | 5/Apr | 10/Apr | 5 |
| 1.3 Introduction to Fourier Transform | 11/Apr | 15/Apr | 4 |
| 1.4 Introduction to Wavelet Transform | 16/Apr | 22/Apr | 6 |
| 2. Review of FE methodology | 23/Apr | 5/May | 12 |
| 2.1. Parametrization of the FE model | 6/May | 13/May | 7 |
| 2.2. Overview of reduced-order modelling | 14/May | 19/May | 5 |
| 2.3. Approaches for the reduction stages | 20/May | 23/May | 3 |
| 2.4. Introduction to ECM algorithm | 24/May | 31/May | 7 |
| 3. Performance comparison and validation of the model | 31/May | 17/Jun | 17 |
| 3.1. Algorithm applicability to the code | 31/May | 5/Jun | 5 |
| 3.2. Numerical assessment | 6/Jun | 10/Jun | 4 |
| 3.3. Performance analysis | 11/Jun | 13/Jun | 2 |
| 3.4. Conclusions and Results | 14/Jun | 17/Jun | 3 |
| 4. Project Management | 13/Mar | 21/Jun | 100 |
| 4.1. Project Charter | 13/Mar | 17/Mar | 4 |
| 4.2. Report | 25/Apr | 20/Jun | 56 |
| 4.3. Budget | 20/Jun | 21/Jun | 1 |

Table 1.3.: Task activity duration for Figure 1.3.

## 2. State of the art

The development of computing, as well as the associated techniques and tools that have emerged over time, has allowed us to perform calculations and simulations that were previously unimaginable regarding the complex physical reality existing in the world. However, although computing devices continue to be improved and computational power is added to them, there are physical restrictions that lead us to believe that not every model can be solved at ease, particularly those requiring dimensional complexity and high precision in small regions.

This increased demand for computational advancement has opened up many fields of research about the effective treatment of available hardware, both through parallel techniques [5] or the treatment of memory and storage availability [6]. There is, however, another approach aiming at the same goal that seeks to discover a more condensed and reduced manner of describing the same problem without resorting to such errors that the problem becomes too distorted. This current trend involves data compression methodologies.

It is important to remark that data compression, or the act of representing a set of information in a more compact form, is actually composed of two different algorithms. A properly named compression algorithm takes an input $\chi$ and generates a representation $\chi_{c}$ that contains fewer bits than the original input, and another reconstruction algorithm acts on the compressed representation $\chi_{c}$ to create a reconstruction $\gamma$. Figure 2.1 depicts these procedures schematically.


Figure 2.1.: Generalized schematic of compression and reconstruction algorithm, extracted from [7].

Data compression algorithms can be divided into two broad classes according to the reconstruction requirements: lossless compression schemes, in which $\gamma$ is identical to the input $\chi$, and lossy compression schemes, which typically provide much higher compression than lossless compression but allow $\gamma$ to differ from $\chi$.

### 2.1. Lossless and lossy compression

Lossless compression techniques are those that do not involve any loss of information, as the term suggests. If the data has been losslessly compressed, the original data can be retrieved exactly from the compressed data. Applications that cannot accept any difference between the original and the recreated data, such as text compression or medical imaging, typically adopt lossless compression.

While there are many situations where it is required that the reconstruction match the original exactly, there are other circumstances in which it is permissible to loosen this condition in order to achieve greater compression. Lossy compression strategies are used in these circumstances.

Lossy compression techniques involve some information loss in the process, with the resulting data often being irrecoverable or hard to precisely recreate. In exchange for allowing this distortion in the reconstruction, it typically gets compression ratios that are substantially higher than those that are feasible with lossless compression.

In the context of this investigation, reversible compression will be foregone in favor of one that yields larger compression ratios. Additionally, discarding the compression error is unnecessary since a numerical approximation of a physical situation is already being tackled, carrying a previous error added to the machine precision. At this stage in the analysis of a complicated physical system, millimetric accuracy is less important than getting close to reality and understanding the size of the issue.

Once an initial compression overview has been developed, the performance of the algorithms treated needs to be compared and optimized. However, due to the variety of application areas, the measure of performance needs to be established in different terms.

### 2.2. Measures of performance

There are several ways in which a compression algorithm can be evaluated. The amount of compression used, the amount of memory needed to construct the method, or how quickly the algorithm runs under specified conditions can all be used to quantify the algorithm's relative complexity.

A very sensible way to assess how well a compression algorithm compresses a given set of data is to compare the ratio of the number of bits required to represent the data before and after compression. This ratio is called the compression ratio. As an example, consider an array of $1024 \times 1024$ pixels that needs $1,048,576$ bytes to be stored. If this array were to be compressed, and the compressed version used 16,384 bytes, then the compression ratio would be $8: 1$.

The fact that the reconstruction and the original data are different in lossy compression made us look for a method for measuring the difference in order to establish how effective a compression technique is. The term used to describe the acceleration of the method, computationally speaking, is the speedup factor, which quantifies the ratio of the time required for the reference implementation to the time required for the improved implementation. In Section 5.1, all these issues are properly addressed.

### 2.3. Dimensionality reduction

Generally speaking, any endeavor aimed at creating a simpler model from a more complex one is referred to as model order reduction [8]. The simpler model is usually referred to as the reduced-order model (ROM), while the more complex one is said to be the full-order or high-fidelity model. One of the most significant challenges when dealing with a full-order model is the establishment of a constitutive link between macro-strain and macro-stresses at the macro-scale that precisely captures the characteristics and arrangement of the different phases at the finer scale (i.e., micro-scale) of the model.

In this regard, hierarchical multiscale approaches [9] have gained prominence. These approaches employ a divide-and-conquer strategy, in which the domain is partitioned into smaller subdomains (e.g., elements) at various scales. However, this division alone does not reduce the model, since the total number of unknowns remains the same. What distinguishes hierarchical multiscale models is the inclusion of simplifying assumptions [10] in the equations governing the interaction (i.e., connection) between contiguous subdomains at each scale. Under these assumptions, the constitutive link can be systematically established [11] using boundary value problems (BVPs) on specific representative subdomains for each point at the coarse scale.


Figure 2.2.: Domain and subdomains interface in the commonly known first-order homogenization [12, 13], extracted from [14]

Methods for solving the BVP range from analytical approaches, which are computationally efficient but limited to simple structures, to direct computational methods, such as the two-level Finite Element ( $\mathrm{FE}^{2}$ ) method [15], which have broader applicability but come with a high computational cost. In between these extremes, semi-analytical techniques, like Transformation Field Analysis (TFA) [16], broaden the area of research while still requiring manageable computational resources.

Nevertheless, these methods continue to rely on previously specified presumptions regarding the constitutive behavior of the relevant phases. To overcome these limitations, an emerging technique that has recently drawn more attention [14], employs the so-called [17] reduced-basis (RB) approximation in the solving of the fine-scale BVPs. Unlike Galerkin approximation procedures, where the basis functions are constructed from polynomials or transcendental functions like sines and cosines, these RB approximations are carried out from computational experiments conducted in an offline stage (commonly termed empirical basis functions [18], meaning "derived from computational experiments").

The experiments involve solving a battery of BVPs for various macro-strain histories and combining the results of these FE calculations into a data set consisting of hundreds or even thousands (depending on the number of time steps) of displacement field solutions, commonly known as snapshots. If all of these snapshots were minimally connected with one another, the dimension of the manifold spanned by them would be too large, making the entire approach unfeasible because it would no longer qualify as a truly reduced basis method. Fortunately, as demonstrated later in this paper, the majority of these snapshots do show strong linear correlations between one another (i.e., they have redundant information) and also contain deformation modes that are unrelated to the accuracy of coarse-scale predictions.

All that is needed to achieve the desired decreased basis and a much reduced dimensional representation of the solution data set is an automatic method (i.e., machine learning or data-driven method) to locate and eliminate this redundant and irrelevant information while retaining, to the greatest extent feasible, its core characteristics. The problem of reducing unneeded complexity from large data sets in order to identify dominating patterns is fundamental to fields such as digital image compression [19], and hence numerous efficient dimensionality reduction (i.e., data compression) techniques already exist.

A generic approach that has been used since the dawn of computing and served as the foundation for many other data-driven methods is the Singular Value Decomposition (SVD). Initially developed by Beltrami and Jordan in [20] and later generalized by Autonne [21], there have been numerous algorithms developed during the last decades for efficiently solving SVD, as seen in [22]. Some of these methods are reviewed in later sections. One of the central uses of the SVD, as well as one of the simplest and most popular compression algorithms, is the Proper Orthogonal Decomposition (POD). The method, which is also known as the Karhunen-Loève decomposition, was originally introduced independently by Karhunen [23] and Loève [24], among others. Although applied mainly to analyze the statistical properties of large-scale data [25, 26], has been employed recently in nonlinear PDE-based models [27], making it suitable for the reduction of large-scale systems [28].

Another approach for data compression is the translation of equations into a coordinate system where expressions decouple and are accessible to computation and analysis. This is what transform-based compression methods are based on, such as the discrete Fourier transform (DFT) and fast Fourier transform (FFT), both of which are derived from the Fourier transform on discrete vectors of data. A proper introduction to the theoretical development of the Fourier transform can be seen in [29]. A Fourier-related transform similar to the DFT is the discrete cosine transform (DCT) proposed by Nasir Ahmed in [30], a method most commonly used in signal processing and image compression.

More recently, the so-called wavelets [31] have surpassed the status of traditional transformbased compression by taking advantage of a multi-resolution decomposition, which is particularly beneficial for a variety of applications. While the usage of wavelets is now most widely used in image compression [32, 33], some techniques [34] arise in the reduction regard, which is the main reason why they need to be covered and studied in more detail.

All in all, the choice of the proper data compression technique to apply depends greatly on the characteristics of the problem, such as the size, complexity, and sparsity of the solution data. In the succeeding section, a further discussion about the intrinsic nature of each technique will be extensively discussed.

## 3. Compression algorithms

The next section of this study will cover a number of important themes. The display of dominant low-dimensional patterns in the data of complex systems will be the first block addressed (i.e., singular values, characteristic modes, and projections to low-rank subspaces) through the usage of powerful data-driven methods (e.g., singular value decomposition and proper orthogonal decomposition) that can effectively sense patterns in the data and form compact representations for modeling and control.

The second block addressed will be centered around the extraction of patterns in the data, where the aim is to find coordinate transforms (e.g., spectral decomposition, the Fourier transform, generalized functions) that reduce the complexity of the system. Despite the fact that these techniques have often only been used for linear dynamics and straightforward idealized geometries, the ability to construct data-driven transformations creates opportunities to apply these strategies to new research challenges with more intricate geometries and boundary conditions.

### 3.1. Singular Value Decomposition

The first technique reviewed in this report is the singular value decomposition (SVD), which will serve as the basis for many other approaches developed subsequently. In brief, SVD will be used to find low-rank approximations to matrices. It is based on the fact that data from complex systems is often low-rank, which means that the high-dimensional data obtained in analyses can be explained by a small number of dominating patterns [4].

An effective and mathematically reliable technique for identifying these patterns in data is the SVD. It takes a high-dimensional set of data points and reduces them to a lower dimensional space that reveals the original data substructure and orders it from the largest variation to the least [35]. SVD is useful for data compression techniques because fluctuations below a certain threshold may be easily disregarded, allowing for significant data reduction with the knowledge that the key relationships of interest have been maintained.

Because the decomposition is modified in the compression algorithm to produce a lowrank approximation matrix, the reconstructed matrix differs slightly from the original matrix. Henceforth, the algorithm performed results in some loss of information (i.e., lossy compression). However, SVD is proved to be numerically stable [4] and, what is more relevant, is guaranteed to exist for any matrix, unlike other decomposition like the eigendecomposition. In the next subsection, a thorough mathematical explanation is discussed.

### 3.1.1. General definition

Singular value decomposition is a unique matrix decomposition based on a theorem from linear algebra [22] that exists for every complex-valued matrix $\mathbf{A} \in \mathbb{C}^{n \times m}$ :

$$
\begin{equation*}
\mathbf{A}=\mathbf{U} \mathbf{\Sigma} \mathbf{V}^{*} \tag{3.1}
\end{equation*}
$$

where $\mathbf{U} \in \mathbb{C}^{n \times n}$ is an orthogonal matrix, $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix, and $\mathbf{V} \in \mathbb{C}^{m \times m}$ is the transpose of an orthogonal matrix. Both $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices ${ }^{1}$ and the * denotes the complex conjugate transpose ${ }^{2}$.

The columns of $\mathbf{U}$ are orthonormal eigenvectors of $\mathbf{A} \mathbf{A}^{T}$, called left singular values, while the columns of $\mathbf{V}$ are orthonormal eigenvectors of $\mathbf{A}^{T} \mathbf{A}$, called right singular values. The diagonal elements of $\boldsymbol{\Sigma}$ are called singular values and are ordered from largest to smallest, which are at the same time the non-negative and non-zero square roots of the eigenvalues of $\mathbf{A} \mathbf{A}^{T}$ and $\mathbf{A}^{T} \mathbf{A}$.

The approach works by ranking the dimensions according to variation and determining which dimension has the greatest variance. Once this dimension has been identified, it is possible to use fewer dimensions to find the closest match to the original data points. That is the main reason why SVD may be thought of as a data compression or reduction technique.

From the definition of SVD seen in Equation (3.1) and knowing $\boldsymbol{\Sigma}$ is diagonal, the rank- $r$ of the SVD approximation is given by the sum of $r$ distinct rank-1 matrices [4]:

$$
\mathbf{A}=\sum_{i=1}^{r} \sigma_{i} \mathbf{u}_{i} \mathbf{v}_{i}^{*}=\sigma_{1} \mathbf{u}_{1} \mathbf{v}_{1}^{*}+\sigma_{2} \mathbf{u}_{2} \mathbf{v}_{2}^{*}+\cdots+\sigma_{r} \mathbf{u}_{r} \mathbf{v}_{r}^{*}
$$

where $\sigma_{i}$ is the $i$-th singular value of matrix $\mathbf{A}, \mathbf{u}_{i}$ and $\mathbf{v}_{i}$ are corresponding singular values of matrix $\mathbf{A}$, and $r=\min (n, m)$. Assuming the fact that singular values are ordered as follows: $\sigma_{1} \geq \sigma_{2} \geq \sigma_{3} \geq \ldots \geq \sigma_{r}$, the aforementioned formula implies that the first term in the sum would contribute the most to the matrix $\mathbf{A}$, while the last term would have the least. Therefore, if only the first $\mathbf{k}$ members of the above summation are taken, the approximation matrix results in:

$$
\mathbf{A} \approx \mathbf{A}^{\prime}=\sum_{i=1}^{k} \sigma_{i} \mathbf{u}_{i} \mathbf{v}_{i}^{*}
$$



Figure 3.1.: Decomposition of input matrix $\mathbf{A}$ into diagonal matrix of singular values $\Sigma$ and matrices of left and right singular vectors.
Orange color illustrates low-rank approximation (Source: Own).

[^0]The quantity of the singular values left out of the approximation formula affects the accuracy of the result, namely $\sigma_{k+1} \ldots \sigma_{r}$. The first singular value is assumed to be orders of magnitude greater than the singular values at the end of the decomposition sequence in the compression process. When $r=k$ or $\sigma_{i}=0$ for all $i>r$, special circumstances apply, and the compression results in lossless compression because the omitted singular values do not add to the sum. To prevent losing crucial data in such situations, approximation errors must be assessed and taken into consideration. A simple way to understand the truncation of singular values is depicted in the above Figure 3.1, where the orange color is there to represent the low-rank approximation of the input.

### 3.1.2. Intuitive interpretation

There is actually an intuitive and elegant visual interpretation beneath the SVD method. It is necessary to bear in mind that any rectangular matrix of $m$ by $n$ has the power to transform a vector in the $m$-th dimension to a vector in the $n$-th dimension, something often called the linear transformation from $\mathbb{R}^{m}$ to $\mathbb{R}^{n}$, exemplified in Figure 3.2. This is more easily understood when thinking about matrix vector multiplication. In particular, a 2 by 3 matrix has the ability to take a vector in $\mathbb{R}^{3}$ down to a vector in $\mathbb{R}^{2}$.


Figure 3.2.: Linear transformation of a vector in $\mathbb{R}^{m}$ to $\mathbb{R}^{n}$ via a rectangular matrix $\mathbf{A}$ (Source: Own).

Therefore, if a rectangular matrix $m$ by $n$ was composed of a square identity matrix $m$ by $m$ and the leftovers until $n$ columns were zero values, the linear transformation applied to the $\mathbb{R}^{m}$ vectors would erase $|n-m|$ dimensions. For instance, if an $\mathbb{R}^{3}$ vector was multiplied by a rectangular matrix 3 by 2 , the vector would translate into the 2 D plane, which means it would lose 1 dimension.

Consider now symmetric matrices, which are square matrices in which on both sides of the diagonal line the entries are identical. They hold a very strong property, which is that the eigenvectors of symmetric matrices are perpendicular to each other; that is to say, if the eigenvectors were normalized and packaged into another matrix, the result would be an orthogonal matrix.

However, most matrices in nature are not symmetrical. But there is a trick to artificially construct symmetry out of nowhere. Considering a rectangular matrix $\mathbf{A} m$ by $n$ and multiplying this matrix by its transpose $\mathbf{A} \mathbf{A}^{T}$, the result is a symmetric square matrix of $m$ by $m$. Were the transpose on the left side $\mathbf{A}^{T} \mathbf{A}$, then the result would be a symmetric square matrix of $n$ by $n$.

The matrices obtained on the previous page are not completely unknown, since they are the same matrices that had already been announced at the beginning of the section, matrices $\mathbf{U}$ and $\mathbf{V}$, which contain the perpendicular (or orthonormal when normalize) eigenvectors of the multiplication of matrix $\mathbf{A}$ from the right and left sides, respectively. As stated in the beginning of the section, both columns of these matrices were given special names, which were left singular values for the columns of $\mathbf{U}$ and right singular values for the columns of V, both depicted in Figure 3.3.

$$
\mathbf{A A}^{T}=\mathbf{U}=\left[\begin{array}{cc}
\mid & \mid \\
\overrightarrow{u_{1}} & \overrightarrow{u_{2}} \\
\mid & \mid
\end{array}\right] \quad \mathbf{A}^{T} \mathbf{A}=\mathbf{V}=\left[\begin{array}{cc}
\mid & \mid \\
\overrightarrow{v_{1}} & \overrightarrow{v_{2}} \\
\mid \overrightarrow{v_{3}} \\
\mid & \mid
\end{array}\right]
$$

Figure 3.3.: Left and right singular values of a rectangular matrix $\mathbf{A}$,

$$
m=3 \text { and } n=2(\text { Source: Own })
$$

It is provable that matrices $\mathbf{U}$ and $\mathbf{V}$ are positive semi-definite (PSD) matrices. This implies that the eigenvalue for each eigenvector is non-negative $\lambda_{i} \geq 0$. Moreover, if the eigenvalues from both matrices were to be sorted in descending order, the overlapping ones would be numerically identical, which means that the biggest eigenvalue of $\mathbf{U}$ equals the biggest eigenvalue of $\mathbf{V}$, and so forth.

Just like the singular vectors, those shared eigenvalues are indirectly derived from the very original matrix A. Computing the square root of each of these eigenvalues, the singular values of matrix $\mathbf{A}$ would be obtained: $\sqrt{\lambda_{i}}=\sigma_{i}$. Proof of the statements made in this section can be reviewed in detail in chapter 3 of Hopcroft's book in [36].

Once again, the statement that any matrix $\mathbf{A}$ can be unconditionally decomposed into three matrices - in which matrix $\boldsymbol{\Sigma}$ is rectangularly diagonal containing the singular values of matrix $\mathbf{A}$ arranged in descending order; the matrix $\mathbf{V}$ and $\mathbf{U}$ are orthogonal matrices that contain the normalized eigenvectors arranged in descending order of their eigenvalue of the multiplication of matrix $\mathbf{A}$ by its transpose both in the right and left sides, respectively. This is clearly visualized in Figure 3.4, where the content of each matrix is dissected.

$$
\left[\begin{array}{l}
\mathrm{A}
\end{array}\right]=\left[\begin{array}{cc}
\mid & \mid \\
\overrightarrow{u_{1}} & \overrightarrow{u_{2}} \\
\mid & \mid
\end{array}\right]\left[\begin{array}{ll}
\sigma_{1} & \\
& \sigma_{2} \\
&
\end{array}\right]\left[\begin{array}{l}
-\vec{v}_{1}^{T} \\
-\vec{v}_{2}^{T}- \\
-\vec{v}_{3}^{T}
\end{array}\right]
$$

Figure 3.4.: Singular Value Decomposition formula dissection of a rectangular matrix $\mathbf{A}, m=3$ and $n=2$ (Source: Own).

The matrix $\mathbf{A}$ itself applies a complicated linear transformation from $\mathbb{R}^{m}$ to $\mathbb{R}^{n}$. Nevertheless, it can be perfectly understood as sequentially applying the three matrices of the decomposition made on Figure 3.4. The $\mathbf{V}^{T}$ matrix is an orthogonal matrix that applies a rotation such that the right singular vectors return to the standard basis. More precisely, the singular vector with the biggest singular value lands on the x -axis, while the singular vector with the second biggest value lands on the y-axis, and so forth until the $n$-th axis.

The $\Sigma$ matrix is a rectangular diagonal matrix that is essentially a square diagonal matrix composed of a dimension eraser, which, as stated previously, is a rectangular matrix that removes $|n-m|$ dimensions. If matrix $\mathbf{A}$ was to be 2 by 3 , then the dimension erased would be the third one. Afterwards, the diagonal matrix stretches each axis based on the singular value $\sigma_{i}$. As the final step, matrix $\mathbf{U}$ rotates the standard basis to align with the left singular vectors. Every step explained can be observed in summary in the geometric interpretation elaborated by Glibert Strang in The Fundamental Theorem of Linear Algebra in Figure 3.5.


Figure 3.5.: Geometric interpretation of the singular value decomposition of a 2 by 2 matrix $\mathbf{A}$, extracted from [37].

### 3.1.3. Method computation

Generally, algorithms for computing singular values are variants of algorithms for computing the eigenvalue decomposition of hermitian square matrices ${ }^{3}$. The goal is to calculate the square roots of the eigenvalues of $\mathbf{A}^{T} \mathbf{A}$ without actually computing $\mathbf{A}^{T} \mathbf{A}$.

Beforehand, in order to prevent needless complexity, all the sections and ensuing techniques have been given as though the matrix were real. Nevertheless, an extension for complex matrices can be applied to the SVD method. The book Numerical Linear Algebra by Lloyd N. Trefethen and David Bau in [38] contains comprehensive documentation on the implementation of complex matrices in the computation of SVD.

As stated before from Equation (3.1), the non-zero singular values of $\mathbf{A} \in \mathbb{R}^{m \times n}(m \geq n)$ are the square roots of the non-zero eigenvalues of $\mathbf{A}^{T} \mathbf{A}$ or $\mathbf{A A}^{T}$.

$$
\mathbf{A}^{T} \mathbf{A}=\left(\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{T}\right)^{T}\left(\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{T}\right)=\mathbf{V} \boldsymbol{\Sigma}^{T} \mathbf{U}^{T} \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{T}=\mathbf{V}\left(\boldsymbol{\Sigma}^{T} \boldsymbol{\Sigma}\right) \mathbf{V}^{T}
$$

Therefore, mathematically speaking, the SVD of A can be calculated analytically as follows:

1. From $\mathbf{A}^{T} \mathbf{A}$;
2. Compute the eigenvalue decomposition of $\mathbf{A}^{T} \mathbf{A}=\mathbf{V} \Lambda \mathbf{V}^{T}$;
3. Let $\boldsymbol{\Sigma}$ be the $m \times n$ non-negative diagonal square root of $\Lambda$;
4. Solve the system $\mathbf{U} \boldsymbol{\Sigma}=\mathbf{A} V$ for unitary $\mathbf{U}$.
[^1]This algorithm is regularly utilized, frequently by individuals who have independently rediscovered the SVD. The matrix $\mathbf{A}^{T} \mathbf{A}$ is known as the covariance matrix of $\mathbf{A}$. The approach is unreliable, however, because it transforms the SVD problem into an eigenvalue problem, which might be significantly more vulnerable to disturbances. The difficulty can be understood from the proof given in [38].

The SVD can be reduced to an eigenvalue problem in a different, stable manner. Assume that $m=n$ and that $\mathbf{A}$ is square. This is not a necessarily arduous limitation because it can be demonstrated that issues with rectangular singular values can be converted to square problems [38]. Consider the symmetric or hermitian $2 m \times 2 m$ matrix:

$$
\mathbf{H}=\left[\begin{array}{cc}
0 & A^{T} \\
A & 0
\end{array}\right] .
$$

Since $\mathbf{A}=\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{T}$ implies $\mathbf{A V}=\mathbf{U} \boldsymbol{\Sigma}$ and $\mathbf{A}^{T} \mathbf{U}=\mathbf{V} \boldsymbol{\Sigma}^{T}=\mathbf{V} \boldsymbol{\Sigma}$, the next is deduced

$$
\left[\begin{array}{cc}
0 & \mathbf{A}^{T} \\
\mathbf{A} & 0
\end{array}\right]\left[\begin{array}{cc}
\mathbf{V} & \mathbf{V} \\
\mathbf{U} & -\mathbf{U}
\end{array}\right]=\left[\begin{array}{cc}
V & V \\
\mathbf{U} & -\mathbf{U}
\end{array}\right]\left[\begin{array}{cc}
\boldsymbol{\Sigma} & 0 \\
0 & -\boldsymbol{\Sigma}
\end{array}\right]
$$

which amounts to an eigenvalue decomposition of $\mathbf{H}$. The singular vectors of $\mathbf{A}$ may be obtained from the eigenvectors of $\mathbf{H}$, and it can be seen that the singular values of $\mathbf{A}$ are the absolute values of the eigenvalues of $\mathbf{H}$.

As a result, one may create the matrix $\mathbf{H}$ and determine its eigenvalue decomposition to produce the SVD of a square matrix $\mathbf{A}$. This method is stable in comparison to using $\mathbf{A}^{T} \mathbf{A}$ or $\mathbf{A}^{T} \mathbf{A}$. The typical SVD methods are based on this concept, albeit in a veiled manner in which no matrices of dimension $m+n$ are explicitly constructed. Additionally, a preliminary unitary reduction to bidiagonal form is a crucial step in facilitating a quick process.

Hermitian eigenvalue problems are often resolved in two-phase computation [38]: first, the matrix is reduced to tridiagonal form, and then the tridiagonal matrix is diagonalized. Since the 1960s, when Golub, Kahan, and other researchers published their work, a similar two-phase method has been accepted for SVD computing. After converting the matrix $\mathbf{A}$ into bidiagonal form, the matrix is diagonalized.
where $\times$ represents a generally non-zero entrance. In Phase 1 of the SVD computation, A is converted into a bidiagonal form by performing separate unitary operations on the left and right. The simplest way is the Golub-Kahan bidiagonalization, which uses Householder reflectors as its core.

Reviewing QR factorization and Householder triangularization is key to fully understanding the following explanation; therefore, Appendix B exposes important remarks that the reader should take note of before moving forward.

Householder reflectors are applied alternately on the left and the right. The right reflection adds a row of zeros to the right of the first superdiagonal, whereas the left reflection introduces a column of zeros below the diagonal, preserving the zeros that were just added to the column. As an illustration, consider a $5 \times 4$ matrix, i.e.,

$$
\begin{aligned}
& \underbrace{\left[\begin{array}{cccc}
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times
\end{array}\right]}_{A} \xrightarrow{U_{1}^{T}} \underbrace{\left[\begin{array}{cccc}
\times & \times & \times & \times \\
0 & \times & \times & \times \\
0 & \times & \times & \times \\
0 & \times & \times & \times \\
0 & \times & \times & \times
\end{array}\right]}_{U_{1}^{T} A} \xrightarrow{V_{1}} \underbrace{\left[\begin{array}{cccc}
\times & \times & 0 & 0 \\
0 & \times & \times & \times \\
0 & \times & \times & \times \\
0 & \times & \times & \times \\
0 & \times & \times & \times
\end{array}\right]}_{U_{1}^{T} A V_{1}} \xrightarrow{U_{2}^{T}} \underbrace{\left[\begin{array}{cccc}
\times & \times & 0 & 0 \\
0 & \times & \times & \times \\
0 & 0 & \times & \times \\
0 & 0 & \times & \times \\
0 & 0 & \times & \times
\end{array}\right]}_{U_{2}^{T} U_{1}^{T} A V_{1}} \\
& \xrightarrow{V_{2}} \underbrace{\left[\begin{array}{cccc}
\times & \times & 0 & 0 \\
0 & \times & \times & 0 \\
0 & 0 & \times & \times \\
0 & 0 & \times & \times \\
0 & 0 & \times & \times
\end{array}\right]}_{U_{2}^{T} U_{1}^{T} A V_{1} V_{2}} \xrightarrow{U_{3}^{T}} \underbrace{\left[\begin{array}{cccc}
\times & \times & 0 & 0 \\
0 & \times & \times & 0 \\
0 & 0 & \times & \times \\
0 & 0 & 0 & \times \\
0 & 0 & 0 & \times
\end{array}\right]}_{U_{3}^{T} U_{2}^{T}} \stackrel{U_{1}^{T} A V_{1} V_{2}}{U_{4}^{T}} \underbrace{\left[\begin{array}{cccc}
\times & \times & 0 & 0 \\
0 & \times & \times & 0 \\
0 & 0 & \times & \times \\
0 & 0 & 0 & \times \\
0 & 0 & 0 & 0
\end{array}\right]}_{U_{4}^{T} U_{3}^{T} U_{2}^{T} U_{1}^{T} A V_{1} V_{2}}=\mathbf{B}
\end{aligned}
$$

Note that the related identity matrices are excluded since, after the second step, no more right-multiplications were required. The last matrix, $\mathbf{B}$, has a bidiagonal shape. The resulting algorithm for $m \times n$ matrices with $(m \geq n)$ dates back to 1965 and is given by [39].

```
Algorithm 1 Golub-Kahan Bidiagonalization
    for \(k=1\) to \(n\) do
        \(x=A_{k: m, k}\)
        \(u_{k}=x+\operatorname{sign}(x(1))\|x\| e_{1}\)
        \(u_{k}=u_{k} /\left\|u_{k}\right\|\)
        \(A_{k: m, k: n}=A_{k: m, k: n}-2 u_{k}\left(u_{k}^{T} A_{k: m, k: n}\right)\)
        if \(k \leq(n-2)\) then
            \(x=A_{k, k+1: n}\)
            \(v_{k}=x+\operatorname{sign}(x(1))\|x\| e_{1}\)
            \(v_{k}=v_{k} /\left\|v_{k}\right\|\)
            \(A_{k: m,(k+1): n}=A_{k: m,(k+1): n}-2\left(\left(A_{k: m,(k+1): n}\right) v_{k}\right) v_{k}^{T}\)
        end if
    end for
```

An improvement on Algorithm 1 is given by the Lawson-Hanson-Chan algorithm. The main idea for this latter algorithm is to first compute the QR factorization of $\mathbf{A}$, i.e., $\mathbf{A}=Q R$ and then apply the Golub-Kahan algorithm to $R$, i.e., $R=\mathbf{U B V}^{T}$. Together, this results in $\mathbf{A}=Q \mathbf{U B V}^{T}$. The advantage of this approach is that the bidiagonalization algorithm has to be applied only to a small $n \times n$ matrix, namely the non-zero part of $R$. Book Numerical Linear Algebra in [38] explores this alternative algorithm.

Finally, the computation of Phase 2 performs the SVD of the bidiagonal matrix B previously calculated. From the 1960s to the 1990s, the standard algorithm for this was a variant of the QR algorithm. More recently, divide-and-conquer algorithms have also become very competitive, as depicted in references such as [40].

### 3.1.4. Eigenfaces example

One of the most remarkable demonstrations of SVD that gives a principled approach to dimensionality reduction of high-dimensional data sets is the so-called eigenfaces example. In this case, this particular example is going to be extracted from the book Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control by Steven L. Brunton and J. Nathan Kutz. All results present in the foreseen chapter have been obtained by running the codes provided by the author in MATLAB ${ }^{4}$.

In this problem, SVD is applied to a large library of facial images to extract the most dominant correlations between images. The result of this decomposition is a set of eigenfaces that define a new coordinate system. This problem was first studied by Sirovich and Kirby in 1987 [41] and expanded on in [42]. It is widely applied to automated facial recognition, as in this particular application presented by Turk and Pentland in 1991 [43].

The demonstration of this eigenface example is going to be conducted using the Extended Yale Face Database B, extracted both from the paper [44] and [4], consisting of 38 people ( 28 from the extended database and 10 from the original database) in cropped and aligned photos under 64 lightning situations, captured with a geodesic dome with a camera facing each person involved.

(a) A single image for each person in the Yale database. Generated using

(b) All images for a specific person under different light conditions.

Figure 3.6.: Images obtained from the Extended Yale Face Database B in grey colormap (Source: Own).

Each picture has a $192 \times 168$ pixel size. Additionally, every facial picture in the library has been transformed into a large column vector of $192 \times 168=32,256$ elements. The first 36 individuals (seen in the left panel from Figure 3.6) from the total of 38 persons in the database will be utilized as our training data for the eigenfaces example, with two individuals being saved as a test set for future usage. In the right panel, a sample of all 64 photographs of a certain individual are displayed.

[^2]As mentioned earlier, each image is reshape into a large column vector, which corresponds to every single column of a data matrix $\mathbf{X}$ of size 32,256 elements. What is neat about the SVD is that the columns of $\mathbf{U}$ have exactly the same size as the columns of $\mathbf{X}$. Therefore, if reshaping $u_{1}$, which is also a column vector of 32,256 elements, the result would be the shape of a face. These particular faces are what are called the eigenfaces in the example.

As per the computation, the average face of all the columns of $\mathbf{X}$ is going to be determined and subtracted from each column vector. Afterwards, the economy-sized ${ }^{5}$ SVD is going to be computed, mainly because the 32,256 elements are not needed, but instead only the first nonzero singular values. The mean-subtracted image vectors are then stacked horizontally as columns in the data matrix $\mathbf{X}$, as shown in the schematic on Figure 3.7. The columns of $\mathbf{U}$ are the eigenfaces, and they may be reshaped back into $192 \times 168$ images to be visualized.


Figure 3.7.: Schematic procedure to obtain eigenfaces from library of faces, extracted from [4].

In Figure 3.8, the first 64 eigenfaces are shown. The first 64 columns of the matrix $\mathbf{U}$ were extracted, with the observation that each of those columns had been reshaped as an image (i.e., an array that appeared to be shaped as a face). Then, as seen in the aforementioned figure, the faces were arranged over a sizable grid of 64 total images of faces of $8 \times 8$.

[^3]Despite the fact that only the first 64 eigenfaces were plotted in Figure 3.8, there were really more calculated eigenfaces, adding up to the complete number of original data columns. As it may be observed from the figure, the first couple of eigenfaces are very ghosty and blurry, corresponding to those features that every face has in common (i.e., eyes, noses, and mouths), while in the proceeding eigenfaces, more information is being captured, such as shadows or extremely bright and dark eyebrows, as well as lightning and facial details.

The main advantage of this decomposition approach is that each person's face can still be accurately approximated using a linear combination of a few of the eigenfaces, even while only maintaining a small subset of the eigenfaces (i.e., the first few hundred).


Figure 3.8.: Images obtained from the Extended Yale Face Database B in grey colormap (Source: Own).

Figure 3.9 reveals the singular values of the data matrix $\mathbf{X}$. The number of modes is displayed on the horizontal axis, showing up to 24 hundred of them, which is naturally the number of columns in matrix $\mathbf{X}$. The magnitude of the $\log$ of the singular values in the diagonal $\boldsymbol{\Sigma}$ is displayed on the vertical axis. The ideal situation for a singular value distribution may be found if the fundamental structure of the singular value distribution curve is thoroughly investigated. This is due to the fact that the energy (i.e., the variance in the faces) is carried by the first few values before it tapers off and becomes increasingly less informative.


Figure 3.9.: Singular values of the data matrix $\mathbf{X}$ (Source: Own)

The next step is to use the eigenfaces obtained in Figure 3.8 to approximate a human face that was outside of the training set used (i.e., the 36 people used in the beginning of the example). For this, the pictures of the 38th person on the list are used for the assessment of how accurately this eigenface-space depicts the 38th person. The first $r$ columns of matrix $\mathbf{U}$ will serve as the so-called eigenface-space into which we will project the large skinny column vector $x_{i}$.

Numerically speaking, the transpose of the basis of the eigenfaces-space of $r$-rank is going to be multiplied by the $x_{i}$ column vector. The result would be the coefficients (i.e., the fingerprint of that person) in the coordinate system, so it results in the mixture of the eigenfaces needed to be added up to get that person's face.

Thus, to recreate an approximation of $x_{i}$ in that space, the previous multiplication by the linear combination of eigenfaces is repeated again, such that

$$
\tilde{x}_{i}=U_{r} \underbrace{U_{r}^{T} x_{i}}_{\alpha},
$$

where $\alpha$ represents the small dimensional vector of exactly the mixture of the first $r$ eigenfaces that $x_{i}$ contains, and $\tilde{x}_{i}$ is the projection of $x_{i}$ onto the first $r$ eigenfaces.

Figure 3.10 illustrates how this person's face gets reconstructed if the first $r$ eigenfaces are kept. The accuracy of the reconstruction of one person's face can be proved using a different number of ranks. In fact, it is seen that the face does rapidly converge to the real face. Once more, as was previously stated in Figure 3.8, while the first images preserve the characteristics of a generic face, as $r$ increases, the details of the face gradually become more distinct. Overall, the approximation is relatively poor for $r \leq 200$, but it improves for $r>400$ to a passable representation of the test image.


Figure 3.10.: Approximate representation of test image using eigenfaces basis of various order $r$. (Source: Own)

### 3.2. Proper Orthogonal Decomposition

Proper orthogonal decomposition (POD) is one of the central uses of the SVD, providing a data-driven, hierarchical coordinate system to represent high-dimensional correlated data [4]. It can be seen as the SVD algorithm applied to partial differential equations (PDEs). As such, it is one of the most important dimensionality reduction techniques available to study complex spatio-temporal systems, which are typically exemplified by nonlinear PDEs.

The central idea of POD is to reduce the dimensionality of a data set consisting of a large number of interrelated variables while retaining as much as possible of the variation (i.e., statistical information) present in the data set [45]. This is accomplished by projecting PDE dynamics to low-rank subspaces, which make it easier to evaluate simulations of the controlling PDE model. Significant speedups in computing are made possible by the low-rank models produced by the ROM, which may enable real-time control of PDE-based systems and/or optimization over parametrized PDE systems.

The optimality of POD is what makes this approach so well-liked in applications where low-dimensional, high-accuracy approximations are required [46]. POD offers a foundation for the modal decomposition of a collection of data, such as experimental or numerical simulation results. The properties of POD imply that it is a preferred basis in the sense that it is designed to optimize approximation accuracy. In order to show how this goal is achieved, the fundamental tenet of approximation processes is first explored.

### 3.2.1. Approximation basis functions

Consider a function $\mathbf{f}(x, t)$ defined over a domain of interest denoted by $\boldsymbol{\Omega}$. The aim is to approximate this function by expressing it as a linear combination of basis functions $\psi_{\mathbf{i}}(\mathbf{x})$, i.e.,

$$
\begin{equation*}
\mathbf{f}(x, t) \approx \sum_{i=1}^{m} a_{i}(t) \psi_{i}(x), \tag{3.2}
\end{equation*}
$$

where $a_{i}(t)$ are time-dependent coefficients representing the unknown amplitudes of the expansion. As the number of terms $m$ increases, the approximation is expected to approach the exact function representation.

To determine the values of the coefficients $a_{i}(t)$, a minimization process is employed. In the case of the least squares approximation, the goal is to minimize the $L^{2}$-norm of the error, defined as:

$$
\begin{equation*}
\left\|\mathbf{f}(x, t)-\sum_{i=1}^{m} a_{i}(t) \psi_{i}(x)\right\|_{L^{2}} \rightarrow \min , \tag{3.3}
\end{equation*}
$$

where $\|\cdot\|_{L^{2}}$ denotes $L^{2}$-norm defined by

$$
\|\mathbf{f}(x, t)\|_{L^{2}}=\int_{\Omega}|\mathbf{f}(x, t)|^{2} d \Omega .
$$

It is important to note that the representation in Equation (3.2) is not unique for a given function $\mathbf{f}(x, t)$. The choice of basis functions $\psi_{i}(x)$ is arbitrary, and each selection corresponds to a different set of amplitudes.

In the standard approximation approach, the responsibility of selecting the basis functions lies with the user. Based on experience, one can determine whether polynomial functions, trigonometric functions, exponential functions, or any other type of function should form the basis [28]. While increasing the number of terms $m$ typically leads to improved accuracy in the approximation given by Equation (3.2), there is no proof that the chosen basis is the optimal one for the given functions [46]. Therefore, it is natural to seek a basis that provides the best possible approximation of the function $\mathbf{f}(x, t)$ for a given number $m$. Proper orthogonal decomposition (POD) specifically addresses the selection of basis functions $\psi_{i}(x)$ and offers a method for constructing an optimal basis for the function under consideration.

### 3.2.2. General definition

There is considerable freedom in the selection of basis functions $\psi_{i}(x)$ for approximating function $\mathbf{f}(x, t)$, as long as the chosen basis is complete and linearly independent. By choosing an orthonormal set of basis functions, satisfying the property [4]

$$
\int_{\Omega} \psi_{k}(x) \cdot \psi_{j}(x) d x=\left\{\begin{array}{ll}
1, & j=k \\
0, & j \neq k
\end{array},\right.
$$

gives some advantages, as the determination of the amplitudes $a_{i}(t)$ can be simplified. In this case, the amplitudes become relatively simple, as they are given by

$$
a_{i}(t)=\int_{\Omega} \mathbf{f}(x, t) \cdot \psi_{i}(x) d \Omega .
$$

Notably, each amplitude depends solely on the corresponding basis function, $\psi_{i}(x)$. If the basis functions are non-orthogonal, determining the amplitudes would involve solving a system of linear equations.

Furthermore, this basis needs to be optimal in the sense that for each value of $m$, the approximation should be as good as possible in terms of the least square error in Equation (3.3). In other words, the aim is to find a sequence of ordered orthonormal functions such that the first two functions provide the best two-term approximation, the first three functions offer the best three-term approximation, and so on. Once identified, these specially ordered orthogonal functions are referred to as the optimal modes for the function $\mathbf{f}(x, t)$, and the representation in Equation (3.2) is known as the proper orthogonal decomposition of $\mathbf{f}(x, t)$.

To proceed with the construction of the optimal POD modes, the dynamics of the function $\mathbf{f}(x, t)$ are sampled at prescribed time intervals. Specifically, a snapshot $\mathbf{u}_{\mathbf{k}}$ contains samples of the system at different spatial locations, denoted as $\mathbf{u}\left(x_{i}, t_{k}\right)$. The continuous functions and modes are then evaluated at discrete spatial locations.

In an analysis, a large data set is typically dealt with, denoted as $\mathbf{X}$, which consists of a number of distinct instances in time:

$$
\mathbf{X}=\left[\begin{array}{cccc}
\mid & \mid & & \mid \\
\mathbf{u}_{1} & \mathbf{u}_{2} & \cdots & \mathbf{u}_{m} \\
\mid & \mid & & \mid
\end{array}\right]
$$

where the columns $\mathbf{u}_{\mathbf{k}}=\mathbf{u}\left(t_{k}\right) \in \mathbb{C}^{n}$ may be measurements from simulations or experiments.

As mentioned earlier, $\mathbf{X}$ usually consists of a time series of data with $m$ distinct measurement instances in time. Often the state-dimension $n$ is significantly larger than $m$ (i.e., on the order of millions or billions in the case of fluid systems), resulting in a tall-skinny matrix as opposed to a short-fat matrix when $n \ll m$.

To uncover the optimal basis set, the singular value decomposition (SVD) is employed for the complex-valued matrix $\mathbf{X} \in \mathbb{C}^{n \times m}$ :

$$
\mathbf{X}=\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^{*}
$$

where $\mathbf{U} \in \mathbb{C}^{n \times n}$ and $\mathbf{V} \in \mathbb{C}^{m \times m}$ are unitary matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ is a matrix with nonnegative entries on the diagonal, with the values ordered from largest to smallest. SVD gives essential insight into developing an appropriate basis set adapted to the individual challenge. In particular, matrix $\mathbf{U}$ is guaranteed to provide the best modes to approximate $\mathbf{X}$ in the $L^{2}$ sense [46], with its columns representing the orthogonal modes necessary for the ideal basis. The matrix $\mathbf{V}$ captures the time history of each modal element, while $\boldsymbol{\Sigma}$ indicates the relative importance of each mode. Remember that the modes are arranged with the most dominant first and the least dominant last.

Typically, the total number of modes $(r)$ is determined by the number of snapshots $m$ taken in constructing $\mathbf{X}$ (where normally $n \gg m$ ). The aim is to determine the minimal number of modes necessary to accurately represent the dynamics of the function, thus selecting a rank- $r$ approximation that accurately captures the true dynamics, where typically $r \ll m$. The quantity of interest is then the low-rank decomposition of the SVD given by

$$
\tilde{\mathbf{X}}=\tilde{\mathbf{U}} \tilde{\Sigma} \tilde{\mathbf{V}}^{*},
$$

where $\|\mathbf{X}-\tilde{\mathbf{X}}\|<\varepsilon$ for a given small value of epsilon. From the columns of the truncated matrix $\tilde{\mathbf{U}}$, the desired basis modes $\psi_{k}$ are extracted, forming the optimal basis matrix $\boldsymbol{\Psi}$ :

$$
\tilde{\mathbf{U}}=\boldsymbol{\Psi}=\left[\begin{array}{cccc}
\mid & & & \mid \\
\psi_{1} & \psi_{2} & \cdots & \psi_{r} \\
\mid & \mid & & \mid
\end{array}\right],
$$

where the truncation preserves the $r$ most dominant modes used in Equation (3.2). The truncated $r$ modes $\left\{\boldsymbol{\psi}_{1}, \boldsymbol{\psi}_{2}, \cdots, \boldsymbol{\psi}_{r}\right\}$ are orthogonal and serve as a low-rank orthogonal basis to represent the dynamics of the function.

The aforementioned processes may be summarized in Algorithm 2, where the basis was computed using SVD. Be aware that it may alternatively have been accomplished using an algorithm to solve the issue of eigenvalue decomposition (e.g., as in the case [47]); in such case, numerous writers refer to the technique by other names, such as PCA or KLD [45].

```
Algorithm 2 POD basis of rank \(r\)
    Set \(\mathbf{X}=\left[x_{1}, x_{2}, \cdots, x_{n}\right] \in \mathbb{R}^{m \times n}\),
    Compute singular value decomposition \([\boldsymbol{\Psi}, \boldsymbol{\Sigma}, \boldsymbol{\Phi}]=\operatorname{svd}(\mathbf{X})\);
    Set \(\psi_{i}=\Psi_{., i} \in \mathbb{R}^{m}\) and \(\lambda_{i}=\boldsymbol{\Sigma}_{i i}^{2}\) for \(i=1, \cdots, r\);
    return POD basis \(\psi_{i=1}^{r}\) and eigenvalues \(\lambda_{i i=1}^{r}\)
```


### 3.2.3. Approximation of surface example

To demonstrate the application of the POD method in the selection of optimal basis elements, an illustration will be provided where the surface of a given function, which exhibits spatial and temporal variations, is approximated. This particular example has been extracted from the article $A n$ introduction to the proper orthogonal decomposition by Anindya Chatterjee, referenced in [48]. All results present in the foreseen chapter have been obtained by running the codes provided by the author in MATLAB.

Let $\mathbf{z}$ be given by

$$
\begin{equation*}
\mathbf{z}(x, t)=\mathrm{e}^{-|(x-0.5)(t-1)|}+\sin (x t), \quad 0 \leq x \leq 1, \quad 0 \leq t \leq 2 . \tag{3.4}
\end{equation*}
$$

The function will be measured at 25 equidistant $x$ points and 50 equidistant time instants $t$. The surface $\mathbf{z}(x, t)$ defined in Equation (3.4) can be seen initially displayed in the top-left corner of Figure 3.11.

By organizing the data into a matrix $\mathbf{Z}$, the SVD of such a matrix is computed. Subsequently, rank approximations of $\mathbf{Z}$ are calculated, ranging from 1 to 3, as depicted in Figure 3.11.

Rank 3 approximation, shown in the bottom-right corner of Figure 3.11, appears to be almost indistinguishable from the actual surface. This is explained in Figure 3.12 (a), which depicts the singular values of $\mathbf{Z}$. Observe how the size of the singular values rapidly decreases, with the fourth one being considerably lower than the third (note how the scale is logarithmic).


Figure 3.11.: (top-left) Representation of the actual surface $\mathbf{z}(x, t))$. (others) Rank approximation of the surface extracted using SVD (Source: own).

In this particular example, the focus has been on generating lower-rank approximations of the data, with the usage of the SVD considered incidental to the computations. To interpret the results in terms of mode shapes within the context of the POD, one can examine the first three columns of matrix $\mathbf{V}$, which represent the three primary mode shapes in the $x$-direction. By projecting the data onto these mode shapes, the temporal evolution of the corresponding modal coordinates can be obtained.

The computation of modal coordinates is straightforward. From the definition established in Equation (3.2), one could discretize such an equation so as to write a matrix product instead of a function product. Introducing matrix $\mathbf{Q}$ such that $\mathbf{U} \boldsymbol{\Sigma}=\mathbf{Q}$ from the SVD operation, where $\mathbf{Q} \in \mathbb{R}^{n \times m}$ and $\mathbf{A}=\mathbf{Q V}^{T}$, the next product may be written

$$
\mathbf{A}=\mathbf{Q} \mathbf{V}^{T}=\sum_{k=1}^{m} q_{k} v_{k}^{T} .
$$

where $q_{k}$ is the $k$-th column of $\mathbf{Q}$ and $v_{k}$ is the $k$-th column of $\mathbf{V}$. Function $\mathbf{z}(x, t)$ is represented as matrix $\mathbf{A}$, the time-dependent function $a_{i}(t)$ is represented by the column matrix $q_{k}$ and the spatial-dependent function $\psi_{i}(x)$ is represented by the row matrix $v_{k}^{T}$.
Therefore, the $k$-th modal coordinate $q_{k}$ is derived by multiplying $u_{k}$, representing the $k$-th column of matrix $\mathbf{U}$ (assuming $\mathbf{U}$ is available through the SVD), by the singular value $\sigma_{k}$. Alternatively, if only the proper orthogonal modes $\mathbf{V}$ are accessible, the projection calculation simplifies to $q_{k}=\mathbf{Z} v_{k}$, where $v_{k}$ denotes the $k$-th column of matrix $\mathbf{V}$. The modal coordinates for the surface are visualized in Figure 3.12 (b). The dominance of the first coordinate is evident (corresponding to the most prominent singular value), while the magnitudes of the second and third coordinates are comparable (approximately equal to the second and third singular values).

It is worth noting that the computation of modal coordinates provides valuable insights into the underlying dynamics of the data. These coordinates serve as a representation of the contribution of each mode shape to the overall behavior of the system.


Figure 3.12.: SVD and POD analysis of the approximation performed
(Source: Own).

### 3.3. Fourier and Wavelet Transform

One of the most foundational and ubiquitous coordinate transformation was introduced by J.B. Joseph Fourier in the early 1800s to investigate the theory of heat [49]. Fourier derived the Fourier transform (and the Fourier series in particular) as a way of approximating solutions of PDEs. He was particularly interested in the heat equation, where he discovered that the Fourier transform was a coordinate transformation capable of diagonalizing the Laplacian operator in the heat equation. What he showed was that the Laplacian (i.e., a mathematical operator) contained eigenvalues and eigenfunctions, which were sines and cosines functions of a particular frequency determined by the boundary conditions and geometry of the object, and that the corresponding eigenvalues were those special frequencies.

Since then, it has been used for all kinds of applications in fields like image compression and solving other partial differential equations. In fact, the singular value decomposition (SVD) studied in the first section of this chapter can be thought of as a data-driven extension of the fast Fourier transform (FFT).

### 3.3.1. Brief introduction

The Fourier transform is introduced as a valuable coordinate transformation for addressing various types of problems, such as data representation, image compression, and solving complex sets of partial differential equations. Before delving into the computational implementation of Fourier transforms on data vectors, it is essential to introduce the concepts of the analytic Fourier series and Fourier transform, which are defined for continuous functions.

A key result in Fourier analysis states that if $f(x)$ is a periodic and piecewise smooth function, it can be expressed as a Fourier series. This series serves as a decomposition or approximation of the function $f(x)$ by an infinite sum of cosines and sines with increasing frequencies. Specifically, when $f(x)$ is $2 \pi$-periodic, it can be written as:

$$
\begin{equation*}
f(x)=\frac{a_{0}}{2}+\sum_{k=1}^{\infty}\left(a_{k} \cos (k x)+b_{k} \sin (k x)\right), \tag{3.5}
\end{equation*}
$$

where the coefficients $a_{k}$ and $b_{k}$ contain the information required to reconstruct the original function $f(x)$ by combining the trigonometric functions or higher-frequency terms. These coefficients can be obtained using the following equations:

$$
\begin{equation*}
a_{k}=\frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos (k x) d x, b_{k}=\frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin (k x) d x . \tag{3.6}
\end{equation*}
$$

It is worth noting the resemblance between the coefficients and the inner product between the function $f(x)$ and the specific trigonometric function associated with index $k$. In other words, the integrals in Equation Equation (3.6) can be rewritten as:

$$
a_{k}=\frac{1}{\|\cos (k x)\|^{2}}\langle f(x), \cos (k x)\rangle, \quad b_{k}=\frac{1}{\|\sin (k x)\|^{2}}\langle f(x), \sin (k x)\rangle .
$$

A geometric interpretation of the Fourier series meaning is helpful in understanding the concepts presented. In Equation (3.5), the function $f(x)$ is expressed as a sum of $a_{k}$ times cosines of $k x$ and $b_{k}$ times sines of $k x$. As previously mentioned, these coefficients $a_{k}$ and $b_{k}$ represent the projections of the function onto the corresponding cosine and sine waves.

Figure 3.13 illustrates two sets of orthogonal bases in a 2-dimensional vector space. One basis comprises the vectors $\vec{x}$ and $\vec{y}$, while the other basis is composed of orthogonal vectors denoted as $\vec{u}$ and $\vec{v}$. Within this vector space, consider the test vector $\vec{f}$. It is possible to represent $\vec{f}$ in either the $(\vec{x}, \vec{y})$ coordinate system or the $(\vec{u}, \vec{v})$ coordinate system.

The representation is straightforward. To express $\vec{f}$ in the $(\vec{x}, \vec{y})$ coordinate system, the projection of $\vec{f}$ onto the $x$-direction, given by the inner product of $\vec{f}$ and $\vec{x}$, is considered. The same procedure applies to the $(\vec{u}, \vec{v})$ coordinate system:

$$
\vec{f}=\langle\vec{f}, \vec{x}\rangle \frac{\vec{x}}{\|\vec{x}\|^{2}}+\langle\vec{f}, \vec{y}\rangle \frac{\vec{y}}{\|\vec{y}\|^{2}} \quad, \quad \vec{f}=\langle\vec{f}, \vec{u}\rangle \frac{\vec{u}}{\|\vec{u}\|^{2}}+\langle\vec{f}, \vec{v}\rangle \frac{\vec{v}}{\|\vec{v}\|^{2}} .
$$

The purpose of this representation is to demonstrate that the definition of the Fourier series aligns with how vectors are expressed on an orthogonal basis in a two-dimensional vector space like $\mathbb{R}^{2}$. The Fourier series utilizes orthogonal functions, similar to how $\vec{x}$ and $\vec{y}$ are orthogonal vectors. By applying the Fourier series, it becomes possible to determine the extent to which $\vec{f}$ aligns with the cosine direction, represented by the $a_{k}$ coefficient.


Figure 3.13.: Change of coordinate of a vector in two dimensions, extracted from [4].

Thus, the Fourier series can be seen as a means of writing $\vec{f}$ in an orthogonal basis of sines and cosines, akin to writing vectors in an orthogonal basis. Lastly, it is important to note that the Fourier series is particularly useful for approximation, as Equation (3.6) provides an exact equality. However, in practice, it is often sufficient to retain a subset of these elements, such as $k=\{1, \cdots, r\}$, to obtain a satisfactory approximation of the original function.

### 3.3.2. General definition

In the following section, the generalization of the Fourier transform from periodic functions to functions defined on an infinite domain will be discussed. The Fourier series concept emerges from the notion that an arbitrary function, denoted as $f(x)$, which exhibits periodicity within a specific domain (e.g., $[-\pi, \pi]$ or $[-L, L]$ ), can be expanded as a sum of periodic sines and cosines. These sines and cosines serve as the fundamental building blocks for higher harmonics, enabling the expansion of $f(x)$. By allowing the upper limit of the domain, denoted as $L$, to tend towards infinity $(L \longrightarrow \infty)$, the Fourier transform is obtained.

The representation of the Fourier series is considered, and the case where the upper domain approaches infinity is examined. This transformation provides a representation of the function $f(x)$ that is no longer strictly periodic and allows for non-zero values at specific points. Consequently, arbitrary functions spanning from negative infinity to positive infinity can be represented using the Fourier transform.

Firstly, the Fourier series defined on the domain $x \in[-L, L)$ is given by:

$$
f(x)=\frac{a_{0}}{2}+\sum_{k=1}^{\infty}\left[a_{k} \cos \left(\frac{k \pi x}{L}\right)+b_{k} \sin \left(\frac{k \pi x}{L}\right)\right]=\sum_{k=-\infty}^{\infty} c_{k} e^{i k \pi x / L},
$$

where the coefficients are determined as

$$
c_{k}=\frac{1}{2 L}\left\langle f(x), \psi_{k}\right\rangle=\frac{1}{2 L} \int_{-L}^{L} f(x) e^{-i k \pi x / L} d x .
$$

The approximation of $f(x)$ is represented by a sum of sines and cosines with discrete frequencies given by $\omega_{k}=k \pi / L=k \Delta \omega$, where $\Delta \omega=\pi / L$. As the limit $L \longrightarrow \infty$ is taken, the quantity $\Delta \omega$ tends to zero $(\Delta \omega \longrightarrow 0)$. Consequently, the resolution to distinguish different frequencies becomes infinitesimally small, and $\Delta \omega$ can be expressed as a differential $\omega(d \omega)$, resulting in:

$$
f(x)=\lim _{\Delta \omega \rightarrow 0} \sum_{k=-\infty}^{\infty} \frac{\Delta \omega}{2 \pi} \underbrace{\int_{-\pi / \Delta \omega}^{\pi / \Delta \omega} f(\xi) e^{-i k \Delta \omega \xi} d \xi}_{\left\langle f(x), \psi_{k}(x)\right\rangle} e^{i k \Delta \omega x}
$$

Here, the Fourier series coefficients $c_{k}$ are substituted, and the limit is taken as $L$ tends to infinity ( $\Delta \omega \longrightarrow 0$ ). Upon analyzing this expression, as $\Delta \omega$ approaches zero, the summation is transformed into an integral spanning from negative infinity to infinity with respect to the variable $\omega$. Therefore, the equation can be rewritten as:

$$
f(x)=\int_{-\infty}^{\infty} \frac{1}{2 \pi} \underbrace{\int_{-\infty}^{\infty} f(\xi) e^{i \omega \xi} d \xi}_{\hat{f}(\omega)} e^{i \omega x} d \omega
$$

Consequently, the Fourier series coefficients are incorporated into a comprehensive expression, and the limit is taken as $L \longrightarrow \infty$ or $\Delta \omega \longrightarrow 0$. The resulting summation is then converted into a Riemann integral, integrating with respect to a dummy variable $\xi$ and the frequency $\omega$. The inner integral, denoted as $\hat{f}(\omega)$, corresponds to what is known as the Fourier transform.

### 3.3.3. Discrete Fourier Transform

The computation of these quantities on a computer using the discrete Fourier transform (DFT) will now be described. Instead of approximating the infinite Fourier transform integral, the focus is on approximating the finite Fourier series. This concept plays a crucial role and eventually leads to the fast Fourier transform (FFT), which stands as one of the most powerful and significant algorithms of the previous century. The discrete Fourier transform represents a mathematical transformation that can be expressed as a matrix multiplication, while the fast Fourier transform presents a computationally efficient method for computing the DFT, particularly suited for large data sets. In essence, the FFT serves as the algorithm employed for computing the DFT.

Earlier, the approximation of periodic functions using infinite sums of sines and cosines was discussed. However, in many cases, an analytical function is not available, and instead, measurement data obtained from experiments or simulations is utilized. The typical scenario involves having discrete values of $f(x)$ at specific locations, such as $x_{1}, x_{2}, x_{3}, \ldots, x_{n}$, resulting in a data vector $\mathbf{f}=\left[f_{1}, f_{2}, f_{3}, \ldots, f_{n}\right]^{T}$.


Figure 3.14.: Discrete data sampled for the discrete Fourier transform, extracted from [4].

The objective is to obtain a vector of Fourier coefficients in the form of $\hat{\mathbf{f}}=\left[\hat{f}_{1}, \hat{f}_{2}, \hat{f}_{3}, \ldots, \hat{f}_{n}\right]^{T}$ for each of these data points $f_{k}$. Analogous to the conversion of the function $f(x)$ into coefficients that multiply the sines and cosines, the data $\mathbf{f}$ is transformed into the Fourier transform vector $\hat{\mathbf{f}}$, consisting of frequency components. Each $\hat{f}_{k}$ represents the contribution of the corresponding frequency to the data, where $\hat{f}_{n}$ corresponds to the highest frequency possible with $n$ data points.

To obtain the aforementioned data vector $\hat{\mathbf{f}}$, the simplest formulation of the DFT is employed, as expressed by

$$
\hat{f}_{k}=\sum_{j=0}^{n-1} f_{j} e^{-i 2 \pi j k / n},
$$

where the $k$-th Fourier coefficient is computed by summing the contributions of all $j$ data points at the $j$-th frequency, scaled by the $k$-th frequency divided by $n$. The interpretation of the exponential term $e^{-i 2 \pi j k / n}$ will be provided later.

Similarly, as observed in a Fourier transform or Fourier series, the inverse discrete Fourier transform (iDFT) allows the reconstruction of the original data if the Fourier transform is available. The iDFT is defined as

$$
f_{k}=\frac{1}{n} \sum_{j=0}^{n-1} \hat{f}_{j} e^{i 2 \pi j k / n} .
$$

In essence, given the data vector $\mathbf{f}=\left[f_{1}, f_{2}, f_{3}, \ldots, f_{n}\right]^{T}$, performing the discrete Fourier transform yields the Fourier frequencies $\hat{\mathbf{f}}=\left[\hat{f}_{1}, \hat{f}_{2}, \hat{f}_{3}, \ldots, \hat{f}_{n}\right]^{T}$. These frequencies represent the contributions necessary to reconstruct the data in $\mathbf{f}$, expressed as

$$
\left\{f_{1}, f_{2}, \cdots, f_{n}\right\} \stackrel{\text { DFT }}{\Longrightarrow}\left\{\hat{f}_{1}, \hat{f}_{2}, \cdots \hat{f}_{n}\right\}
$$

It is important to note that all terms in the series involve the multiplication of exponentials, which are integer multiples of $e$ raised to the power of $i 2 \pi j k / n$, where $i$ denotes the complex imaginary unit $i=\sqrt{-1}$. This exponential term defines a fundamental frequency $\omega_{n}=e^{-2 \pi i / n}$, determining the range of sines and cosines that can be approximated using $n$ discrete values. This fundamental frequency serves as a fundamental unit of work. In an interval with $n$ data points, each Fourier transform involves summing integer multiples of this fundamental frequency multiplied by the data, similar to the inverse Fourier transform.

Hence, the fundamental frequency $\omega_{n}$ is utilized to compute a matrix that facilitates the multiplication of the data, resulting in the Fourier transform. Although both the Fourier transform and the inverse Fourier transform involve summations over the data, it is not practical to perform the entire summation for each $k=1,2,3, \ldots$. Such an approach would be laborious, especially when implementing it in code.

Instead, a more efficient approach involves expressing the summation for each $k$ in terms of a matrix operation. This operation consists of multiplying the data vector by a matrix to obtain the Fourier transform vector. The matrix, known as the DFT matrix, is defined as

$$
\left[\begin{array}{c}
\hat{f}_{1} \\
\hat{f}_{2} \\
\hat{f}_{3} \\
\vdots \\
\hat{f}_{n}
\end{array}\right]=\underbrace{\left[\begin{array}{ccccc}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega_{n} & \omega_{n}^{2} & \cdots & \omega_{n}^{n-1} \\
1 & \omega_{n}^{2} & \omega_{n}^{4} & \cdots & \omega_{n}^{2(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_{n}^{n-1} & \omega_{n}^{2(n-1)} & \cdots & \omega_{n}^{(n-1)^{2}}
\end{array}\right]}_{\text {DFT Matrix }}\left[\begin{array}{c}
f_{1} \\
f_{2} \\
f_{3} \\
\vdots \\
f_{n}
\end{array}\right] .
$$

In summary, it is important to note that even when dealing with a data vector instead of an analytic function, a discrete version of the Fourier series can still be computed. The transformation from data to Fourier coefficients and the inverse transformation can both be expressed as matrix operations utilizing the DFT matrix. These operations involve complex numbers, resulting in a complex-valued matrix and complex Fourier coefficients. The magnitude of each coefficient represents the presence of sine and cosine components, while the phase describes the phase relationship between these components. Therefore, with discrete data, it is possible to leverage the Fourier transform to analyze frequency components and gain insights into the underlying patterns and structures within the data.

### 3.3.4. Fast Fourier Transform

The discrete Fourier transform (DFT) matrix is computationally expensive to compute and multiply with the data. Consequently, the fast Fourier transform (FFT) algorithm has become synonymous with the DFT due to its efficiency. In practice, the FFT is used to compute the discrete Fourier transform instead of explicitly constructing and multiplying the DFT matrix. This algorithm offers significant speed and efficiency advantages.

The primary motivation for utilizing the FFT is to achieve faster computations. The DFT calculation has a computational complexity of $\mathcal{O}\left(n^{2}\right)$, meaning that as the number of data points $(n)$ increases, the number of required multiplications grows quadratically. This can result in slow and expensive computations, particularly when dealing with larger values of $n$, such as images. In contrast, the FFT achieves the same Fourier transform but with a computational complexity of $\mathcal{O}(n \log (n))$. This "fast scaling" is nearly linear in $n$, with the small logarithmic correction becoming less significant as $n$ grows larger. For example, when $n$ is a thousand, the logarithm of $n$ might be three, but when $n$ is a billion, the logarithm of $n$ is only nine. Thus, the logarithmic term does not increase as rapidly as $n$.

For the case where $n$ is a power of 2 , such as $n=1024=2^{10}$, the FFT algorithm offers further efficiency gains. By rearranging the entries of the data vector $\mathbf{f}$, the Fourier transform $\hat{\mathbf{f}}$ can be computed more efficiently. In this scenario, the DFT matrix, denoted as $\mathbf{F}_{1024}$, can be expressed as a product of smaller matrices, including the identity matrix $\left(\mathbf{I}_{512}\right)$ and the diagonal matrix $\left(\mathbf{D}_{512}\right)$.

$$
\hat{\mathbf{f}}=\mathbf{F}_{1024} \mathbf{f}=\left[\begin{array}{ll}
\mathbf{I}_{512} & -\mathbf{D}_{512} \\
\mathbf{I}_{512} & -\mathbf{D}_{512}
\end{array}\right]\left[\begin{array}{cc}
\mathbf{F}_{512} & \mathbf{0} \\
\mathbf{0} & \mathbf{F}_{512}
\end{array}\right]\left[\begin{array}{c}
\mathbf{f}_{\text {even }} \\
\mathbf{f}_{\text {odd }}
\end{array}\right]
$$

The matrices $\mathbf{F}_{512}$ represent the Fourier transform for a smaller size, and the vector $\mathbf{f}_{\text {even }}$ contains the even-index coefficients of $\mathbf{f}$, while $\mathbf{f}_{\text {odd }}$ contains the odd-index coefficients. $\mathbf{D}_{512}$ matrix is given by

$$
\mathbf{D}_{512}=\left[\begin{array}{ccccc}
1 & 0 & 0 & \cdots & 0 \\
0 & \omega & 0 & \cdots & 0 \\
0 & 0 & \omega^{2} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & \omega^{511}
\end{array}\right]
$$

By leveraging the diagonal structure of the sub-blocks, the multiplication becomes simpler and more efficient, as the sub-blocks are smaller in size (512 instead of 1024) and contain numerous zeros. Moreover, the matrices $\mathbf{F}_{512}$ can be further divided into even smaller blocks, allowing the same efficiency trick to be applied recursively. This recursive subdivision continues until reaching a $2 \times 2$ matrix $\left(\mathbf{F}_{2}\right)$, which is straightforward to multiply.

$$
\mathbf{F}_{1024} \longrightarrow \mathbf{F}_{512} \longrightarrow \mathbf{F}_{256} \longrightarrow \cdots \longrightarrow \mathbf{F}_{4} \longrightarrow \mathbf{F}_{2},
$$

The FFT takes advantage of the DFT matrix's inherent symmetry and intelligently reorders the indices to exploit redundancies and simplify calculations. By recursively reducing the computation, significant reductions in complexity are achieved, eventually resulting in a computational complexity of $\mathcal{O}(n \log (n))$.

### 3.3.5. Wavelet Transform

It is noteworthy to mention wavelets, which can be regarded as an advanced version of the Fourier transform [50]. The Fourier transform facilitates the decomposition of a function into a sum of sines and cosines, thus revealing its frequency content. It has been established that these sinusoidal functions constitute an orthogonal basis for the representation of desired functions. In this section on wavelets, a generalization of this notion will be explored, extending the concept of orthogonal basis to encompass other orthogonal functions that may offer improved representation for specific function types. Wavelets find extensive applications in image and audio compression, revolutionizing signal compression and representation in the digital age [4].

When dealing with time series data, precise temporal information about the function's temporal location is available, but there is a lack of knowledge regarding the frequencies present at that specific moment. Conversely, in a Fourier transform, the frequency content can be precisely determined, but the temporal occurrence of these frequencies remains unknown. Consequently, the Fourier domain provides high frequency resolution but significant uncertainty regarding the timing of frequency occurrences.

To address these limitations, the development of the spectrogram (as illustrated in Figure 3.15 (c)) was motivated. The spectrogram assigns equal importance to both time and frequency and can be envisioned as a grid-like structure that sacrifices some time resolution from the original time series and some frequency resolution from the Fourier transform. Nonetheless, it provides valuable insights into the activation and deactivation times of individual frequencies, thus offering partial temporal and frequency information.


Figure 3.15.: Illustration of resolution limitations and uncertainty in time-frequency analysis, extracted from [4].

The wavelet transform, also known as multi-resolution analysis [50], introduces multiple scales in both time and frequency. The observation that indicates that very low frequencies tend to persist over extended durations without significant changes over time is going to be followed. Consequently, a baseline frequency is established at the lowest level, assuming continuous activation throughout time, without concerning ourselves with precise timing.

Moving to the subsequent level, the frequency range is divided in half, sacrificing some frequency resolution but providing information about occurrence times within the time series data set. As we ascend to higher levels, more frequency resolution is obtained, as higher frequencies exhibit more rapid changes over time. However, the ability to precisely identify which high frequencies are activated diminishes.

Figure 3.15 (d) presents a multi-resolution time-frequency decomposition achieved using the wavelet decomposition technique. This representation acknowledges that lower frequencies change relatively slowly over time, needing less temporal accuracy. On the other hand, higher frequencies demand increased temporal accuracy, introducing more uncertainty regarding the exact frequency of activations and deactivations. Although the example illustrates three levels, practical applications may involve ten or more levels for this decomposition.

Therefore, the wavelet decomposition can be viewed as an optimized spectrogram, effectively allocating information where it is most relevant. It provides reduced information for low frequencies while enhancing temporal resolution for higher frequencies, and vice versa for frequency resolution.

The fundamental concept in wavelet analysis shares similarities with Fourier decomposition, involving the projection of a given time series or spatial data onto an orthogonal basis. However, in the wavelet transform, this orthogonal basis comprises a hierarchical arrangement of functions that decrease in size within temporal or spatial windows, as depicted in Figure 3.15 (d).

The core concept of wavelet analysis revolves around a function known as the mother wavelet, denoted as $\psi(t)$. From this mother wavelet, smaller wavelets can be derived using the following equation:

$$
\psi_{a, b}(t)=\frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)
$$

If we consider $\psi$ as a Gaussian function, adjusting the parameter $b$ would shift the Gaussian function in time, selecting the specific window within the time series. On the other hand, the parameter $a$ controls the size of the wavelet. As $a$ increases, smaller windows are obtained, corresponding to higher levels, while $b$ slides the window across the time signal.

Consequently, the wavelet transform, denoted by $\mathcal{W} \psi$, can be expressed as the inner product between the function $f$ and the wavelet $\psi a, b$ :

$$
\mathcal{W}_{\psi}(f)(a, b)=\left\langle f, \psi_{a, b}\right\rangle
$$

To facilitate comprehension, an example of a wavelet, the Haar wavelet, will be presented. The Haar wavelet, discovered in 1910 [51], exhibits the following expression:

$$
\psi(t)=\left\{\begin{array}{cc}
1 & 0 \leq t<1 / 2 \\
-1 & 1 / 2 \leq t<1 \\
0 & \text { otherwise }
\end{array}\right.
$$

This wavelet consists of a step function with values of 1 and -1 over the entire interval of the wavelet transformation. The first wavelet, displayed in Figure 3.16, represents the mother wavelet $\psi_{1,0}$ scaled with $a=1$ (no resizing) and $b=0$ (no shift). By utilizing this mother wavelet, smaller wavelets can be constructed, such as $\psi_{1 / 2,0}$ (scaled down by $a=1 / 2$, no shift), or $\psi_{1 / 2,1 / 2}$. The process can be continued, yielding a variety of wavelets. The function $\psi_{1,0}$ corresponds to the lower region of the decomposition presented in Figure 3.15 (d), while $\psi_{1 / 2,0}$ represents the upper-left portion, and $\psi_{1 / 2,1 / 2}$ corresponds to the right portion.

It is worth noting that all these wavelets are orthogonal functions. Consequently, the inner product of $\psi_{1,0}$ with $\psi_{1 / 2,0}$ results in 0 because, when $\psi_{1,0}$ is $1, \psi_{1 / 2,0}$ is half 1 and half -1 , leading to cancellation. Similar orthogonality relationships hold between the bottom and top wavelets. Thus, the construction of these wavelets ensures orthogonality as the shape is shrunk and divided into halves.


Figure 3.16.: Three Haar wavelets for the first two levels of the multi-resolution in Figure 3.15 (d), extracted from [4].

In summary, wavelets strike a balance between time-series analysis and the Fourier transform, resembling an enhanced spectrogram that selectively allocates resolution according to the signal's requirements. This approach is motivated by natural signals of interest to humans, where low-frequency components change relatively slowly over time, while high-frequency components exhibit more rapid variations. Consequently, the wavelet decomposition adapts the resolution based on these frequency characteristics.

## 4. Compression of equations

While techniques like SVD and Wavelet transform have been successfully used for compressing images and large data set of information, as seen in the previous chapter, applying similar compression methods to the Finite Element (FE) equations presents a unique challenge. The compression of equations refers to the process of reducing the complexity and computational cost associated with solving finite element equations at a given scale.

One approach to compressing the FE equations, and the main focus of this thesis, is the so-called projection-based, reduced-order models [8], particularly through Galerkin projection. The existence of low-dimensional representations for a particular parameterized finite element problem is based on the assumption that the state variable may be precisely estimated by a linear approximation of a few global basis vectors [14]. By doing so, the computational burden is significantly reduced, enabling faster and more efficient simulations.

Data compression algorithms, such as SVD or POD, play a crucial role in determining this reduced-order subspace. They analyze a set of representative displacement fluctuation solutions over a so-called training sample [8], and identify the dominant modes of deformation. These dominant modes represent the most significant contributors to the system's response and can be used to construct a basis for the reduced-order subspace. By retaining only these dominant modes, the dimensionality of the problem is reduced, resulting in a compressed set of equations that still accurately capture the behavior of the system [10].

### 4.1. General Overview

This research focuses on developing a cost-effective low-dimensional model for governing equations that involve nonlinear relationships between state variables and input parameters. The construction of such a model involves two sequential stages [52]:

1. Projection onto the reduced basis: In this stage, an approximation of the state variables is introduced into the governing equation. The resulting equation is formulated in the space spanned by the basis vectors, effectively reducing the number of unknowns. This simplification facilitates the process of solving the equation. However, for general nonlinear problems, evaluating the residual still depends on the size of the finite element mesh. Consequently, a second reduction stage may be necessary.
2. Approximation of the nonlinear term: This stage, known as hyper-reduction [53], presents challenges and remains a topic of discussion in the model reduction community. Various approaches have been proposed in the literature to address this additional dimensionality reduction stage.

Let $\boldsymbol{F}^{h} \in \mathbb{R}^{N}$ denote ${ }^{1}$ the full-order term with a general nonlinear relationship involving input variables and state variables. Specifically, $\boldsymbol{F}^{h}$ refers to the vector of finite element nodal internal forces. The projection onto the reduced-order space is denoted by $\boldsymbol{F} \in \mathbb{R}^{n}$ ( $n \ll N$ ), and the connection between these variables is established through the basis vector matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times n}\left(\boldsymbol{F}=\boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{F}^{\boldsymbol{h}}\right)$. Approaches for approximating $\boldsymbol{F}$ can be broadly classified into nodal vector approaches and integral approaches [8].

### 4.1.1. Nodal Vector Approximation Approaches

In nodal vector approximation approaches, the finite element vector $\boldsymbol{F}^{h}$ is approximated by a low-dimensional interpolant $\boldsymbol{F}^{h} \approx \boldsymbol{R}_{\boldsymbol{F}} \boldsymbol{F}_{z}^{h}$. Here, $\boldsymbol{R}_{\boldsymbol{F}} \in \mathbb{R}^{N \times m}$ is the interpolation matrix, and $\boldsymbol{F}_{z}^{\boldsymbol{h}}$ represents the entries of $\boldsymbol{F}^{h}$ corresponding to the degrees of freedom $(\boldsymbol{z} \subset\{1,2, \cdots, N\})$ where interpolation occurs. The interpolation matrix is obtained by computing a basis matrix for $\boldsymbol{F}^{h}$ and determining a set of indices that minimize the error over a representative set of snapshots of $\boldsymbol{F}^{h}$. Offline methods like the Empirical Interpolation Method (EIM) [54] can be used to determine these interpolation indices. The nodal vector approximation approach originated from the work of Everson and Sirovich [55], being the first proposed method for handling nonlinear terms in model order reduction.

### 4.1.2. Integral Approximation Approaches

In the finite element context, $\boldsymbol{F}$ can be seen not only as a projection of a large vector into a reduced-order space $\left(\boldsymbol{F}^{h}=\boldsymbol{\Phi} \boldsymbol{F}^{h}\right)$ but also as the result of integrating over the concerned domain $\Omega \subset \mathbb{R}^{d}$ (with $d=2$ in this case), where the reduced-order variable $\boldsymbol{f}=\boldsymbol{\Psi}^{\boldsymbol{T}} \boldsymbol{f}^{h}\left(\boldsymbol{f}: \Omega \longrightarrow \mathbb{R}^{N}\right)$. Consequently, the problem can be viewed as the approximation of an integral rather than a vector. This can be addressed by either seeking a low-dimensional approximation of the integrand or approximating the integral itself as a weighted sum of the integrand evaluated at optimal sampling points.

1. Interpolation of the integrand: In this approach, the procedure is similar to vector approaches, with the difference being that the interpolation is performed on the integrand rather than the integral [56]. By approximating $\boldsymbol{f}(x)$ as a sum of interpolation functions $\boldsymbol{R}_{g}(\boldsymbol{x})$ evaluated at the sampling points $x_{g} \in \boldsymbol{z}$, the integral can be expressed as a sum of the product of matrix weights $\boldsymbol{Q}_{g}$ and the integrand evaluated at the interpolating points.
2. Cubature methods: Cubature methods approximate the integral by a finite sum of positive scalar weights $\left\{\omega_{g}\right\}_{g=1}^{m}$ multiplied by the integrand at selected sampling points. These methods [57,58] are based on the principles of Gaussian quadrature for polynomial functions. The strategy involves selecting a reduced set of points and associated positive weights from the integration points of the FE mesh. This selection aims to minimize the integration error over representative samples of the integrand. The use of positive scalar weights in cubature methods ensures that the Jacobian matrix maintains the spectral properties of its full-order counterpart, preserving symmetry and positive definiteness in structural problems. This advantage sets cubature methods apart from interpolatory schemes, which can have a negative impact on the robustness of finite element models [59].
[^4]This research employs a novel cubature approach called the Empirical Cubature Method (ECM) [8], whose resolution guideline is introduced in Section 4.4.1.

### 4.2. Parametrized Finite Element Model

The model aiming to be simplified is a parametrized nonlinear equation known as the full-order or high-fidelity model. It represents the motion of a system using the finite element method in a Lagrangian formulation. The equation can be expressed as:

$$
\begin{equation*}
\boldsymbol{M}^{h} \ddot{\boldsymbol{d}}^{h}(\boldsymbol{\mu})+\boldsymbol{F}^{h}\left(\boldsymbol{d}^{h}, \boldsymbol{d}_{0}^{h} ; \boldsymbol{\mu}\right)=\boldsymbol{F}_{\mathrm{ext}}^{h}(\boldsymbol{\mu})-\boldsymbol{M}_{0}^{h} \ddot{\boldsymbol{d}}_{0}^{h}(\boldsymbol{\mu}) \tag{4.1}
\end{equation*}
$$

In this equation, $\boldsymbol{d}^{h} \in \mathbb{R}^{N}$ and $\boldsymbol{d}_{0}^{h} \in \mathbb{R}_{0}^{N}$ represent the vectors of unknown and prescribed nodal displacements, respectively. The sizes of these vectors are determined by the number of unrestricted degrees of freedom $N$ and the number of restricted degrees of freedom $N_{0}$, which remain constant throughout the analysis due to fixed Dirichlet boundaries.

The double dot notation denotes the second derivative with respect to time, representing the acceleration $\left(\ddot{\boldsymbol{d}}^{h}=\partial^{2} \boldsymbol{d}^{h} / \partial t^{2}\right)$. The parameter set is denoted by $\boldsymbol{\mu}$, and its corresponding space is $\mathcal{D}$. These parameters can include variations in prescribed boundary conditions, body forces, material properties, and other factors.

The mass matrices $\boldsymbol{M}^{h} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{M}_{0}^{h} \in \mathbb{R}^{N \times N_{0}}$ represent the inertial forces caused by the acceleration of unrestricted and restricted degrees of freedom, respectively. These matrices are assumed to be independent of the parameter set $\boldsymbol{\mu}$.

On the other hand, $\boldsymbol{F}^{h} \in \mathbb{R}^{N}$ and $\boldsymbol{F}_{e x t}^{h} \in \mathbb{R}^{N}$ are the vectors of nodal internal and external forces, respectively. For simplicity, it is assumed that $\boldsymbol{F}_{\text {ext }}^{h}$ does not depend on the state variables. The integration rule used to evaluate $\boldsymbol{F}^{h}$ involves pairs $\left\{\boldsymbol{x}_{g}, W_{g}\right\}_{g=1}^{M}$, where $\boldsymbol{x}_{g} \in \Omega$ represents the position of the $g$-th integration point, $W_{g}$ is the corresponding weight (including the Jacobian of the finite element at that point), and $M$ is the total number of integration points. Consequently, it can be approximated $\boldsymbol{F}^{h}$ as:

$$
\begin{equation*}
\boldsymbol{F}^{h}=\int_{\Omega} \boldsymbol{f}^{h} d \Omega \approx \sum_{g=1}^{M} W_{g} \boldsymbol{f}^{h}\left(\boldsymbol{x}_{g}\right) \tag{4.2}
\end{equation*}
$$

where $\boldsymbol{f}^{h}\left(\boldsymbol{x}_{\boldsymbol{g}}\right): \mathcal{D} \longrightarrow \mathbb{R}^{N}$ represents the sparse internal force vector at the $g$-th integration point. In a small strain setting, $\left.\boldsymbol{f}^{h}\left(\boldsymbol{x}_{\boldsymbol{g}}\right)=\boldsymbol{B}^{h^{T}}\left(\boldsymbol{x}_{\boldsymbol{g}}\right) \boldsymbol{\sigma}\right)\left(\boldsymbol{x}_{\boldsymbol{g}}\right)$, where $\boldsymbol{B}^{h} \in \mathbb{R}^{N \times s}$ is the classical (global) strain-displacement finite element matrix at the point $\boldsymbol{x}_{\boldsymbol{g}}$, and $\boldsymbol{\sigma} \in \mathbb{R}^{s}$ is the stress vector (where $s=4$ for plane stress/strain problems). The nonlinearity between $\boldsymbol{f}^{h}$ and the state variable $\boldsymbol{d}^{h}$ can arise from geometric effects (large strains) and/or material behavior.

Lastly, the constitutive relationship between the stress vector and both $\boldsymbol{\mu}$ and the deformation history at each integration point is represented by:

$$
\left.\mathcal{H}\left(\boldsymbol{\sigma}, \boldsymbol{d}^{h}, \boldsymbol{d}_{0}^{h}, \boldsymbol{\xi} ; \boldsymbol{\mu}\right)\right|_{\boldsymbol{x}_{g}}=\mathbf{0}, \quad g=1,2, \ldots, M
$$

where $\boldsymbol{\xi}$ represents the vector of internal variables. The statement is summarized in Box I.

### 4.3. First Reduction Stage

### 4.3.1. Unknown Nodal Displacements

The concept of model reduction is based on the idea that, for any given input parameter $\boldsymbol{\mu} \in \mathcal{D}$, the displacement solution can be approximated using a set of $n$ linearly independent basis vectors $\boldsymbol{\Phi}_{i} \in \mathbb{R}^{N}(i=1,2, \cdots, n)$, where $n \ll N$. This approximation can be expressed as follows:

$$
\begin{equation*}
\boldsymbol{d}^{h}(\boldsymbol{\mu}) \approx \boldsymbol{\Phi} \boldsymbol{d}(\boldsymbol{\mu}), \tag{4.3}
\end{equation*}
$$

where $\boldsymbol{\Phi}=\left[\boldsymbol{\Phi}_{1} \boldsymbol{\Phi}_{2} \cdots \boldsymbol{\Phi}_{n}\right]$ is the displacement basis matrix, and $\boldsymbol{d} \in \mathbb{R}^{n}$ is the vector of unknown reduced displacements.

To compute the basis matrix $\boldsymbol{\Phi}$, one common approach involves solving the full-order problem for representative values of the input parameters $\{\boldsymbol{\mu}\}_{j=1}^{P}\left(\boldsymbol{\mu}^{j} \in \mathcal{D}\right)$. The corresponding solutions are collected in a snapshot matrix:

$$
\boldsymbol{X}_{d}:=\left[\begin{array}{llll}
\boldsymbol{d}^{h}\left(\boldsymbol{\mu}^{1}\right) & \boldsymbol{d}^{h}\left(\boldsymbol{\mu}^{2}\right) & \cdots & \boldsymbol{d}^{h}\left(\boldsymbol{\mu}^{P}\right)
\end{array}\right],
$$

which is then processed using data compression algorithms, such as singular value decomposition (SVD) or other compression techniques. By applying SVD, the approximation $\boldsymbol{X}_{d} \approx \boldsymbol{\Phi} \boldsymbol{\Sigma}_{\boldsymbol{\Phi}} \boldsymbol{V}_{\boldsymbol{\Phi}}^{T}$ is obtained, where $\boldsymbol{\Sigma}_{\boldsymbol{\Phi}}$ and $\boldsymbol{V}_{\boldsymbol{\Phi}} \in \mathbb{R}^{P \times n}$ are truncated matrices of singular values and right singular vectors, respectively.

### 4.3.2. Prescribed Nodal Displacements and External Forces

To create a Reduced Order Model (ROM) that is independent of the size of the underlying Finite Element (FE) mesh, it is necessary to approximate not only the solution vector $\boldsymbol{d}^{h}$ but also the input vectors of prescribed displacements ( $\boldsymbol{d}_{0}^{h}$ ) and external forces $\left(\boldsymbol{F}_{e x t}^{h}\right)$. Computational savings achieved by compressing the input vectors may be smaller than the savings obtained from approximating unknown displacements and internal forces.

Typically, both the prescribed displacements and external forces (i.e., the input vectors) can be approximated exactly as linear combinations of a few spatial basis vectors. For example, if the prescribed displacements are uniform in space, a single spatial mode is sufficient. If the spatial variation is linear, then two spatial modes are needed, and so on. The coefficients in these linear combinations can be obtained through interpolation.

Let's consider the nodal vector of prescribed displacements, denoted as $\boldsymbol{d}_{0}^{h}(\boldsymbol{\mu})$. Assuming an exact approximation, it can be expressed as:

$$
\begin{equation*}
\boldsymbol{d}_{0}^{h}(\boldsymbol{\mu})=\left.\left(\boldsymbol{\Xi} \boldsymbol{\Xi}_{\mathbf{b}}^{-1}\right) \boldsymbol{d}_{0}^{h}\right|_{\mathbf{b}}(\boldsymbol{\mu}) \tag{4.4}
\end{equation*}
$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{N_{0} \times n_{0}}$ is the corresponding basis matrix, $\mathbf{b} \subseteq\left\{1,2, \cdots, N_{0}\right\}$ is a set of $n_{0}$ admissible interpolation indices, $\boldsymbol{\Xi}_{b}$ denotes the block matrix of $\boldsymbol{\Xi}$ corresponding to rows $\mathbf{b}$ (which is invertible), and $\left.\boldsymbol{d}_{0}^{h}\right|_{\mathbf{b}}$ represents the entries of $\boldsymbol{d}_{0}^{h}$ corresponding to indices $\mathbf{b}$. By introducing the variables:

$$
\boldsymbol{\Phi}_{0}:=\boldsymbol{\Xi} \boldsymbol{\Xi}_{\mathbf{b}}^{-1} \quad \text { and } \quad \boldsymbol{d}_{0}:=\left.\boldsymbol{d}_{0}^{h}\right|_{\mathbf{b}} .
$$

The expression for $\boldsymbol{d}_{0}^{h}(\boldsymbol{\mu})$ in Equation (4.4) can be written in a format similar to the approximation of the unknown displacement in Equation (4.3):

$$
\begin{equation*}
\boldsymbol{d}_{0}^{h}(\boldsymbol{\mu})=\boldsymbol{\Phi}_{0} \boldsymbol{d}_{0}(\boldsymbol{\mu}) \tag{4.5}
\end{equation*}
$$

The basis matrix $\boldsymbol{\Xi}$ can be obtained by collecting the vector of prescribed displacements for the training input parameters in a single matrix $\boldsymbol{X}_{d 0} \in \mathbb{R}^{N_{0} \times P}$, and applying the singular value decomposition (SVD) to obtain the matrix of left singular vectors. The set of interpolation indices $\mathbf{b}$ can be determined using methods like the Discrete Empirical Interpolation Method (DEIM) [60].

A similar approach can be used to construct the interpolant of $\boldsymbol{F}_{e x t}^{h}$. By expressing it as:

$$
\begin{equation*}
\boldsymbol{F}_{e x t}^{h}=\left.\left(\boldsymbol{\Theta} \boldsymbol{\Theta}_{c}^{-1}\right) \boldsymbol{F}_{e x t}^{h}\right|_{e} \tag{4.6}
\end{equation*}
$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{N \times n_{f}}$ and $\boldsymbol{c} \subset\{1,2, \cdots, N\}$ are the corresponding basis matrix and set of interpolation points, respectively.

### 4.3.3. Projection onto the Reduced-Order Space

By substituting the expressions for the unknown displacement in Equation (4.3) and the approximated prescribed displacements in Equation (4.5) into the FE balance Equation (4.1) and performing a Galerkin projection by multiplying by $\boldsymbol{\Phi}^{T}$, it is obtained:

$$
\begin{equation*}
\left(\boldsymbol{\Phi}^{T} \boldsymbol{M}^{h} \boldsymbol{\Phi}\right) \ddot{\boldsymbol{d}}+\boldsymbol{\Phi}^{T} \boldsymbol{F}^{h}=\boldsymbol{\Phi}^{T} \boldsymbol{F}_{e x t}^{h}-\left(\boldsymbol{\Phi}^{T} \boldsymbol{M}_{0}^{h} \boldsymbol{\Phi}_{0}\right) \ddot{\boldsymbol{d}}_{0} \tag{4.7}
\end{equation*}
$$

Defining the reduced mass matrices $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{M}_{0} \in \mathbb{R}^{n \times n_{0}}$ as:

$$
\boldsymbol{M}:=\boldsymbol{\Phi}^{T} \boldsymbol{M}^{h} \boldsymbol{\Phi} \quad \text { and } \quad \boldsymbol{M}_{0}:=\boldsymbol{\Phi}^{T} \boldsymbol{M}_{0}^{h} \boldsymbol{\Phi}_{0}
$$

and the reduced vectors of internal $\boldsymbol{F} \in \mathbb{R}^{n}$ and external forces $\boldsymbol{F}_{\text {ext }} \in \mathbb{R}^{n}$ as:

$$
\begin{equation*}
\boldsymbol{F}=\boldsymbol{\Phi}^{T} \boldsymbol{F}^{h} \quad \text { and } \quad \boldsymbol{F}_{e x t}=\boldsymbol{\Phi}^{T} \boldsymbol{F}_{e x t}^{h} \tag{4.8}
\end{equation*}
$$

the balance equation in Equation (4.7) can be rewritten as:

$$
\begin{equation*}
\boldsymbol{M} \ddot{\boldsymbol{d}}+\boldsymbol{F}=\boldsymbol{F}_{e x t}-\boldsymbol{M}_{0} \ddot{\boldsymbol{d}}_{0} \tag{4.9}
\end{equation*}
$$

The initial and boundary conditions in the ROM become:

$$
\boldsymbol{d}_{0}=\boldsymbol{g}_{b}^{h}(\boldsymbol{\mu}), \quad \boldsymbol{d}(0)=\boldsymbol{u}_{0}, \quad \text { and } \quad \dot{\boldsymbol{d}}(0)=\boldsymbol{\nu}_{0}
$$

where $\boldsymbol{u}_{0}=\boldsymbol{\Phi}^{T} \boldsymbol{u}_{0}^{h}$ and $\boldsymbol{\nu}_{0}=\boldsymbol{\Phi}^{T} \boldsymbol{\nu}_{0}^{h}$.
Finally, by substituting Equation (4.6) into the second Equation (4.8), the reduced external force vector can be represented as:

$$
\boldsymbol{F}_{e x t}=\left.\boldsymbol{R}_{e x t} \boldsymbol{F}_{e x t}^{h}\right|_{e}
$$

where $\boldsymbol{R}_{e x t}=\boldsymbol{\Phi}^{T}\left(\boldsymbol{\Theta} \boldsymbol{\Theta}_{c}^{-1}\right)$.

### 4.3.4. Internal Forces

The only remaining term in the reduced balance Equation (4.9) that depends on the complexity of the finite element (FE) mesh is the reduced vector of internal forces $\boldsymbol{F} \in \mathbb{R}^{n}$. By multiplying Equation (4.2) by $\boldsymbol{\Phi}^{T}$, then:

$$
\boldsymbol{F}=\int_{\Omega} \boldsymbol{\Phi}^{T} \boldsymbol{f}^{h} d \Omega \approx \sum_{g=1}^{M} W_{g} \boldsymbol{f}\left(\boldsymbol{x}_{g},: \cdot\right),
$$

where $\boldsymbol{f}:=\boldsymbol{\Phi}^{T} \boldsymbol{f}^{h}$. To complete the model-order reduction process, a more efficient integration rule is required for $\boldsymbol{F}$ that takes into account the fact that displacements, strains, stresses, and internal forces now reside in low-dimensional spaces. This step, thus, is to be performed in a second reduction stage, where the "reference" model is no longer the finite element model itself but the ROM described in this first reduction stage.

For the convenience of the reader and ease of reference, the offline operations necessary to generate such a reduced-order problem have been resumed in an algorithm exposed in Algorithm 3. Additionally, the statement of the model itself is summarized in Box II.

```
Algorithm 3 First Reduction Stage
Require: Representative input parameters \(\left\{\boldsymbol{\mu}^{i}\right\}_{i=1}^{P}\)
Ensure: Reduced-order matrices and vectors
    1: Solve the finite element problem (from Box I) for each \(\boldsymbol{\mu}^{i}\)
    : Store the resulting vectors of unrestricted nodal displacements \(\boldsymbol{d}^{h}\), prescribed nodal
    displacements \(\boldsymbol{d}_{0}^{h}\), and external forces \(\boldsymbol{f}_{\text {ext }}\) in snapshot matrices \(\boldsymbol{X}_{d}, \boldsymbol{X}_{d 0}\), and \(\boldsymbol{X}_{\text {fext }}\),
    respectively
    3: Apply SVD or any other compression technique to \(\boldsymbol{X}_{d}, \boldsymbol{X}_{d 0}\), and \(\boldsymbol{X}_{f e x t}\), obtaining basis
    matrices \(\boldsymbol{\Phi} \in \mathbb{R}^{N \times n}, \boldsymbol{\Psi} \in \mathbb{R}^{N_{0} \times n_{0}}\), and \(\boldsymbol{\Theta} \in \mathbb{R}^{N \times n_{f}}\)
    Determine the interpolation indices \(\boldsymbol{b} \subseteq\left\{1,2, \cdots, N_{0}\right\}\) and \(\boldsymbol{c} \subset\{1,2, \cdots, N\}\) of the basis
    matrices \(\boldsymbol{\Xi}\) and \(\boldsymbol{\Theta}\) using, for instance, DEIM
    Compute reduced-order matrices: \(\boldsymbol{\Phi}_{0}=\boldsymbol{\Xi} \boldsymbol{\Xi}_{b}^{\boldsymbol{- 1}}, M=\boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{M}^{h} \boldsymbol{\Phi}, M_{0}=\boldsymbol{\Phi}^{\boldsymbol{T}} M_{\mathbf{0}}^{\boldsymbol{h}} \boldsymbol{\Phi}_{\mathbf{0}}\),
    \(\boldsymbol{R}_{\mathrm{ext}}=\boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{\Theta} \boldsymbol{\Theta}_{c}^{-1}, \boldsymbol{u}_{0}=\boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{u}_{0}^{\boldsymbol{h}}\), and \(\boldsymbol{\nu}_{0}=\boldsymbol{\Phi}^{\boldsymbol{T}} \boldsymbol{\nu}_{0}^{\boldsymbol{h}}\)
```


### 4.4. Second Reduction Stage

The second reduction stage builds on the reduced-order problem stated in Box II. The main difference lies in the evaluation of the integral of the reduced internal forces, which now only requires computing the integrand at a significantly smaller number of points, denoted as $m \ll M$. Consequently, the computational effort associated with solving the constitutive equations diminishes by a factor of $M / m$, and the memory requirements for storing the history of internal variables also decrease. The reduced-order operators, namely $\boldsymbol{\Phi}_{\mathbf{0}}, \boldsymbol{M}, \boldsymbol{M}_{\mathbf{0}}, \boldsymbol{R}_{\text {ext }}$, and $\boldsymbol{B}_{\boldsymbol{g}}=\boldsymbol{B}_{\boldsymbol{g}}^{\boldsymbol{h}} \boldsymbol{\Phi}$ (where $g=1,2, \cdots, M$ ), have already been determined in the offline phase of the first reduction stage (see Box II) and, hence, do not need to be computed again. For computing the integral of the reduced internal forces, the Empirical Cubature Method (ECM) is employed.

### 4.4.1. Empirical Cubature Method

This section introduces the Empirical Cubature Method (ECM) for approximating the integral of the reduced vector of internal forces. The goal is to approximate the integral of the reduced vector of internal forces. Beginning by summarizing the optimized cubature scheme proposed by An et al. [57]. Let $\boldsymbol{f}_{I}^{j}(\boldsymbol{x})=f_{I}\left(\boldsymbol{x}, \boldsymbol{\mu}^{j}\right)$ denote the $I$-th component ( $I=1,2, \ldots, n$ ) of the integrand at point $\boldsymbol{x} \in \Omega$ corresponding to the solution for input parameter $\boldsymbol{\mu}^{j}(j=1,2, \ldots, P)$.

The optimized cubature scheme approximates the integral of $\boldsymbol{f}_{I}^{j}$ as the sum of weighted evaluations at selected points:

$$
F_{I}^{j}=\int_{\Omega} \boldsymbol{f}_{I}^{j} d \Omega \approx \sum_{g=1}^{m} \omega_{g} \boldsymbol{f}_{I}^{j}\left(\overline{\boldsymbol{x}}_{g}\right)
$$

Here, $\mathcal{Z}=\left\{\overline{\boldsymbol{x}}_{g}\right\}_{g=1}^{m}$ represents the integration points, and $\boldsymbol{\omega}=\left[\omega_{1}, \omega_{2}, \ldots, \omega_{m}\right]^{T}$ are the positive weights. The positions of the integration points and the weights are determined by minimizing the integration error over all components and training samples:

$$
(\boldsymbol{\omega}, \mathcal{Z})=\arg \min _{\boldsymbol{w} \in \mathbb{R}_{+}^{m}, \overline{\mathcal{Z}}_{g} \in \Omega} \sqrt{\sum_{I=1}^{n} \sum_{j=1}^{P}\left(e_{I}^{j}\right)^{2}} \text {, where } e_{I}^{j}=\sum_{g=1}^{m} \omega_{g} f_{I}^{j}\left(\overline{\boldsymbol{x}}_{g}\right)-\int_{\Omega} f_{I}^{j} d \Omega \text {. }
$$

This minimization problem can be formulated in matrix format as:

$$
\begin{equation*}
(\boldsymbol{\omega}, \mathcal{Z})=\arg \min _{\boldsymbol{w} \in \mathbb{R}_{+}^{m}, \tilde{\mathcal{Z}}_{\boldsymbol{\mathcal { }}} \in \Omega}\left\|\boldsymbol{J}_{\mathcal{Z}} \boldsymbol{w}-\boldsymbol{b}\right\| \tag{4.10}
\end{equation*}
$$

where $\|\cdot\|$ refers to the standard Euclidean norm, and

$$
\boldsymbol{J}_{\mathcal{Z}}:=\left[\begin{array}{cccc}
\boldsymbol{f}^{1}\left(\overline{\boldsymbol{x}}_{1}\right) & \boldsymbol{f}^{1}\left(\overline{\boldsymbol{x}}_{2}\right) & \cdots & \boldsymbol{f}^{1}\left(\overline{\boldsymbol{x}}_{m}\right) \\
\boldsymbol{f}^{2}\left(\overline{\boldsymbol{x}}_{1}\right) & \boldsymbol{f}^{2}\left(\overline{\boldsymbol{x}}_{2}\right) & \cdots & \boldsymbol{f}^{2}\left(\overline{\boldsymbol{x}}_{m}\right) \\
\cdots & \cdots & \vdots & \cdots \\
\boldsymbol{f}^{P}\left(\overline{\boldsymbol{x}}_{1}\right) & \boldsymbol{f}^{P}\left(\overline{\boldsymbol{x}}_{2}\right) & \cdots & \boldsymbol{f}^{P}\left(\overline{\boldsymbol{x}}_{m}\right)
\end{array}\right], \quad \boldsymbol{b}:=\left[\begin{array}{c}
\int_{\Omega} \boldsymbol{f}^{1} d \Omega \\
\int_{\Omega} \boldsymbol{f}^{2} d \Omega \\
\vdots \\
\int_{\Omega} \boldsymbol{f}^{P} d \Omega
\end{array}\right], \quad \boldsymbol{f}^{j}=\left[\begin{array}{c}
f_{1}^{j} \\
f_{2}^{j} \\
\vdots \\
f_{n}^{j}
\end{array}\right]
$$

While the optimized cubature approach gives a great approximation, it can be computationally costly for problems with a large number of samples. To address this issue, the ECM introduces a dimensionality reduction process for the integrand. This reduction is similar to the displacement vector reduction in the initial stage. By employing a dimensionality reduction technique, a set of basis functions is determined for the integrand, enabling a more efficient representation. The minimization problem of Equation (4.10) is then formulated in terms of these basis functions, reducing the number of computations required.

In the continuous case, the basis functions $\Lambda_{i}$ for the integrand can be obtained using techniques like POD applied to the ensemble of snapshots. However, standard approaches may encounter problems when the integral of the snapshots is zero for all input parameters. To overcome this issue, the ECM adopts an expanded basis approach (EBA) [8] that uses a modified basis to avoid ill-posedness. By employing this expanded basis, the ECM addresses the challenges posed by quasi-static problems with no external forces.

Nevertheless, in common finite element implementations, the value of the integrand is only determined at the integration points of each finite element. Thus, for practical reasons, it is convenient to recapitulate the analysis outlined before from a "discrete" perspective, where the integral desiring to approximate can no longer be treated as a linear operator that maps continuous functions into $\mathbb{R}$. Instead, it should be viewed as an operator that maps vectors from $\mathbb{R}^{M}$ (where $M$ is the number of finite element ( FE ) integration points) to $\mathbb{R}$. The matrix representation of this operator can be derived from the expression for the approximated integral of $f_{I}$ as follows:

$$
F_{I}=\int_{\Omega} f_{I} d \Omega \approx \sum_{g=1}^{M} W_{g} f_{I}\left(\boldsymbol{x}_{g}\right)=\sum_{g=1}^{M} \sqrt{W_{g}}\left(\sqrt{W_{g}} f_{I}\left(\boldsymbol{x}_{g}\right)\right)=\sqrt{\boldsymbol{W}}^{T} \mathcal{F}_{I}, \quad I=1,2, \ldots, n
$$

Here, $\sqrt{\boldsymbol{W}} \in \mathbb{R}^{M}$ is defined as:

$$
\sqrt{\boldsymbol{W}}:=\left[\begin{array}{llll}
\sqrt{W_{1}} & \sqrt{W_{2}} & \cdots & \sqrt{W_{M}}
\end{array}\right]^{T}
$$

and $\mathcal{F}_{I} \in \mathbb{R}^{M}$ is a column vector that gathers the values of the integrand at all FE points, multiplied by the square root of each finite element integration weight:

$$
\mathcal{F}_{I}:=\left[\begin{array}{c}
\sqrt{W_{1}} f_{I}\left(\boldsymbol{x}_{1}\right) \\
\sqrt{W_{2}} f_{I}\left(\boldsymbol{x}_{2}\right) \\
\vdots \\
\sqrt{W_{M}} f_{I}\left(\boldsymbol{x}_{M}\right)
\end{array}\right]
$$

The matrix $\sqrt{\boldsymbol{W}^{T}}$ serves as the representation of the integral operator when the domain space $\mathbb{R}^{M}$ is equipped with the standard scalar product. The discrete formulation approximates the $L_{2}$ norm of a function $f_{I}$ over the domain $\Omega$. The function is assessed at a number of integration points, and the contributions are weighted and added to arrive at this approximation. This can be represented as:

$$
\left\|f_{I}\right\|_{L_{2}(\Omega)}^{2}=\int_{\Omega} f_{I} f_{I} d \Omega \approx \sum_{g=1}^{M} W_{g} f_{I}\left(\boldsymbol{x}_{g}\right) f_{I}\left(\boldsymbol{x}_{g}\right)=\left\|\mathcal{F}_{I}\right\|^{2} .
$$

Here, $\boldsymbol{x}_{g}$ represents the coordinates of the integration point $g$, $W_{g}$ denotes the corresponding FE weight, and $\mathcal{F}_{I}$ is the vector of function evaluations at the integration points. However, a limitation arises when FE integration rules with negative weights are present. To overcome this limitation, the absolute values of the FE integration weights can be used. By considering the absolute values, the discrete approximation can be modified as follows:

$$
F_{I} \approx \sum_{g=1}^{M}\left|W_{g}\right| f_{I}\left(\boldsymbol{x}_{g}\right)=\left\langle\operatorname{sgn}(\boldsymbol{W}), \mathcal{F}_{I}\right\rangle,
$$

In this modified formulation, $\operatorname{sgn}(\boldsymbol{W})$ represents a vector of the same dimension as $\boldsymbol{W}$ and contains the signs of the FE integration weights. The angle brackets denote the standard scalar product between vectors. By considering the absolute values of the FE integration weights, the discrete formulation can handle both positive and negative weights, allowing for a more flexible approximation when negative weights are present.

## Basis Matrices

The ECM replaces the integral operator with a reduced-order representation, leading to a further reduction in computational cost. Specifically, it seeks to find an approximation to the matrix $\sqrt[\boldsymbol{W}^{T}]{ }$ by constructing a reduced-order basis for the space spanned by $\sqrt{\boldsymbol{W}}$.

In order to solve the discrete version of the integral equation, the basis matrix for the nonlinear term $\mathcal{F}_{I}$ (where $I=1,2, \ldots, n$ ) needs to be determined. The basis matrix is constructed by combining a basis matrix for the range of $\sqrt{\boldsymbol{W}}{ }^{T}$ (which represents the space spanned by $\sqrt{\boldsymbol{W}}$ ) and a basis matrix $\boldsymbol{\Lambda}$ for the projection of $\mathcal{F}_{I}$ onto the kernel of $\sqrt{\boldsymbol{W}}^{T}$. The expanded basis matrix is given by:

$$
\text { Expanded basis matrix }=\left[\begin{array}{lllll}
\boldsymbol{\Lambda}_{1} & \boldsymbol{\Lambda}_{2} & \cdots & \boldsymbol{\Lambda}_{p} & \sqrt{\boldsymbol{W}}
\end{array}\right] .
$$

To compute $\boldsymbol{\Lambda}$, the ECM uses a snapshot-based approach with the snapshot matrix $\boldsymbol{X}_{\mathcal{F}}$, which contains the snapshots of $\mathcal{F}_{I}^{j}$ for all components $I$ and all training configurations $j$. The projection of each column of $\boldsymbol{X}_{\mathcal{F}}$ onto $\mathcal{N}\left(\sqrt{\boldsymbol{W}}^{T}\right)$ can be calculated by subtracting its orthogonal projection onto $\mathcal{R}\left(\sqrt{\boldsymbol{W}}^{T}\right)$. This yields:

$$
\hat{\mathcal{F}}_{I}^{j}=\mathcal{F}_{I}^{j}-\frac{\sqrt{\boldsymbol{W}}}{\|\sqrt{\boldsymbol{W}}\|}\left(\frac{\sqrt{\boldsymbol{W}}}{}{ }^{T} \mathcal{F}_{I}^{j}\right)=\left[\begin{array}{c}
\sqrt{W_{1}}\left(f_{I}^{j}\left(\boldsymbol{x}_{1}\right)-F_{I}^{j} / V\right)  \tag{4.11}\\
\sqrt{W_{2}}\left(f_{I}^{j}\left(\boldsymbol{x}_{2}\right)-F_{I}^{j} / V\right) \\
\vdots \\
\sqrt{W_{M}}\left(f_{I}^{j}\left(\boldsymbol{x}_{M}\right)-F_{I}^{j} / V\right)
\end{array}\right] .
$$

The entries of $\hat{\mathcal{F}}_{I}^{j}$ are essentially the same as in the continuous case, where each entry is the difference between the integrand at a given point and its volume average, multiplied by the square root of the corresponding finite element integration weight. Collecting all these vectors in the matrix $\hat{\boldsymbol{X}}_{\mathcal{F}} \in \mathbb{R}^{M \times n P}$ :

$$
\hat{\boldsymbol{X}}_{\mathcal{F}}=\left[\begin{array}{lllllllll}
\mathcal{F}_{1}^{1} & \ldots & \mathcal{F}_{n}^{1} & \mathcal{F}_{1}^{2} & \ldots & \mathcal{F}_{n}^{2} & \mathcal{F}_{1}^{P} & \ldots & \mathcal{F}_{n}^{P} \tag{4.12}
\end{array}\right]
$$

Next, a dimensionality reduction technique such as the Singular Value Decomposition (SVD) is applied to approximate $\hat{\boldsymbol{X}}_{\mathcal{F}}$ as

$$
\hat{\boldsymbol{X}}_{\mathcal{F}} \approx \boldsymbol{\Lambda} \boldsymbol{\Sigma}_{\Lambda} \boldsymbol{V}_{\Lambda}^{T}
$$

where $\boldsymbol{\Sigma}_{\Lambda}$ and $\boldsymbol{V}_{\Lambda}^{T}$ are the matrices of singular values and right singular vectors associated with the selected dominant left singular vectors $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times p}$.

To handle large snapshot matrices and minimize memory requirements, two strategies can be adopted. First, if the problem's geometry allows it, a reduced basis (RB) can be constructed to approximate the range of $\sqrt{\boldsymbol{W}}^{T}$, see [8]. This reduces the dimensionality of the problem and allows for more efficient computations. Second, if the snapshot matrix $\boldsymbol{X}_{\mathcal{F}}$ is too large to store in memory, the partitioned version of the SVD in [8], which precludes the necessity of manipulating the whole matrix, is used.

## Discrete Integral Equation

With the basis matrix $\boldsymbol{\Lambda}$ and the range space basis matrix $\sqrt{\boldsymbol{W}}$, the discrete integral equation can be constructed in the following form:

$$
\left[\begin{array}{lllll}
\boldsymbol{\Lambda}_{1} & \boldsymbol{\Lambda}_{2} & \cdots & \boldsymbol{\Lambda}_{p} & \sqrt{\boldsymbol{W}}
\end{array}\right]\left[\begin{array}{c}
\boldsymbol{C}_{1} \\
\boldsymbol{C}_{2} \\
\vdots \\
\boldsymbol{C}_{p} \\
\boldsymbol{D}
\end{array}\right]=\left[\boldsymbol{X}_{\mathcal{F}}\right]
$$

where $\boldsymbol{C}_{1}, \boldsymbol{C}_{2}, \ldots, \boldsymbol{C}_{p}$ are the unknown coefficients corresponding to the basis functions in the kernel of $\sqrt{\boldsymbol{W}}^{T}$, and $\boldsymbol{D}$ is the unknown coefficient vector corresponding to the basis functions in the range of $\sqrt{\boldsymbol{W}}$. This equation can be solved using various numerical techniques, such as least squares methods or iterative solvers. The solution provides the coefficients $\boldsymbol{C}_{1}, \boldsymbol{C}_{2}, \ldots, \boldsymbol{C}_{p}$ and $\boldsymbol{D}$, used to reconstruct the solution of the integral equation.

## Reconstructing the Solution

Once the coefficients $\boldsymbol{C}_{1}, \boldsymbol{C}_{2}, \ldots, \boldsymbol{C}_{p}$ and $\boldsymbol{D}$ are obtained, the solution of the integral equation can be reconstructed by evaluating the basis functions at the desired evaluation points. For a given evaluation point $\boldsymbol{x}_{\text {eval }}$, the reconstructed solution $\mathcal{F}_{I}\left(\boldsymbol{x}_{\text {eval }}\right)$ for component $I$ can be calculated as follows:

$$
\mathcal{F}_{I}\left(\boldsymbol{x}_{\text {eval }}\right)=\sum_{k=1}^{p} \boldsymbol{\Lambda}_{I, k} \cdot \boldsymbol{C}_{k}+\boldsymbol{W}_{I} \cdot \boldsymbol{D}
$$

where $\boldsymbol{\Lambda}_{I, k}$ denotes the $I$-th component of the $k$-th column of $\boldsymbol{\Lambda}$, and $\boldsymbol{W}_{I}$ denotes the $I$-th row of $\sqrt{\boldsymbol{W}}$ (e.g. $I$-th component could be displacements, strain, or stress fields). By evaluating this expression for each component and at the desired evaluation points, the solution of the integral equation can be reconstructed. The offline steps to determine the integration points and their associated weights are summarized in Algorithm 4. Likewise, the statement of the "hyper-reducer" order problem is set in Box III.

```
Algorithm 4 Second Reduction Stage
Require: Representative input parameters \(\left\{\boldsymbol{\mu}^{i}\right\}_{i=1}^{P}\)
```

Ensure: Integration points and weights for reduced-order problem
1: Solve the reduced-order problem (from Box II) for each $\boldsymbol{\mu}^{i}$
2: Store reduced internal forces in snapshot matrix $\boldsymbol{X}_{\mathcal{F}} \in \mathbb{R}^{M \times n P}$, and FE integration weights in vector $\boldsymbol{W} \in \mathbb{R}^{M}$
Compute the matrix of zero-integral snapshots $\hat{\boldsymbol{X}}_{\mathcal{F}}$ using Equation (4.11) applied to columns of $\boldsymbol{X}_{\mathcal{F}}$
: Determine orthogonal basis matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{M \times p}$ for the column space of $\hat{\boldsymbol{X}}_{\mathcal{F}}$ and the $p$ leading left singular vectors from SVD of $\hat{\boldsymbol{X}}_{\mathcal{F}}$
Construct matrices $\boldsymbol{J} \in \mathbb{R}^{p+1 \times M}$ and $\boldsymbol{b} \in \mathbb{R}^{p+1}$ for cubature optimization problem: $\boldsymbol{J}=\left[\begin{array}{ll}\boldsymbol{\Lambda} & \sqrt{\boldsymbol{W}}\end{array}\right]^{T}$ and $\boldsymbol{b}=\left[\begin{array}{ll}\mathbf{0}^{T} & V\end{array}\right]^{T}$, where $V=\sum_{i=1}^{M} W_{i}$
Determine integration points $\boldsymbol{z} \in \mathbb{N}^{m}$ and weights $\boldsymbol{\omega} \in \mathbb{R}_{+}^{m}$ using Appendix C

### 4.5. Summary

## Box I. Statement of the finite element problem

Given $\boldsymbol{\mu} \in \mathcal{D}, \boldsymbol{g}^{\boldsymbol{h}}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{N_{0}}, \boldsymbol{u}_{0}^{h}, \boldsymbol{\nu}_{0}^{h} \in \mathbb{R}^{N}, \boldsymbol{F}_{\text {ext }}^{h}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{N}$, find $\boldsymbol{d}^{h}:[0, T] \rightarrow \mathbb{R}^{N}$ such that

$$
\boldsymbol{M}^{h} \ddot{\boldsymbol{d}}^{h}+\boldsymbol{F}^{h}=\boldsymbol{F}_{\mathrm{ext}}^{h}-\boldsymbol{M}_{0}^{h} \ddot{\boldsymbol{d}}_{0}^{h}, \text { where } \boldsymbol{F}^{h}=\int_{\Omega} \boldsymbol{f}^{h} d \Omega \approx \sum_{g=1}^{M} W_{g} \boldsymbol{f}^{h}\left(\boldsymbol{x}_{\boldsymbol{g}},: \cdot\right) .
$$

subject to the Dirichlet boundary and initial conditions

$$
\boldsymbol{d}_{\mathbf{0}}^{\boldsymbol{h}}=\boldsymbol{g}^{h}, \boldsymbol{d}^{h}(0)=\boldsymbol{u}_{0}^{h}, \dot{\boldsymbol{d}}^{h}(0)=\boldsymbol{\nu}_{0}^{h}, \text { and the constitutive equations }
$$

$$
\left.\mathcal{H}\left(\boldsymbol{\sigma}, \boldsymbol{d}^{h}, \boldsymbol{d}_{0}^{h}, \boldsymbol{\xi} ; \boldsymbol{\mu}\right)\right|_{\boldsymbol{x}_{\boldsymbol{g}}}=\mathbf{0}, \quad g=1,2, \ldots, M .
$$

## Box II. Statement of the reduced-order problem

## Without approximation of internal forces

Given $\boldsymbol{\mu} \in \mathcal{D}, \boldsymbol{g}_{\boldsymbol{b}}^{\boldsymbol{h}}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{n_{0}}, \boldsymbol{u}_{\mathbf{0}}, \boldsymbol{\nu}_{\mathbf{0}} \in \mathbb{R}^{n},\left.\boldsymbol{F}_{\text {ext }}^{h}\right|_{e}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{n_{f}}$, find $\boldsymbol{d}:[0, T] \rightarrow \mathbb{R}^{n}$ such that

$$
\boldsymbol{M} \ddot{\boldsymbol{d}}+\boldsymbol{F}=\boldsymbol{F}_{\text {ext }}-\boldsymbol{M}_{0} \ddot{\boldsymbol{d}}_{0}, \text { where } \boldsymbol{F}=\int_{\Omega} \boldsymbol{f} d \Omega \approx \sum_{g=1}^{M} W_{g} \boldsymbol{f}\left(\boldsymbol{x}_{\boldsymbol{g}},: \cdot\right) .
$$

with $\left(\boldsymbol{f}=\boldsymbol{\Phi}^{T} \boldsymbol{f}^{h}\right)$ and $\boldsymbol{F}_{\boldsymbol{e x t}}=\left.\boldsymbol{R}_{\boldsymbol{e x t}} \boldsymbol{F}_{\boldsymbol{e x t}}^{\boldsymbol{h}}\right|_{e}$,
subject to the Dirichlet boundary and initial conditions $\boldsymbol{d}_{\mathbf{0}}=\boldsymbol{g}_{b}^{h}, \boldsymbol{d}(0)=\boldsymbol{u}_{0}, \dot{\boldsymbol{d}}(0)=\boldsymbol{\nu}_{0}$, and the constitutive equations

$$
\left.\mathcal{H}\left(\boldsymbol{\sigma}, \boldsymbol{d}, \boldsymbol{d}_{0}, \boldsymbol{\xi} ; \boldsymbol{\mu}\right)\right|_{\boldsymbol{x}_{g}}=\mathbf{0}, \quad g=1,2, \ldots, M .
$$

## Box III. Statement of the hyper-reduced order problem

## With approximation of internal forces

Given $\boldsymbol{\mu} \in \mathcal{D}, \boldsymbol{g}_{b}^{\boldsymbol{h}}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{n_{0}}, \boldsymbol{u}_{\mathbf{0}}, \boldsymbol{\nu}_{\mathbf{0}} \in \mathbb{R}^{n},\left.\boldsymbol{F}_{\text {ext }}^{h}\right|_{e}(\boldsymbol{\mu}):[0, T] \rightarrow \mathbb{R}^{n_{f}}$, find $\boldsymbol{d}:[0, T] \rightarrow \mathbb{R}^{n}$ such that
$\boldsymbol{M} \ddot{\boldsymbol{d}}+\boldsymbol{F}=\boldsymbol{F}_{\mathrm{ext}}-\boldsymbol{M}_{0} \ddot{\boldsymbol{d}}_{0}$, where $\boldsymbol{F}=\int_{\Omega} \boldsymbol{f} d \Omega \approx \sum_{g=1}^{M} W_{g} \boldsymbol{f}\left(\boldsymbol{x}_{\boldsymbol{g}},: \cdot\right)$.
with $\left(\boldsymbol{f}=\boldsymbol{\Phi}^{T} \boldsymbol{f}^{h}\right)$ and $\boldsymbol{F}_{\text {ext }}=\left.\boldsymbol{R}_{\text {ext }} \boldsymbol{F}_{\text {ext }}^{h}\right|_{e}$,
subject to the Dirichlet boundary and initial conditions
$\boldsymbol{d}_{\mathbf{0}}=\boldsymbol{g}_{b}^{h}, \boldsymbol{d}(0)=\boldsymbol{u}_{0}, \dot{\boldsymbol{d}}(0)=\boldsymbol{\nu}_{0}$, and the constitutive equations

$$
\left.\mathcal{H}\left(\boldsymbol{\sigma}, \boldsymbol{d}, \boldsymbol{d}_{0}, \boldsymbol{\xi} ; \boldsymbol{\mu}\right)\right|_{\boldsymbol{x}_{g}}=\mathbf{0}, \quad g=1,2, \ldots, m .
$$

## 5. Numerical Assessment

In this chapter, a comprehensive numerical assessment is conducted to evaluate the effectiveness of the proposed model-order reduction strategy. The assessment is performed on a representative structural example, specifically a quasistatic bending problem involving a stress-plane beam subjected to infinitesimal elastoplastic deformations. The beam, whose dimensions are of length $L=3 \mathrm{~m}$ and height $H=0.1 \mathrm{~m}$, is fixed at the left end and experiences a prescribed rotation of $\mu_{\max }=15^{\circ}$ at the right end. The parameter $\mu$ varies within the range of $\left[0,+\mu_{\max }\right]$, allowing for a thorough investigation of the beam's behavior.

For this analysis, a total of $P=200$ snapshots are generated to capture the beam's response under various loading conditions. The beam is discretized using quadratic elements, each equipped with 9 Gauss points to accurately represent the deformation characteristics. The finite element mesh can be seen in Figure 5.1 in GID.
$\stackrel{\ldots}{\longleftrightarrow}$
(a) Zoom out of the mesh of the stress-plane beam.

(b) Zoom in of the mesh of the stress-plane beam.

Figure 5.1.: Mesh characterization of the stress-plane beam in GID.
(Source: Own).

As observed from the above Figure 5.1 where the mesh is depicted, the size of the Elements is rectangular (i.e., not the same in the $x$-direction and the $y$-direction) and it is of size $X=$ 0.0375 m and size $Y=0.02 \mathrm{~m}$, providing an entire mesh with $N_{\text {elem }}=400$ elements. The numbering is also provided in Figure 5.1 (b).

Moreover, the material properties of the beam are defined by a single material with a Young's modulus of $E=70 \times 10^{3} \mathrm{MPa}$, a Poisson's ratio of $\nu=0.3$ and a material density $\rho=7850$ $\mathrm{kg} / \mathrm{m}^{3}$. These properties govern the beam's mechanical behavior, providing crucial input for the analysis.

In order to accurately analyze and solve a finite element problem, appropriate boundary conditions need to be applied to the geometry. As said earlier, in this case, two boundary conditions are applied: one fixed end and one prescribed rotation. In Figure 5.2, the boundary conditions of the finite element mesh are visually depicted, showcasing the fixed end on the left side (hence, the prescribed displacements) and the prescribed rotation on the right end.


Figure 5.2.: Boundary conditions applied to the geometry of the problem (Source: own).

By conducting this numerical assessment on the representative bending problem, the aim is to validate the effectiveness and accuracy of the proposed model-order reduction strategy in capturing the beam's elastoplastic response under different loading conditions. The obtained results will serve as a basis for evaluating the performance and applicability of the proposed approach in practical engineering applications.

Various parameters play a crucial role in accurately representing and analyzing the behavior of the beam. One of the fundamental parameters is the number of nodes, depicted as $N_{\text {node }}$, which determines the spatial discretization of the domain under consideration. Increasing the number of nodes leads to a finer mesh, enabling a more accurate representation of the system's geometry and capturing localized effects more effectively. In this case, the number of nodes defined is of $N_{\text {node }}=1771$.

Another important parameter is the number of Gauss points $M$ used for numerical integration within each element. Gauss points are strategically placed within an element to approximate the integrals of constitutive equations and evaluate element-level quantities. Increasing the number of Gauss points enhances the accuracy of the numerical integration, enabling a more precise representation of material behavior, such as stress and strain distributions. In this case, the number of Gauss points is 9 for each element, summing up a total of $M=9 N_{\text {elem }}=3600$. To address the problem of incompressibility while preserving the displacement-based formulation discussed earlier, researchers have developed a method known as the "B-bar" approach [61], applied in this particular case.

The goal of the hyper-reduced order model (from now on, HROM) is to predict both the displacement obtained in the application of the loads to the geometry of the problem for large strains and the second Piola-Kirchhoff stress (abbreviated as PK2stress), as a measure of the internal forces or stresses within a deforming solid body. Consider a deformation mapping from the reference configuration (denoted by 0 ) to the current configuration (denoted by $t$ ). The deformation is described by a displacement field, which maps points in the reference configuration to their corresponding locations in the current configuration. The displacement field is denoted by $\mathbf{u}(\mathbf{X}, t)$, where $\mathbf{X}$ represents a point in the reference configuration and $t$ denotes time.

Assume that there is a material point located at $\mathbf{X}$ in the reference configuration. When the material undergoes deformation, the new position of this point in the current configuration is given by

$$
(\mathbf{x}=\mathbf{X}+\mathbf{u}(\mathbf{X}, t) .
$$

The PK2 stress tensor, denoted by $\mathbf{P}$, is defined as follows:

$$
\mathbf{P}=\frac{\partial \mathbf{S}}{\partial \mathbf{F}}
$$

where $\mathbf{S}$ is the first Piola-Kirchhoff stress tensor and $\mathbf{F}$ is the deformation gradient tensor. The deformation gradient tensor, $\mathbf{F}$, describes the change in the geometry of the material during deformation. It is defined as the derivative of the displacement field with respect to the reference coordinates:

$$
\mathbf{F}=\frac{\partial \mathbf{x}}{\partial \mathbf{X}}=\mathbf{I}+\frac{\partial \mathbf{u}}{\partial \mathbf{X}},
$$

where $\mathbf{I}$ is the identity tensor. The first Piola-Kirchhoff stress tensor, $\mathbf{S}$, relates the forces acting on a surface element in the current configuration to the corresponding area element in the reference configuration. It is defined as:

$$
\mathbf{S}=\frac{\partial \Psi}{\partial \mathbf{F}}
$$

where $\Psi$ is the strain energy density function, which depends on the material behavior. Therefore, the PK2 stress tensor, P, can be obtained by differentiating the first Piola-Kirchhoff stress tensor, S, with respect to the deformation gradient tensor, $\mathbf{F}$.

In this particular numerical assessment, predicting the PK2 stress along with the displacement helps capture the mechanical response of the problem accurately, especially for large strains. The PK2 stress tensor will provide a measure of the internal forces within the material, taking into account the deformation and rotation of material elements.

After this brief explanation of the variables of the study, the hyper-reduced model used for the computational assessment of the numeric problem is addressed. As explained in the preceding sections, the construction of the desired HROM involves two sequential dimensionality reduction stages. The first reduction stage consists of the creation of a reduced-order model with no approximation of internal forces (henceforth labeled ROM). To arrive at the ROM, the steps depicted in the outlined Box II) are followed. The first step is to run FE analysis for representative values of such input parameters (the training inputs). Furthermore, as mentioned earlier, the time domain for each input history $(t \in[0, T]$, where $T=1 \mathrm{~s}$ ) is discretized into 200 equally spaced steps, $P=200$ snapshots.

Thus, the matrices are to be stored in memory and processed by dimensionality reduction in the first reduction stage: the matrix of displacements $\boldsymbol{X}_{d}$ and the matrix of PK2-stresses $\boldsymbol{X}_{P K 2 \text { stress }}$. Given that a large number of snapshots have been generated, holding uncertainty about the full dynamics of the unknown system, it is possible that the equilibrium state is reached with a reduced number of snapshots. To validate this, singular value decomposition (SVD) is performed to determine the number of necessary modes and store the found values in a more reduced and compact form. In this case, the rank obtained for both the displacement snapshot matrix and the PK2-stress snapshot matrix is 11 modes, as depicted in Figure 5.3. The region in between modes 4 and 8 has been enlarged and converted to a logarithmic scale for visualization purposes.


Figure 5.3.: (right) SVD truncation error versus number of basis vectors employed in the approximation $(n)$. (left) The portion between modes 4 and 8 is shown in magnified form in logarithmic scale (Source: own)

To ensure accurate results for both displacements and PK2 stress, an initial truncation error of $\epsilon^{s}=\epsilon^{\sigma}=10^{-3}$ is set. In a second reduction stage, the truncation error is reduced to $\epsilon^{d}=10^{-4}$ due to the desired stress accuracy. The number of displacement modes is also determined, revealing that out of a total of 11 possible modes, only 4 displacement modes are necessary to satisfy the stress accuracy requirement, as depicted in Figure 5.5.

Based on the selected displacement modes, the corresponding number of modes for PK1 and PK2 stresses turns out to be 14 and 12 , respectively. Consequently, the matrix of internal virtual work modes has a total of 96 columns, calculated by multiplying the number of displacement modes (4) and PK1 stress modes (14). Performing the singular value decomposition (SVD) on this matrix without any truncation yields 46 modes. Therefore, to achieve exact integration of these virtual work modes, 47 points $(46+1)$ are required.

Analyzing the curve of the relative SVD truncation error as a function of the number of modes displayed in Figure 5.4, it becomes evident that the error rapidly decreases as more modes are included. Further iterations lead to the discovery that a truncation error of $\epsilon^{f}=5 \times 10^{-5}$ is enough, corresponding to 18 integration points out of a total of 3600 . This choice satisfies both the displacement and stress accuracy requirements, as can be examined later by analyzing the contour plot of normal PK2 stresses in the $x$-direction shown in Figure 5.9.


Figure 5.4.: Truncation error versus number of modes (matrix of internal work snapshots) (Source: own)
$\stackrel{+}{\leftrightarrow}$
(a) Mode $1-S_{1}=20.88$

(c) Mode $3-S_{3}=0.015$

## $\ddagger$.

$\qquad$
(d) Mode $4-S_{4}=0.0056$

Figure 5.5.: Deformed shapes corresponding to the first 4 dominant displacement modes $\left[\boldsymbol{\Phi}_{\mathbf{1}}, \boldsymbol{\Phi}_{\mathbf{2}}, \boldsymbol{\Phi}_{\mathbf{3}}, \boldsymbol{\Phi}_{\mathbf{4}}\right]$ (Source: own).

When employing the element-based strategy for selecting integration points, it is found that the rank of the snapshot matrix is practically the same as in the points-based approach, resulting in a rank of 47 . This implies that $47+1=48$ elements would be necessary to achieve exact integration of the reduced internal virtual work. However, through the second reduction stage, it is determined that by setting the truncation error to $\epsilon^{f}=5 \times 10^{-5}$, the specified quality requirements are met, thus requiring only 18 finite elements.

Considering that each element employs the number of 9 Gauss points, this leads to a total of $18 \cdot 9=162$ integration points, which represents a $4.5 \%$ of the total number of Gauss points of the problem, but it is still large considering that the points-based approach only required 18 points for integration. Hence, it is seen that the element-based approach requires nine times more integration points than the point-based approach. The weights associated with the 18 elements in the element-based approach can be visualized in Figure 5.6 (a).


Figure 5.6.: Characterization of the ECM integration points (for $m=18$ ) (Source: Own).

The location and arrangement of the selected $m=18$ elements within the mesh can be observed in Figure 5.7. In this visualization, the chosen elements are highlighted as rectangular red boxes, enabling to easily identify their positions. The specific elements selected are labeled as $m_{\text {elem }}=[314,130,54,171,26,8,399,11,30,2,112,1,65,7,397,61,37,12]$.

Figure 5.7.: Location of the $m=18$ integration points chosen by the ECM algorithm (Source: own).

Next, it is required to study the extent to which the integration error affects the quality of the response predicted by the HROM (in terms of the output of interest). To this end, Figure 5.8 depicts the evolution of the variables of interest for the hyper-reduced case where $m=18$ integration points. As depicted in the figure, deviations between FE response and HROM graphs are practically imperceptible, which means that the accuracy obtained is quite considerable, even more so when the results of Table 5.1 are taken into account.



Figure 5.8.: Evolution over time of the variables of the 1st DOF. Results computed with the FE model and the ROM using $m=18$ and the full set of FE integration points (Source: own)

Using GID as a postprocessing software tool proves crucial in visually presenting the results within the mesh itself. In this case, the contour plot of the PK2 stress in Figure 5.9 is plotted in the mesh, which is presented as deformed in the final step of the displacements by a factor of 1 for the presence of large strains. The depicted Figure 5.9 shows the difference in the final PK2 stress results between computing a problem of $M=3600$ integration points (top) and $m=18$ integration points (bottom), proving astonishing results.


Figure 5.9.: Contour plot of normal 2nd Piola-Kirchhoff stress in the x-direction by the FE model (top) and the HROM (bottom) (Source: own).

### 5.1. Performance Analysis

In Table 5.1, a comprehensive summary of the dimensions and computation time of both the high-fidelity finite element ( FE ) model and the hyper-reduced order model (HROM) is presented, highlighting the remarkable compression achieved by the HROM. The table also includes the corresponding compression ratios, illustrating the significant reduction in the number of displacement unknowns and integration points.

The compression achieved is truly astonishing, as the number of displacement unknowns and integration points is reduced by more than one order of magnitude. These compression ratios are reflected in equally remarkable speedup factors, with the HROM completing the computation of 200 time steps in approximately 1 second, while the FE model took more than 20 seconds. These timings were obtained using the vectorized MATLAB program Appendix A running on a machine with a clock speed of $3.6 \mathrm{GHz}, 8 \mathrm{~GB}$ of RAM, and 8 Intel Core-i7 processors, operating on Windows.

|  | FEM | HROM | Compression ratio |
| :--- | :--- | :--- | :--- |
| Number of unrestricted DOFs | $N=3498$ | $n=4$ | 874.5 |
| Number of restricted DOFs | $N_{0}=44$ | $n_{0}=3$ | 14.67 |
| Number of integration points | $M=3600$ | $m=18$ | 200 |
| Computation time | 20.5952 s | 1.059 s | 19.45 |

Table 5.1.: Comparison of the dimensions and computation time of the FE problem and the HROM, along with the corresponding compression ratios.

The comparison presented in Table 5.1 highlights the substantial advantages of employing the HROM over the traditional FE approach. The drastic reduction in the number of degrees of freedom (DOFs) and integration points results in significant computational efficiency, as evidenced by the impressive speedup factor. This efficiency gain enables rapid analysis and exploration of different loading scenarios, facilitating quicker decision-making processes in engineering and design applications.

Furthermore, these findings not only underscore the effectiveness of the HROM in terms of computational efficiency but also emphasize its potential for accelerating the development and optimization of structural systems. The astonishing compression ratios and speedup factors achieved through the HROM demonstrate its viability as a valuable tool for efficient and accurate analysis of complex structures, leading to substantial time and cost savings in engineering projects.

## 6. Future research

In this chapter, potential avenues for future research are discussed, building upon the findings and methodologies presented in this thesis. The study thus far has focused on investigating data compression algorithms, such as singular value decomposition (SVD), proper orthogonal decomposition (POD), and Fourier and Wavelet transforms. Successful application of SVD and POD techniques to a finite element mesh, as demonstrated in the previous chapter, forms the basis for further exploration. However, several unexplored areas warrant attention for future investigations.

1. Exploration of Fourier and Wavelet Transforms: Fourier and Wavelet transforms offer alternative approaches for data compression in addition to SVD and POD. These transforms provide powerful tools for analyzing signals and extracting relevant information. Future studies can involve the application of Fourier and Wavelet transforms to finite element data and the evaluation of their effectiveness in capturing important features and reducing computational complexity.

Fourier transforms, based on the principle of expressing a function as a sum of sinusoidal components, provide a frequency-domain representation of the data. By decomposing the data into its frequency components, Fourier transforms reveal the underlying spectral characteristics. This representation is particularly useful for capturing periodic or oscillatory behavior in the data. By exploring the application of Fourier transforms to finite element data, researchers can assess their ability to compress and represent structural response data in the frequency domain. This can lead to insights into the dominant frequencies or modes of vibration within the system.

Wavelet transforms, on the other hand, offer a time-frequency representation that provides a more localized analysis compared to Fourier transforms. Wavelets are functions that can capture both frequency and temporal information simultaneously. By using wavelet transforms, researchers can decompose the data into different scales or resolutions, revealing both the frequency content and the temporal localization of features. This can be particularly beneficial when dealing with transient or nonstationary phenomena, where the behavior of the system changes over time. By applying wavelet transforms to finite element data, researchers can assess their ability to capture localized events, such as stress concentrations or dynamic response characteristics, and potentially achieve more efficient data compression.

By comparing the performance of these different data compression techniques, researchers can evaluate their ability to capture important features of the data while reducing computational complexity. It may also be possible to combine multiple techniques, such as SVD or POD, with Fourier or Wavelet transforms, to leverage their complementary strengths and achieve even more effective data compression and representation.
2. Investigation of Alternative Model Order Reduction Techniques: While the current research successfully applied the Empirical Cubature Method (ECM) to solve the finite element basis functions, there are other model order reduction techniques that warrant further exploration. Future studies can focus on comparing and evaluating alternative methods such as Reduced Basis Methods (RBM) or Dynamic Mode Decomposition (DMD) for solving the finite element basis functions.

Reduced Basis (RB) methods offer an attractive approach for model order reduction by constructing a low-dimensional subspace that captures the essential features of the system's response. These methods aim to identify a small number of basis functions, or modes, that effectively represent the behavior of the system. The basis functions are typically obtained by solving a reduced system using a small set of training samples or snapshots. By investigating the application of RB to finite element data, researchers can assess its accuracy in approximating high-fidelity solutions and its computational efficiency in reducing the dimensionality of the problem. This exploration can help identify scenarios where RB outperforms ECM or offers distinct advantages in terms of accuracy and efficiency.

Dynamic Mode Decomposition (DMD) is another promising model order reduction technique that focuses on capturing the dominant coherent structures or modes in a dynamical system. DMD identifies the spatial and temporal patterns in the data and decomposes them into a set of dynamically relevant modes. These modes can be used to reconstruct the system's response with reduced dimensionality. Future studies can explore the application of DMD to finite element data and assess its effectiveness in capturing the dominant structural behavior and reducing the computational complexity. By comparing the performance of DMD with ECM and other techniques, researchers can gain insights into the strengths and limitations of different model order reduction approaches.
3. Integration of Multiple Data Compression Techniques: Another promising direction for future research involves the integration of multiple data compression techniques to enhance the efficiency and accuracy of reduced-order models. Combining SVD or POD with Fourier or Wavelet transforms can potentially leverage the complementary strengths of each method, leading to more effective data compression and representation. By developing hybrid approaches that harness the advantages of different algorithms, researchers can advance the field by improving computational efficiency, model accuracy, and capturing complex dynamic behavior.
4. Application to Complex Engineering Systems: The current research is primarily focused on applying data compression algorithms and model order reduction techniques to a representative structural example. Future studies should extend the investigation to more complex engineering systems, such as multi-component systems, composite materials, or fluid-structure interactions. By applying the developed methodologies to these challenging scenarios, researchers can assess their generality, robustness, and potential limitations, thereby expanding the practical applicability of the techniques.

All in all, future research endeavors should encompass several key areas to advance the field of data compression and model order reduction in engineering analysis and design. Addressing these areas of research contributes to the advancement of efficient simulation and optimization of complex systems while maintaining a high level of accuracy and reliability.

## 7. Concluding remarks

The aim of this thesis was to navigate and provide a comprehensive investigation and evaluation of various compression algorithms and techniques, including singular value decomposition (SVD), proper orthogonal decomposition (POD), Fourier analysis, and Wavelet analysis, within the context of reduced order modeling (ROM). These techniques have been meticulously explored and successfully employed to achieve a significant reduction in dimensionality, paving the way for enhanced computational efficiency and accelerated simulations that ensure a great deal of accuracy.

The study has made substantial progress in understanding and describing the finite element model, which serves as the foundation for the implementation and evaluation of the reduced order model using, in this particular case, the empirical cubature method (ECM). By leveraging the power of these compression algorithms, the thesis has demonstrated the ability to extract essential information from high-dimensional systems while significantly reducing the computational complexity.

The utilization of a vectorized code provided by the professor (Appendix A), combined with the student's enhancements and expansions, has played a pivotal role in implementing the dimensionality reduction techniques and enabling efficient execution of the HROM computation. This code implementation has showcased the practical feasibility and efficacy of applying these compression algorithms to real-world engineering problems.

The numerical assessment carried out in this research has yielded astonishing results. The achieved compression factors and speedup have surpassed initial expectations, confirming the effectiveness of the implemented methodologies. The computational time required for simulations has been drastically reduced, enabling rapid analysis and decision-making processes that are useful when more complex cases are studied, such as the cases of the articles on which this method is based $[9,8,10]$.

While acknowledging the remarkable achievements presented in this thesis, it is important to recognize that further exploration and expansion of these methodologies are still possible. The field of compression algorithms offers vast opportunities for future research, including the investigation of more algorithms, hybrid approaches, and the assessment of complex dynamical systems to allow the method to shine. Despite this, through this research journey, the author has gained profound insights into the details of compression algorithms and their application to ROM, which are certain to be useful in the author's professional career.

In conclusion, the author takes great pride in the accomplishments achieved throughout this thesis and expresses deep satisfaction in the lessons learned. The transformative impact of mastering these compression algorithms and dimensionality reduction techniques is poised to shape the future of computational modeling and simulation, making similar research a significant contribution to the engineering community.

## 8. Environmental implications

In the modern world, the effects of human activity on the environment are a growing source of serious worry. Understanding and dealing with the environmental effects of diverse activities has never been more important given the multitude of environmental issues our world is facing, such as climate change, deforestation, and pollution, just to name a few. This thesis on compression techniques has the potential to significantly affect the environment in a number of ways.

One of the key ways in which this impact can occur is through the reduction of computational resources required for data analysis. Compression techniques aim to simplify complex datasets by reducing the number of variables while retaining important information. By effectively reducing the dimensionality of data, these techniques can significantly lower the computational power and energy needed for processing and analysis.

Furthermore, dimensionality reduction can also contribute to the development of more efficient algorithms and models. Traditional machine learning and data analysis methods often struggle with high-dimensional datasets, leading to increased computational requirements and energy consumption. By employing dimensionality reduction techniques, researchers can streamline the analysis process, resulting in more efficient algorithms that consume fewer resources.

Additionally, dimensionality reduction techniques can contribute to the development of sustainable solutions in various domains. For example, in environmental monitoring, where large amounts of data are collected from sensors and devices, dimensionality reduction can help identify critical features and patterns efficiently. This can lead to optimized resource allocation, improved decision-making, and reduced environmental impact by minimizing unnecessary data collection and processing.

Despite not being its primary objective, the side effects of this thesis can have a positive environmental impact. It is crucial for researchers and practitioners to consider the environmental implications of their work and strive for the implementation of eco-friendly practices throughout the lifecycle of their research.

## 9. References

[1] Multiphysics Cyclopedia. Detailed explanation of the finite element method (FEM). Available from URL: https://www.comsol.com/multiphysics/finite-element-method?parent=physics-pdes-numerical-042-62 (visited on 04/25/2023).
[2] Sebastian Götschel and Martin Weiser. Lossy Compression for Large Scale PDE Problems. eng. Tech. rep. 19-32. Takustr. 7, 14195 Berlin: ZIB, 2019. Available from URL: https://arxiv.org/pdf/1907.00667.pdf (visited on 03/05/2023).
[3] Sandeep Agrawal et al. "A Case for PARAM Shavak: Ready-to-Use and Affordable Supercomputing Solution". In: 2017 International Conference on High Performance Computing $\mathcal{E}^{3}$ Simulation (HPCS). 2017, pp. 396-401. Available from DOI: 10.1109/ HPCS. 2017.66.
[4] Steven L. Brunton and J. Nathan Kutz. Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control. Cambridge University Press, 2019. Available from DOI: 10.1017/9781108380690.
[5] K.N. Chiang and R.E. Fulton. "Concepts and implementation of parallel finite element analysis". In: Computers 8 Structures 36.6 (1990), pp. 1039-1046. Available from DOI: https://doi.org/10.1016/0045-7949(90)90211-J.
[6] Sebastian Götschel et al. "Reducing Memory Requirements in Scientific Computing and Optimal Control". In: Multiple Shooting and Time Domain Decomposition Methods. Ed. by Thomas Carraro et al. Cham: Springer International Publishing, 2015, pp. 263287. ISBN: 978-3-319-23321-5. Available from DOI: https://doi.org/10.1007/978-3-319-23321-5_10.
[7] Khalid Sayood. "1 - Introduction". In: Introduction to Data Compression (Third Edition). Ed. by Khalid Sayood. Third Edition. The Morgan Kaufmann Series in Multimedia Information and Systems. Burlington: Morgan Kaufmann, 2006, pp. 1-11. ISBN: 978-0-12-620862-7. Available from DOI: https://doi .org/10.1016/B978-012620862-7/50001-8.
[8] J.A. Hernández, M.A. Caicedo, and A. Ferrer. "Dimensional hyper-reduction of nonlinear finite element models via empirical cubature". In: Computer Methods in Applied Mechanics and Engineering 313 (2017), pp. 687-722. Available from DOI: https: //doi.org/10.1016/j.cma.2016.10.022.
[9] O Lloberas Valls. "Multiscale domain decomposition analysis of quasi-brittle materials". English. PhD thesis. Delft University of Technology, 2013. IsBN: 978-94-6191-875-8. Available from DOI: 10.4233/uuid:73541199-f32f-40bf-bfc6-8abcc5dd9108.
[10] J.A. Hernández. "A multiscale method for periodic structures using domain decomposition and ECM-hyperreduction". In: Computer Methods in Applied Mechanics and Engineering 368 (2020), p. 113192. ISSN: 0045-7825. Available from DOI: https: //doi.org/10.1016/j.cma.2020.113192.
[11] Dietmar Gross and Thomas Seelig. "Fracture Mechanics: With an Introduction to Micromechanics". In: Fraunhofer IWM (Jan. 2011). Available from DOI: 10.1007/ b22134.
[12] Marc Geers, Varvara Kouznetsova, and W.A.M. Brekelmans. "Multi-scale computational homogenization: Trends and challenges". In: Journal of Computational and Applied Mathematics 234 (Aug. 2010), pp. 2175-2182. Available from DOI: 10.1016/ j.cam.2009.08.077.
[13] V.G. Kouznetsova, M.G.D. Geers, and W.A.M. Brekelmans. "Multi-scale second-order computational homogenization of multi-phase materials: a nested finite element solution strategy". In: Computer Methods in Applied Mechanics and Engineering 193.48 (2004). Advances in Computational Plasticity, pp. 5525-5550. ISSN: 0045-7825. Available from DOI: https://doi.org/10.1016/j.cma.2003.12.073.
[14] J.A. Hernández et al. "High-performance model reduction techniques in computational multiscale homogenization". In: Computer Methods in Applied Mechanics and Engineering 276 (2014), pp. 149-189. Available from DOI: https://doi.org/10.1016/j. cma.2014.03.011.
[15] "FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre $\mathrm{SiC} / \mathrm{Ti}$ composite materials". In: Computer Methods in Applied Mechanics and Engineering 183.3 (2000), pp. 309-330. Available from DOI: https://doi.org/10.1016/ S0045-7825(99)00224-8.
[16] George J. Dvorak. "Transformation Field Analysis of Inelastic Composite Materials". In: Proceedings: Mathematical and Physical Sciences 437.1900 (1992), pp. 311-327. ISSN: 09628444. Available from URL: http://www.jstor.org/stable/52200 (visited on $04 / 12 / 2023$ ).
[17] C. Prud'homme et al. "Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods ". In: Journal of Fluids Engineering 124.1 (Nov. 2001), pp. 70-80. Available from DOI: 10.1115/1. 1448332.
[18] P. Krysl, S. Lall, and J. E. Marsden. "Dimensional model reduction in non-linear finite element dynamics of solids and structures". In: International Journal for Numerical Methods in Engineering 51.4 (2001), pp. 479-504. Available from DOI: https://doi org/10.1002/nme. 167.
[19] D. (David) Salomon. Data compression : the complete reference. eng. 4th ed. London: Springer, 2007. ISBN: 1846286026.
[20] C. C. Mac Duffee. "Matric equations". In: The Theory of Matrices. Ed. by C. C. Mac Duffee. Ergebnisse der Mathematik und Ihrer Grenzgebiete. Berlin, Heidelberg: Springer, 1933. ISBN: 978-3-642-99234-6. Available from DOI: 10.1007/978-3-642-99234-6_8. Available from URL: https://doi.org/10.1007/978-3-642-99234-6_8 (visited on $05 / 20 / 2023$ ).
[21] L. Autonne. "Sur les groupes linéaires, réels et orthogonaux". French. In: Bulletin de la Société Mathématique de France 30 (1902), pp. 121-134. Available from DOI: 10.24033/bsmf. 665.
[22] Alan Cline and Inderjit Dhillon. "Computation of the Singular Value Decomposition". In: Handbook of Linear Algebra. Chapman \& Hall/CRC, Dec. 2013, pp. 1027-1039. Available from DOI: 10.1201/9781420010572.
[23] Kari Karhunen. Zur Spektraltheorie stochastischer Prozesse. German. Suomalaisen Tiedeakatemian toimituksia. Helsinki: Suomalainen Tiedeakatemia, 1946, p. 7.
[24] R. Payen. "Fonctions aléatoires du second ordre à valeurs dans un espace de Hilbert". fr. In: Annales de l'institut Henri Poincaré. Section B. Calcul des probabilités et statistiques 3.4 (1967), pp. 323-396. Available from URL: http://www.numdam.org/ item/AIHPB_1967__3_4_323_0/ (visited on 04/09/2023).
[25] Edward N. Lorenz. Empirical orthogonal functions and statistical weather prediction. eng. Scientific report / Statistical Forecasting Project ; no. 1. Cambridge: Massachusetts Institute of Technology, Department of Meteorology, 1956.
[26] J.L. Lumley. Yaglom and V.I. Tatarski (eds). Nauka, Moscow: Atmospheric Turbulence and Wave Propagation, 1967, pp. 166-178.
[27] S. Ravindran. "Reduced-Order Adaptive Controllers for Fluid Flows Using POD". In: J. Sci. Comput. 15 (Dec. 2000), pp. 457-478. Available from DOI: 10.1023/A: 1011184714898.
[28] P Astrid. "Reduction of Process Simulation Models: A Proper Orthogonal Decomposition Approach". In: (Jan. 2004), pp. 21-30.
[29] Mark A. Pinsky. Introduction to Fourier analysis and wavelets / Mark A. Pinsky. eng. Graduate studies in mathematics ; v. 102. Providence, R.I: American Mathematical Society, 2009. ISBN: 9780821847978.
[30] N. Ahmed, T. Natarajan, and K.R. Rao. "Discrete Cosine Transform". In: IEEE Transactions on Computers C-23.1 (1974), pp. 90-93. Available from DOI: 10.1109/TC. 1974.223784.
[31] A.S. Lewis and G. Knowles. "Image compression using the 2-D wavelet transform". In: IEEE Transactions on Image Processing 1.2 (1992), pp. 244-250. Available from DOI: 10.1109/83.136601.
[32] S.G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 11.7 (1989), pp. 674-693. Available from DOI: 10.1109/34.192463.
[33] I. Daubechies. "The wavelet transform, time-frequency localization and signal analysis". In: IEEE Transactions on Information Theory 36.5 (1990), pp. 961-1005. Available from DOI: 10.1109/18.57199.
[34] Mani Mehra. "Wavelet-Galerkin Methods". In: Wavelets Theory and Its Applications: A First Course. Singapore: Springer Singapore, 2018, pp. 121-133. Available from DOI: https://doi.org/10.1007/978-981-13-2595-3_7.
[35] Štěpán Beneš and Jaroslav Kruis. "Singular Value Decomposition used for compression of results from the Finite Element Method". In: Advances in Engineering Software 117 (2018), pp. 8-17. Available from DOI: https://doi.org/10.1016/j.advengsoft. 2017.12.007.
[36] John Hopcroft. "Computer Science in the Information Age". In: Frontiers in Algorithmics. Ed. by Franco P. Preparata, Xiaodong Wu, and Jianping Yin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 2-2. ISBN: 978-3-540-69311-6.
[37] Gilbert Strang. "The Fundamental Theorem of Linear Algebra". In: The American Mathematical Monthly 100.9 (1993), pp. 848-855. Available from URL: http://www. jstor.org/stable/2324660 (visited on 03/28/2023).
[38] Lloyd N. Trefethen. Numerical linear algebra. English. Philadelphia: SIAM, 1997. ISBN: 9780898713619.
[39] G. Golub and W. Kahan. "Calculating the Singular Values and Pseudo-Inverse of a Matrix". In: Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis 2.2 (1965), pp. 205-224. Available from DOI: 10.1137/0702016.
[40] Michele Alessandrini et al. "Singular Value Decomposition in Embedded Systems Based on ARM Cortex-M Architecture". In: Electronics 10.1 (2021). Available from DOI: 10.3390/electronics10010034.
[41] L. Sirovich and M. Kirby. "Low-dimensional procedure for the characterization of human faces". In: J. Opt. Soc. Am. A 4.3 (May 1987), pp. 519-524. Available from DOI: 10.1364/JOSAA.4.000519.
[42] M. Kirby and L. Sirovich. "Application of the Karhunen-Loeve procedure for the characterization of human faces". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 12.1 (1990), pp. 103-108. Available from DOI: 10.1109/34.41390.
[43] Matthew Turk and Alex Pentland. "Eigenfaces for Recognition". In: Journal of Cognitive Neuroscience 3.1 (Jan. 1991), pp. 71-86. Available from DOI: 10.1162/jocn. 1991.3.1.71.
[44] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. "From few to many: illumination cone models for face recognition under variable lighting and pose". English. In: IEEE transactions on pattern analysis and machine intelligence 23.6 (2001), pp. 643660. Available from DOI: 10.1109/34.927464.
[45] Y.C. Liang et al. "Proper Orthogonal Decomposition and its applications - Part I: Theory". English. In: Journal of sound and vibration 252.3 (2002), pp. 527-544. Available from DOI: https://doi.org/10.1006/jsvi. 2001.4041.
[46] Vladimir Buljak. Inverse Analyses with Model Reduction: Proper Orthogonal Decomposition in Structural Mechanics. English. 1. Aufl. Computational Fluid and Solid Mechanics. Berlin, Heidelberg: Springer-Verlag, 2011. ISBN: 3642227023. Available from URL: https://link-springer-com.recursos.biblioteca.upc.edu/book/10.1007/978-3-642-22703-5 (visited on 04/10/2023).
[47] Stefan Volkwein. "Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling". In: Lecture Notes, University of Konstanz (Jan. 2012), pp. 1-11. Available from URL: https://www.researchgate. net/publication/268518240_Proper_ Orthogonal_Decomposition_Theory_and_Reduced-Order_Modelling (visited on 04/09/2023).
[48] Anindya Chatterjee. "An introduction to the proper orthogonal decomposition". In: Current Science 78.7 (2000), pp. 808-817. Available from URL: http://www.jstor. org/stable/24103957 (visited on 04/09/2023).
[49] The Analytical Theory of Heat. Trans. by Alexander Freeman. Cambridge Library Collection - Mathematics. Cambridge, 2009. ISBN: 978-1-108-00178-6. Available from DOI: 10.1017/CB09780511693205.
[50] S.G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 11.7 (1989), pp. 674-693. Available from DOI: 10.1109/34.192463.
[51] A. Haar. "Zur Theorie der orthogonalen Funktionensysteme. (Erste Mitteilung)." In: Mathematische Annalen 69 (1910), pp. 331-371. Available from DOI: https: //doi.org/10.1007/BF01456326.
[52] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. "Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations". In: International Journal for Numerical Methods in Engineering 86.2 (2011), pp. 155-181. Available from DOI: https://doi.org/10.1002/nme. 3050.
[53] D. Ryckelynck. "Hyper-reduction of mechanical models involving internal variables". In: International Journal for Numerical Methods in Engineering 77.1 (2009), pp. 75-89. Available from DOI: https://doi.org/10.1002/nme.2406.
[54] Maxime Barrault et al. "An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations". In: Comptes Rendus Mathematique 339.9 (2004), pp. 667-672. ISSN: 1631-073X. Available from DOI: https: //doi.org/10.1016/j.crma.2004.08.006.
[55] R. Everson and L. Sirovich. "Karhunen-Loève procedure for gappy data". In: J. Opt. Soc. Am. A 12.8 (Aug. 1995), pp. 1657-1664. Available from DOI: https://doi.org/ 10.1364/JOSAA. 12.001657.
[56] Grepl, Martin A. et al. "Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations". In: ESAIM: M2AN 41.3 (2007), pp. 575-605. Available from DOI: https://doi.org/10.1051/m2an:2007031.
[57] Steven S. An, Theodore Kim, and Doug L. James. "Optimizing Cubature for Efficient Integration of Subspace Deformations". In: ACM Trans. Graph. 27.5 (Dec. 2008). Available from DOI: https://doi.org/10.1145/1409060.1409118.
[58] Christoph von Tycowicz et al. "An Efficient Construction of Reduced Deformable Objects". In: ACM Trans. Graph. 32.6 (Nov. 2013). Available from DOI: https: //doi.org/10.1145/2508363.2508392.
[59] Annika Radermacher and Stefanie Reese. "POD-based model reduction with empirical interpolation applied to nonlinear elasticity". In: International Journal for Numerical Methods in Engineering 107.6 (2016), pp. 477-495. Available from DOI: https://doi. org/10.1002/nme. 5177.
[60] Saifon Chaturantabut and Danny C. Sorensen. "Discrete Empirical Interpolation for nonlinear model reduction". In: Proceedings of the $48 h$ IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference. 2009, pp. 4316-4321. Available from DOI: https://doi.org/10.1109/CDC. 2009.5400045.
[61] Juan C Simo and Thomas JR Hughes. Computational inelasticity. Vol. Interdisciplinary Applied Mathematics. Springer Science \& Business Media, 2006. Available from DOI: https://doi.org/10.1007/b98904.


[^0]:    ${ }^{1}$ A square matrix $\mathbf{X}$ is unitary if $\mathbf{U} \mathbf{U}^{*}=\mathbf{U}^{*} \mathbf{U}=\mathbf{I}$
    ${ }^{2}$ For real-valued matrices, this is the same as the regular transpose $\mathbf{X}^{*}=\mathbf{X}^{T}$

[^1]:    ${ }^{3}$ Hermitian matrices can be understood as the complex extension of real symmetric matrices.

[^2]:    ${ }^{4}$ The database from which the faces were extracted can also be obtained inside the author's data.

[^3]:    5 When $n \geq m$, the matrix $\boldsymbol{\Sigma}$ has at most $m$ nonzero elements on the diagonal and it is possible to exactly represent $\mathbf{x}$ using an economy-sized SVD version.

[^4]:    ${ }^{1}$ The superindex " $h$ " denotes FE nodal quantities, while bare symbols represent RO variables. Upper-case and lower-case symbols represent the dimensions of the FE and RO problems (e.g. $N$ and $n$, respectively).

