Using a Variational Autoencoder-based Strategy to enable Nonlinear Model Order Reduction in Computational Mechanics

MSc Thesis

Jasmijn van Riggelen

Delft University of Technology

TU Delft

Using a Variational Autoencoder-based Strategy to enable Nonlinear Model Order Reduction in Computational Mechanics

by

Jasmijn van Riggelen

Student Number

4864581

Chair:	Dr.ir. F.P. (Frans) van der Meer
Daily Supervisor:	Dr. I. (Iuri) B.C.M. Rocha
External Supervisor:	Dr. A. (Katerina) Varveri
Project Duration:	April 2024 – February 2025
Faculty:	Faculty of Civil Engineering, Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

ACKNOWLEDGMENTS

This thesis marks the end of my studies here in Delft, and I would like to give a word of thanks to all who helped me along. For this thesis specifically, I thank my committee for enabling me to work on this topic, which prompted me to learn about it from the moment I heard about it. Iuri, thank you for guiding me through, week after week, and for taking the time to explain concepts to me until it clicked. Frans and Katerina, thank you for critical questions and for kind words. Shout out to Joep Storm for showing me how to use DelftBlue.

Studying at TU Delft has been one of the main characteristics of my life in the last seven years. I'm grateful for everyone that I got to meet in that time and decided to stick with me. My BSc friend group, where we pushed each other to do every single assignment, and especially Wieke and Bells for staying close. Through U-BASE, my time as an MSc student was amazing: coffee breaks, activities from dodgeball to karaoke, trips to Belgium, Germany and South Korea, made from cool to great due to all the people involved. Of course, it wouldn't have been the same without being able to do a board year with the lovely Board 9: Gabriel, Luuk, Tessel, Rithu and Deborah.

Last but not least, I thank my family. Pap en mam, jullie staan altijd achter me en jullie hebben me zo veel gegeven. Mijn zussen, Floor, Roos en Margriet, bij jullie kan ik altijd terecht wanneer ik iets nodig heb en koffie op zondag is een toppunt van mijn week. Zonder jullie had ik dit niet kunnen doen.

Jasmijn van Riggelen
Delft, February 2025

ABSTRACT

High-fidelity models in computational mechanics, applied in structural engineering and material research, are developed to be accurate, detailed and robust. These models enable new ways of research, such as developing highly-tailored microstructures in materials. To bypass excessive computational times needed to execute these models, Reduced Order Models (ROMs) can be implemented, which reduce the dimensionality of a problem by projecting it onto a reduced-order latent space. While projection-based ROM usually makes use of Proper Orthogonal Decomposition (POD) to obtain a linear basis needed for Galerkin projections, this thesis describes the development of a Variational Autoencoder-based (VAE) approach, which allows for learning a nonlinear latent space. POD-based and a VAE-based techniques are developed and applied for a case study of a simply supported beam, loaded into plasticity by a single movable point load. The ROM results are compared to the Full Order Model (FOM) results. Results are analysed and compared between the POD- and VAE-based techniques, and for different (hyper)parameters within the VAE-ROM results. The results show the VAE-based ROM outperforms POD-counterparts, especially in extrapolation and with the condition that a large enough dataset was used for training the VAE. Of the analysed parameters used for training the VAEs, the influence of dataset size and latent space dimensionality is observed to be significant, while the degree of regularization in the VAE and the activation function are of limited influence. Furthermore, this study shows that the validation loss of a VAE during training is not a good indicator for the performance of the VAE-based ROM, and suggests to consider the VAE-ROM error already during training of the VAE.

CONTENTS

1

INTRODUCTION

1.1. BACKGROUND

The computational mechanics field is concerned with developing deterministic and probabilistic computational models that simulate the mechanical behaviour of materials and structures. High-fidelity models developed in this field aim to be accurate and robust, and are used in structural engineering and materials research. Not only conventional construction materials, like concrete and steel, are investigated, but also the behaviour of newer and more advanced materials is simulated: think of fiber reinforced and laminated materials. To better design with these materials, high-fidelity models are pivotal. One of the most common ways to model the behaviour of construction materials is to use the Finite Element Method (FEM), which can be used to calculate a numerical approximation of different material and structural behaviours.

High-fidelity models developed to simulate the intricate microstructure of a material quickly become complex, resulting in long run times and thus significant computational costs. This may be acceptable when running a model a few times, but when used for applications where repeated evaluations of the simulation model are needed, such as in optimisation, inverse problems and multiscale problems, these high computational requirements can make the use of the models unfeasible, even when applying relatively coarse discretizations.

In such cases, a Reduced Order Model (ROM) can be implemented: this technique reduces the dimensionality of the problem by transforming the full-order model into a reduced-order one, containing only a small number of features, with minimal loss of information. The computational effectiveness of the Reduced Order Models is achieved by the offline/online split of the work. The offline state, where the ROM is developed, is typically split into two parts. First, the high-fidelity model is solved for different parameter values. Second, these solutions, or snapshots, are used to construct a basis to provide projection into reduced, or latent, space. Moving then to the online stage, the governing equations, for example from the FEM discretization, are solved in the reduced space. In the online stage, the ROM is able to solve for new parameters values in less time than the Full Order Model (FOM) [1].

The dimensionality reduction is often executed through Proper Orthogonal Decomposition (POD), related to Principal Component Analysis (PCA), which identifies a set of orthogonal basis vectors within the high-dimensional dataset. These basis vectors are then used to project the governing equations of the full-order model to a reduced latent space - a process known as Galerkin projection [2, 3]. By solving the global equilibrium problem within this reduced latent space, the time needed to compute the solutions can be significantly reduced.

POD-based ROM is an efficient technique to reduce the computational cost of the model, but it has one major limitation. By definition, POD can only capture linear patterns that occur in the data into the modes that make up the columns of the basis matrix. While there are ways to mitigate this problem, once a model becomes highly nonlinear, the number of modes within the linear basis needed to capture this nonlinearity keeps increasing. This again makes the model larger and more complex, which can

have negative effects on robustness and computational cost.

In multiple fields, developments have been made using deep learning for dimensionality reduction of nonlinear high dimensional data (e.g., fluid dynamics [4–6]), and for building data-driven surrogate models to approximate the results as obtained by the high-fidelity models [7]. A specific type of neural network, the autoencoder, is of particular interest as it can reduce high dimensional data nonlinearly [8], because it maps the data to a low dimensional space and reconstructs the data from this reduced space back to its original input dimension through nonlinear layers.

1.2. RESEARCH OBJECTIVE AND SCOPE

The goal of this research is to overcome the limitations of applying POD based techniques to nonlinear problems by developing the use of nonlinear manifold learning methods. The study aims to either supplement or replace traditional POD with more advanced techniques that can capture nonlinear behaviour more effectively. In order to do so, a Variational Autoencoder (VAE) is employed, in the context of projection-based model order reduction. To aim and scope this research, the following research question is formulated:

Given the limitations of linear model order reduction techniques in nonlinear mechanical equilibrium analysis, would a Variational Autoencoder-based strategy be an improvement for a Proper Orthogonal Decomposition-based Reduced Order Model?

To further specify this objective, the following subquestions are conveyed:

- What are the limitations of a POD-based technique compared to techniques involving nonlinear reduction?
- To what extent is it beneficial to replace this POD-based technique with a Variational Autoencoder-based technique, and why?
- What is the relation between the VAE validation loss and the performance of the VAE-based ROM, and how do different (hyper)parameters influence this performance?

1.3. METHODOLOGY

Linear POD-based ROMs and nonlinear VAE-based ROMs are developed and compared for a case study. Snapshots are collected for the case of a simply supported beam, deforming under a single point load, placed on one of the discrete load locations. From the snapshots that the FOM of this case study provides, linear bases are extracted using Singular Value Decomposition (SVD) to enable POD-based ROM. Next to this, a displacement-dependent arc-length incremental-integration scheme is developed, which can be used for projection-based ROM and can incorporate the basis extraction from a VAE. VAEs are trained for different ranges of (hyper)parameters, after which their performance is compared through the VAE-ROM results.

1.4. THESIS OUTLINE

First, a theoretical background is given in chapter 2, going into Finite Element Modelling, Singular Value Decomposition and Variational Autoencoders. In chapter 3, the researched case study is discussed, as well as the development of the different models and techniques applied to it. The performance of the POD-based ROMs is shown in chapter 4, after which the VAE-ROM results are presented, compared and discussed in chapter 5. Finally, the conclusions and recommendations are compiled in chapter 6.

2

THEORETICAL FRAMEWORK

In this chapter, the main theoretical building blocks of the research done in this thesis are introduced, starting with the Finite Element Method. From this framework, the concept of model order reduction is build up, starting from more conceptual theory, to mathematical components and implementation. Lastly, the Variational Autoencoder is discussed.

2.1. THE FINITE ELEMENT METHOD

In this thesis, a Full Order Model (FOM) refers to a model where the Finite Element Method (FEM) is applied. FEM is a numerical method used to solve field problems, such as heat conduction or stress analysis, that are generally posed with partial differential equations (PDEs). The main components of this numerical method are outlined in this section, as based on explanations by Rocha [9] and Aragón and Duarte [10].

In FEM, a continuous domain Ω is subdivided into discrete elements (Ω_e), which are interconnected by nodes. The solution of the field variable u, which can represent for example displacements or temperatures, is approximated by a linear combination of polynomial shape functions:

$$u^h\Big|_{\Omega_e} = \sum_{i=1}^{n} N_i \bar{u}_i, \tag{2.1}$$

where $\bar{u}_i$ are the nodal values, N_i are the shape functions and n the number of nodes in the field. This discretized representation transforms the continuous problem into a finite-dimensional one.

2.1.1. Stress equilibrium

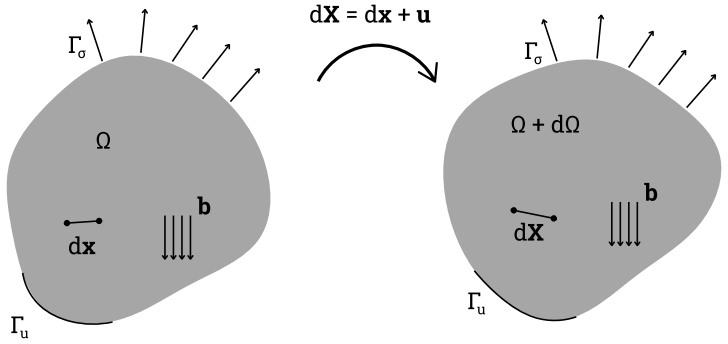
FEM can be applied to solve the equilibrium of stresses under applied load or deformations, as is its application in this thesis. A body strains under stress until equilibrium is reached, which is expressed by the following PDE:

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = \mathbf{0} \tag{2.2}$$

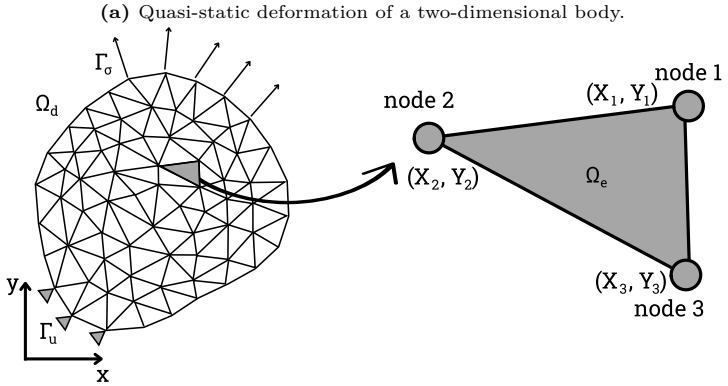
where $\boldsymbol{\sigma}$ is the stress tensor and $\boldsymbol{b}$ the body force vector. The relation between stress and strain in a material is defined by the constitutive law:

$$\boldsymbol{\sigma} = \mathcal{D}(\boldsymbol{\varepsilon}), \tag{2.3}$$

where $\mathcal{D}$ is the constitutive matrix and $\boldsymbol{\varepsilon}$ is the strain tensor. For small strains, the strain tensor is defined as:



(a) Quasi-static deformation of a two-dimensional body.



(b) Finite element partition of Ω_d into triangular elements, each Ω_e.

Figure 2.1: Domain Ω, showing deformation and discretization. Image adapted from [9] and [11].

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\right)^{\mathrm{T}}\right), \tag{2.4}$$

where $\mathbf{u}$ denotes the displacement field. To complete the problem, boundary conditions are defined on the surfaces Γ_u and Γ_σ such that:

$$\mathbf{u}|_{\Gamma_u} = \mathbf{u}^p \quad \boldsymbol{\sigma}\mathbf{n}|_{\Gamma_\sigma} = \mathbf{t}^p. \tag{2.5}$$

At surface Γ_u, Dirichlet (or essential) boundary conditions are applied with prescribed displacement $\mathbf{u}^p$ and at surface Γ_σ, Neumann (or natural) boundary conditions are imposed, in this case a stress $\boldsymbol{\sigma}$ normal to the surface, as denoted by $\mathbf{n}$, which is to be equal to the prescribed traction $\mathbf{t}^p$. The boundary conditions ensure the problem is fully defined.

2.1.2. Discrete weak form

The equilibrium as described by Eq. 2.2 is known as the strong form of the problem, which needs to be satisfied at every point in the body. For more complex domains or materials, it is challenging or even unfeasible to enforce this directly. Thus, the weak form is introduced, where the equilibrium is enforced in an integral or averaged sense across the domain.

By substituting the discretised solution u^h (Eq. 2.1) into the weak form and introducing matrix $\mathbf{B}$ containing the derivatives of the shape functions, the following system of equations for the global equilibrium is obtained:

$$\mathbf{f}^{int}(\bar{\mathbf{u}}) = \mathbf{f}^{ext} \tag{2.6}$$

with $\mathbf{f}^{int}$ and $\mathbf{f}^{ext}$ as the global internal and external force vectors, respectively, defined as:

$$\mathbf{f}^{int} = \mathop{\mathrm{A}}_{e=1}^{n_e} \left(\int_{\Omega_e} \mathbf{B}_{u,e}^T \boldsymbol{\sigma} \, \mathrm{d}\Omega \right), \quad \mathbf{f}^{ext} = \mathop{\mathrm{A}}_{e=1}^{n_e} \left(\int_{\Omega_e} \mathbf{N}_{u,e}^T \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma_{\sigma_e}} \mathbf{N}_{u,e}^T \mathbf{f}_p \, \mathrm{d}\Gamma \right) \tag{2.7}$$

for number of elements n_e in the mesh, with A as the standard finite element assembly operator, which ensures that components are added to the right position in the matrices to represent the corresponding global degree of freedom (DOF) locations. To solve Eq. 2.6, the tangent stiffness matrix $\mathbf{K}^{\Omega}$ is computed:

$$\mathbf{K}^{\Omega} = \frac{\partial \mathbf{f}^{\Omega}}{\partial \bar{\mathbf{u}}} = \mathop{\mathrm{A}}_{e=1}^{n_e} \left(\int_{\Omega_e} \mathbf{B}_{u,e}^T \mathbf{D} \mathbf{B}_{u,e} \, \mathrm{d}\Omega \right), \tag{2.8}$$

with $\mathbf{D}$ as the material's tangent constitutive matrix:

$$\mathbf{D} = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}} \tag{2.9}$$

2.1.3. Arc length controlled iteration scheme

In nonlinear problems, the global equilibrium problem of Eq. 2.6 can be solved with an incremental-iterative algorithm, based on the Newton-Raphson method. As it is to be solved iteratively, it is rewritten as a residual vector:

$$\boldsymbol{r}(\bar{\boldsymbol{u}}) = \boldsymbol{f}^{ext} - \boldsymbol{f}^{int}(\bar{\boldsymbol{u}}). \tag{2.10}$$

In linear problems the solution is always unique, which can also be the case for nonlinear problems. For these nonlinear problems, a displacement controlled iteration scheme can be used to obtain the unique solution: the scheme can largely be applied when an equivalent displacement control could be used in an experiment [12, 13]. Boundary conditions and imposed deformations of the Dirichlet variant are usually imposed by altering the stiffness matrix in the solver, effectively imposing these conditions on the system. However, for many nonlinear problems, the solution is not unique. If a solution is found, it may not be the solution that is sought. Using displacement control to solve these systems can also lead to divergence from the solver path. To keep to the path, arc length control can be used. As a starting point for this method, Eq. 2.10 can be put as

$$\boldsymbol{r}(\bar{\boldsymbol{u}}, \lambda) = \lambda \hat{\boldsymbol{f}} - \boldsymbol{f}^{int}(\bar{\boldsymbol{u}}), \tag{2.11}$$

where $\boldsymbol{r}(\boldsymbol{u}, \lambda)$ is the residual based on the displacement field $\bar{\boldsymbol{u}}$ and load factor λ, and $\hat{\boldsymbol{f}}$ is a unit force vector. This equation contains too many unknowns, so a constraint equation is introduced. First proposed by Riks [14], constraint equation g often takes a form similar to

$$g = \Delta \mathbf{u}^T \Delta \mathbf{u} + \Delta \lambda^2 \hat{\mathbf{f}}^T \hat{\mathbf{f}} - \Delta \ell^2, \tag{2.12}$$

where $\Delta \ell$ is a prescribed length that limits the radius of the arc. As g is a constraint equation, it should approximate to zero within a given tolerance. The linearized system of equations then becomes

$$\begin{bmatrix} \mathbf{K} & -\hat{\mathbf{f}} \\ \mathbf{h}^T & s \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ -g \end{bmatrix}, \tag{2.13}$$

where:

$$\boldsymbol{h} = \frac{\partial g}{\partial \boldsymbol{u}}, \quad s = \frac{\partial g}{\partial \lambda}. \tag{2.14}$$

Through decomposition of the displacement increment, the load factor update can be uncoupled as follows:

$$\Delta\lambda = \frac{g + \boldsymbol{h}^T \Delta \boldsymbol{u}^I}{s + \boldsymbol{h}^T \Delta \boldsymbol{u}^{II}} \tag{2.15}$$

The general algorithm can be found in Appendix B. For more details on arc length methods and different constraint equations, see [12, 13].

2.2. MODEL ORDER REDUCTION

2.2.1. Extrinsic vs. intrinsic dimensionality

At the core of reducing data dimensionality is the fact that many high-dimensional mathematical models of natural and engineered systems contain an intrinsically low-dimensional solution manifold [15]: in other words, high-dimensional data does not necessarily contain high-dimensional information. This distinction lies in the concepts of extrinsic and intrinsic dimensionality. The extrinsic dimension refers to the dimensionality of the space in which the data is embedded, while the intrinsic dimension reflects the lower-dimensional manifold where the data actually resides. Without this low intrinsic dimension of the data, it would not be possible to reduce the extrinsic dimension without significant loss of information [16]. The intrinsic solution-manifold dimensionality is at most the number of parameters in the system plus one, provided the full order solution has a unique solution of each parameter instance, as explained in Remark 2.1 of [5].

2.2.2. Galerkin projection

Galerkin projection, or the Galerkin Method, as it is often called outside of ROM applications, finds an approximation to field $u(x)$ through a linear combination of basis functions of the form

$$\bar{u}(x) = \phi_0(x) + \sum_{i=1}^{r} z_i \phi_i(x) = \phi_0(x) + z_1\phi_1(x) + \cdots + z_i\phi_i(x) + \cdots + z_r\phi_r(x). \tag{2.16}$$

The linearly-independent basis functions $\phi_i(x), i = 0, ..., r$ together form the basis matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times r}$, where the number of rows corresponds with the number of degrees of freedom (DOF) in the system (x). Non-homogeneous boundary conditions, if any, are satisfied by $\phi_0(x)$ to ensure that the remaining basis functions are homogeneous at the boundaries of the domain [17]. The basis functions are preselected and constructed from data using an optimization procedure, often a data-driven method. Connecting this to the concepts of extrinsic and intrinsic dimensionality, the basis, as constructed by these basis functions, projects the data from extrinsic dimension n to a reduced representation $\boldsymbol{z} \in \mathbb{R}^r$.

2.2.3. Proper Orthogonal Decomposition

One of the most widely used methods for constructing a basis for projecting the equilibrium problem is Proper Orthogonal Decomposition (POD). While POD can be used as an umbrella term covering Karhunen-Loève decomposition (KLD), Principal Component Analysis (PCA), and Singular Value Decomposition (SVD) [18], in this thesis, POD is used interchangeably with PCA performed with SVD.

POD works by extracting eigenvectors from the covariance matrix of the data, $D = \{\boldsymbol{x}_i\}_{i=1}^m$ with $\boldsymbol{x}_i \in \mathbb{R}^n$, which gives the optimal linear representation of this covariance matrix [1]. The largest $r < n$ eigenvectors are selected as basis functions to form the basis of the reduced (latent) space. Dimensionality reduction is then achieved by projecting the high-dimensional data onto the subspace spanned by the selected r eigenvectors, or modes, through Galerkin Projection. This projection represents a linear mapping from the full dimension space to the latent space, which means this technique is limited to capturing linear patterns in the data. Because of this linearity, using PCA to reduce data dimensionality is computationally economic, causing it to be widely applied [16]. There are various implementations of this technique. For example, POD can be applied at each time step of a simulation, providing a linear approximation of a nonlinear system. Another variant is Robust PCA, which separates the relevant data from sparse outliers, making it suitable for handling noisy or corrupted datasets [19, 20].

2.2.4. Singular Value Decomposition

SVD provides a numerical matrix decomposition that is guaranteed to exist and can be used as the underlying algorithm for PCA [21, 22]. This is advantageous, because there is no need to compute the eigenvectors of a covariance matrix, which can be computationally costly once the dataset gets larger. Let $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ be the snapshot matrix, of which the SVD decomposition is

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ & \vdots & \\ - & \mathbf{v}_m^T & - \end{bmatrix}, \tag{2.17}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices of size $(n \times n)$ and $(m \times m)$ respectively, with orthonormal columns: the left and right singular vectors. Matrix $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ has the so-called singular values on its diagonal, which are non-negative real numbers ordered from largest to smallest, and zeros off the diagonal. Larger singular values correspond to the directions in the data that capture more variance. The left singular vectors of $\boldsymbol{X}$, as found in $\boldsymbol{U}$, must be equal to the eigenvectors of the covariance matrix of $\boldsymbol{X}$, as demonstrated in [22].

2.2.5. Nonlinear model order reduction

Although POD is able to find the optimum linear representation of high-dimensional data, it is limited by its global linearity. Due to this, it cannot represent certain nonlinear physical simulations, and might be insufficient when dealing with complex real-world data. POD cannot distinguish between a lack of structure in the data and a nonlinear structure, which is an issue as data is often inherently nonlinear in domains where dimensionality reduction is useful [23–26].

Another limitation of POD is that it tries to capture the full dataset into global features: no matter the state or location that the solver in the ROM is concerned with, the basis will stay the same. For a simple structure, a fitting linear subspace can be found, but if the data has varying characteristics in different regions of the space, this cannot be extracted by POD [26]. An example of these limitations for local problems are prominent in [3], where a damage propagation problem is reduced with a modified POD. While the authors already state that a simple linear combination of the modes is not sufficient to accurately reproduce the solution for increasing damage, even with modifications, the computational costs have to go up from simple POD to get more accurate results.

Adding linear modes to the basis to improve the quality of reduction has a negative impact on the efficiency of the model, as each mode added means an additional value to solve for. A way to address this can be to linearize the subspace, which is a nonlinear manifold, with techniques such as Local Tangent Space Alignment (LTSA). LTSA finds local coordinates for the given data and aligns them with the manifold coordinates, realizing dimensionality reduction in this way. However, when applying the algorithm to a too small neighbourhood of local coordinates, it is unstable, and a too large neighbourhood distorts the local geometry of the original data. Furthermore, switching between the local elements can decrease robustness [16]. Switching from a linear subspace to a nonlinear manifold, which is able to capture nonlinear relationships within the data, is needed to improve representations of nonlinear structures and to facilitate consistency in methodologies [25]. There are nonlinear dimensionality reduction algorithms that are more efficient, such as kernel PCA, Laplacian eigenmaps and Isomaps, but reconstruction of data in its original dimensionality is not straightforward or sometimes not even possible for these techniques, limiting their applicability [24].

2.3. VARIATIONAL AUTOENCODER

In this thesis, a Variational Autoencoder (VAE) is used to find the nonlinear relationships within a dataset. An autoencoder is a type of Feedforward Neural Network (FNN) that maps sample $\mathbf{x}$ to the

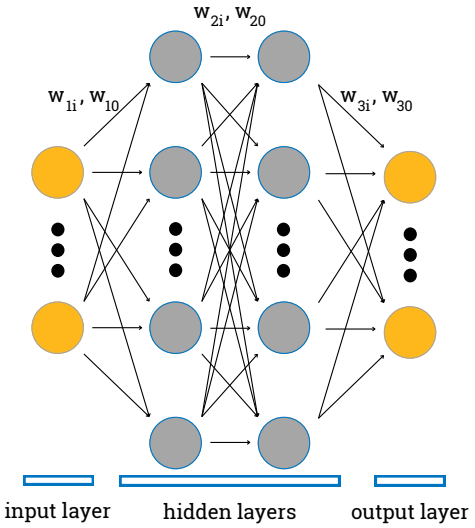


Figure 2.2: General structure of a Feedforward Neural Network.

latent version $\mathbf{z}$ of that sample and back to a reconstruction of that sample $\tilde{\mathbf{x}}$. Each latent variable of an autoencoder can capture more information than a POD basis function, meaning that for small numbers of modes, autoencoders can be used to reach a higher level of compression which cannot be matched by POD [1]. Next to that, Variational Autoencoders are able to provide reconstruction of the data in its original dimensionality. This section gives an overview of the key concepts needed to understand this technique, starting with Feedforward Neural Networks.

2.3.1. Feedforward Neural Networks

Neural networks (NNs) are machine learning models, the simplest of which is the Feedforward Neural Network (FNN), an example of which is shown in Fig. 2.2. FNNs use pattern recognition within data to make predictions for new cases, based on the information it has gained from the training data. In other words, it creates a map from the input data to the outputs, on which it can predict new 'routes' for new input data, as long as it is using the same map. Among many others, applications of NNs include image and speech recognition, autonomous vehicles and language translation [27].

A neural network generally consists of an input layer, hidden layers and an output layer. The layers consist of neurons, which depend on (at least) the neurons of the previous layer, or in case of the input layer, on each feature in the data. The state of a neuron in the current layer is constructed through linear combinations of the state of the neurons in the previous layer s_{l-1}:

$$a_l = \sum_{i}^{D} w_{li} s_{(l-1)i} + w_{l0}. \tag{2.18}$$

In these functional transformations, parameters w_{li} are known as the weights, and w_{l0} as the biases. The nonlinearity of the NNs can be obtained through activation functions $h(\cdot)$, which transform the quantities a_l, known as activations, into the state of the neurons in the current layer, s_l.

$$s_l = h(a_l) \tag{2.19}$$

The weights and biases as introduced in Eq. 2.18 are the parameters that are optimized during the training of the NN. At the start of the training sequence, they are initialized, after which the objective is to find the set of parameters that give the maximum likelihood of the training data. In order to do this, the training data makes forward and backward passes through the network. With the forward passes, the output for every data point in the dataset is obtained using Eq. 2.18 and 2.19, which is

then compared to the true target value using an error or loss function, such as the mean squared error (MSE):

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y(\boldsymbol{x}_i) - \hat{y}(\boldsymbol{x}_i))^2, \tag{2.20}$$

with $y(\boldsymbol{x}_i)$ as the true target value and $\hat{y}(\boldsymbol{x}_i)$ as the predicted output of the network. Every time the full training dataset is seen by the model, an epoch has passed. This can happen at once, or in minibatches to accelerate learning. During training, the error as computed by the loss function is minimised by adjusting the parameters connecting the layers, which is done using backpropagation. In backpropagation, the gradient of the error to the parameters is calculated, which can then be used to update these parameters accordingly with techniques like stochastic gradient descent (SGD). A more detailed derivation of these processes can be found in [28]. While the training loss gives information about how well the model has learned the training data, it does not provide information on how it will perform on unseen data. For this, usually a validation dataset is employed, which is not used to update the parameters during training, but instead gives a way to calibrate the hyperparameters of the model. Hyperparameters are all non-trainable parameters, like the activation function and the number of layers. Based on this validation loss, the best model during the full training sequence can be selected.

2.3.2. Autoencoders

Described above is how a Neural Network learns patterns in data between input and output values, known as supervised learning. However, when using an NN for dimensionality reduction, the inputs of the model are the same as its outputs, learning an autoassociative mapping. Such a model is called an autoencoder (AE). Input and output vectors have the same dimension n and the weights of the model are optimized to minimize the reconstruction error, typically calculated with the MSE loss function of Eq. 2.20. During training, every sample $\mathbf{x_i}$ is compressed to latent representation $\mathbf{z}_i$ of and back to a reconstruction of that sample $\tilde{\boldsymbol{x}}_i$, see Fig. 2.3. The latent representation dimension is smaller than the dimension of the input vector, meaning a perfect reconstruction of all input vectors is generally impossible. The part of the autoencoder mapping from the input to the latent is the encoder, the part mapping the latent variable back to the reconstructed output is the decoder.

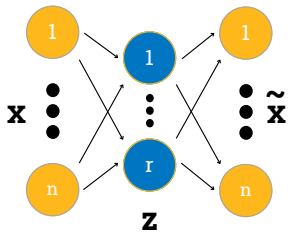


Figure 2.3: An autoencoder with two layers of weights. Without nonlinear activation functions, this network is still equivalent to linear POD analysis.

As many NNs, autoencoders typically have nonlinear activation functions for the hidden layers in the network. For a shallow network as depicted in Fig. 2.3, without nonlinear activation functions, the minimum error solution found would be equivalent to a projection onto the principal component subspace, meaning it will have the same result as when applying PCA to the dataset. Important to note is how, different from the principal components found by PCA, basis modes obtained through a VAE are not nested for reduction to different dimensions: reducing data from dimension n to $r2$, the first $r1$ vectors, with $r1 < r2$, are not the optimal solution for reducing from dimension n to $r1$, as would be the case in POD. With a VAE, a new autoencoder would have to be trained [22]. With nonlinear layers, a VAE can now essentially perform nonlinear POD. This advantage does come with drawbacks, as this is now a nonlinear optimization problem, making it more computationally intensive and bringing a risk of finding local minima for the loss function. For more information about autoencoders, the reader is referred to [28]. While the multilayer autoencoder can find a nonlinear mapping to latent dimension, the model has no probabilistic interpretation, no self-regularisation and no possibility of generating new data.

2.3.3. Stochastic behaviour and regularization

As an extension of a plain or conventional autoencoder, a Variational Autoencoder (VAE) does offer probabilistic interpretation, self-regularisation and the possibility to generate new data. While the generation of new data is not needed in this case, it is useful to have a form of self-regularisation within the machine learning model, as this helps prevent overfitting, which is especially relevant in cases with limited data available. How these advantages are obtained is explained here, based on the explanations given in [29] and [30].

A VAE still works with the real and latent variables $\mathbf{x}$ and $\mathbf{z}$, of which a pair forms the following distribution:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) \tag{2.21}$$

with

$$\begin{cases} p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I}) \\ p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|f_\theta(\mathbf{z}), \sigma^2\mathbf{I}) \end{cases} \tag{2.22}$$

where $\boldsymbol{\theta}$ refers to the parameters in the decoder. This can now be seen as a two-step generative model, where $p(\mathbf{z})$ (prior) is tied to the generation of a random latent variable $\mathbf{z}$ and $p(\mathbf{x}|\mathbf{z})$ is the likelihood of observing $\mathbf{x}$ given $\mathbf{z}$, which ties to the decoder. The function $f_\theta(\mathbf{z})$ is nonlinear, facilitating the nonlinear mapping. With Bayes' theorem, the posterior can be written as

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}, \tag{2.23}$$

which is not Gaussian. To be able to work with this structure, a Gaussian approximate posterior $q_\varphi(\mathbf{z})$ with mean $\mu_q(\mathbf{x})$ and diagonal covariance matrix with components $\sigma_q(\mathbf{x})$ is assumed, instead of calculating the true posterior $p(\mathbf{z}|\mathbf{x})$. With this, the stochastic process of the VAE can be represented as

$$\mathbf{x} \overset{q_\varphi}{\longmapsto} \mathbf{z} \overset{p_\theta}{\longmapsto} \tilde{\mathbf{x}}, \tag{2.24}$$

where q_φ represents the encoder and p_θ the decoder. Training the VAE is then done through maximizing the so-called Evidence Lower Bound (ELBO). This function is derived through maximizing the likelihood of the dataset, leading to the following:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_{q_\varphi}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \beta KL(q_\varphi||prior). \tag{2.25}$$

The ELBO, $\mathcal{L}$, depends on both $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$. The reconstruction term $\mathbb{E}_{q_\varphi}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})]$ increases when $\tilde{\boldsymbol{x}}$ is closer to the original $\boldsymbol{x}$. The regularization term $-KL(q_\varphi||prior)$ measures how similar the assumed approximate posterior $q_\varphi(\mathbf{z})$ is to the prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$, using Kullback-Leibler divergence (KLD). These two terms of Eq. 2.25 cannot both be maximized, which is how the regularization term helps prevent overfitting. This regularization term can be scaled by the KLD-weight β. Substantially, β indicates if outliers are seen as noise or as an integral part of the data that should be learned: a high value of β, thus a highly regularized model, discards outliers as noise.

One final part is needed to connect all the parts of the VAE for training. While the KL term can be computed exactly, the reconstruction term is approximated with a single sample from $q_\varphi(\boldsymbol{z})$, which is done indirectly through the reparametrization trick. This trick samples a variable $\boldsymbol{\epsilon}$ from the standard normal distribution and computes

$$\boldsymbol{z} = \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon}, \tag{2.26}$$

which ties the two halves of the model together, allowing it to be trained. This reparametrization trick also enables the trained model to produce new data, as a new datapoint can be generated from any sample from a standard normal distribution.

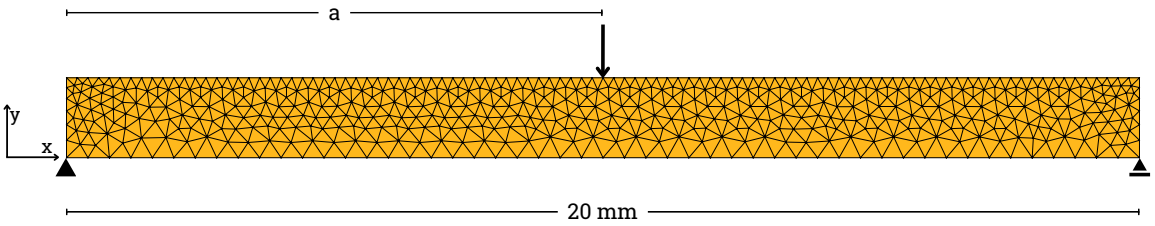
3

DEVELOPMENT

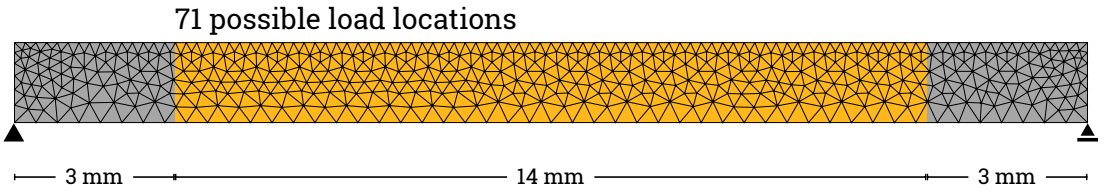
This chapter goes through the specifics of how the different parts needed for this research are developed. This starts with the case study and the dataset based on it. Then, it is explained how the linear bases are acquired and what it can look like. The different steps for nonlinear reduction are set out, as well as the modified arc length module used to run the FEM simulations.

3.1. CASE STUDY

To be able to compare different nonlinear model order reduction techniques, they need to be applied to a case study. The case study in this thesis is based on the case study used in [31], which is a simply supported 2D beam, loaded with a single point load in downwards direction at distance a from the left support, see Fig. 3.1a. The beam itself is solid and rectangular, made from a homogeneous, elastoplastic material.



(a) Simply supported beam with point load



(b) All possible load locations

Figure 3.1: Case study

The beam is discretized with a mesh and solved using the Finite Element Method, as described in section 2.1. For determining the step size, the number of steps, and the tolerance, an iterative process was needed to balance needing to reach nonlinearity for the load locations in the middle, while keeping convergence for the load cases near the edges of the beam. Where the original case study used a step size of -0.0055, a tolerance of 1e-10, and an unknown number of steps, this case uses 75 steps of size -0.04 mm, with a tolerance of 1e-6. Other parameters used in the case study are found in Table 3.1. The full length of the beam is 20 mm and will be loaded between $a = 3$ mm and $a = 17$ mm to avoid

excessive stress concentrations at the supports. This corresponds to 71 possible load locations, as also illustrated in Fig. 3.1b.

Table 3.1: Parameters used in the case study.

Parameters
J2 material
Plane stress
E = 2500 MPa
$\nu = 0.37$
yield function = $64.80 - 33.6 \cdot \exp(\frac{\kappa}{-0.003407}) - 10.21 \cdot \exp(\frac{\kappa}{-0.06493})$
Triangular elements with three nodes
Gauss integration of polynomial order 1

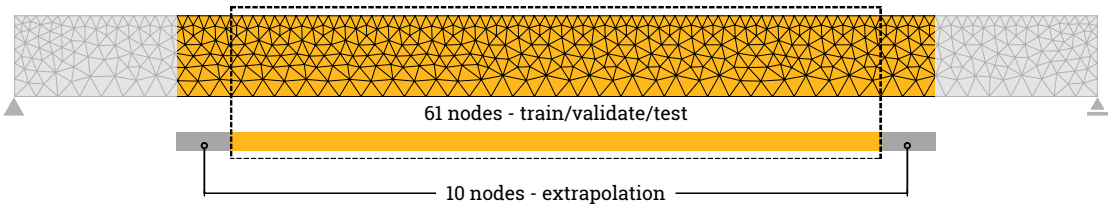
3.2. DATASET

Loading 71 load locations with 75 steps leads to 5325 full field solutions that serve as datapoints. However, as different models need to be set up, trained and compared, it is crucial to create separate datasets for separate purposes. How these datasets are constructed is explained in this section, and visualized in Fig. 3.2.

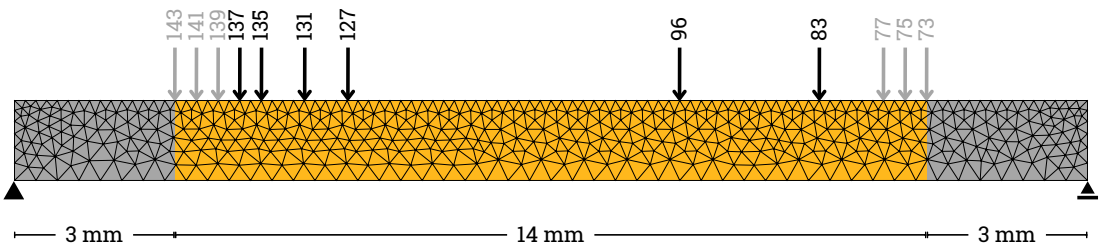
In order to see how well different models perform on cases outside the range of data they have been trained on, ten load locations (five on each side) are detached from the aforementioned 71 load locations. From these ten locations, six will be used as 'robust test locations', for which the ROMs will have to extrapolate to provide results. These locations are indicated by the grey arrows in Fig. 3.2b.

After these points are removed from the overall pool of snapshots, there are now 61 load locations left for training, validation and testing. For the 'regular' testing, six load locations are randomly selected, and their corresponding snapshots are again stored in a separate file, containing a dataset with 450 data points. For one of these selections, the test locations are displayed as the black arrows in Fig. 3.2b.

The remaining 55 load locations can be used for training and validation, for which an 80/20 split is used. A shuffle is applied over the locations, but not over the full dataset, making again sure that data points are grouped per location. In order to provide interpretable data for constructing both the linear and nonlinear bases, the data is not normalized.



(a) Regions on the beam to draw the datasets from.



(b) Example of the test set resulting from a random draw over the 61 nodes (random seed 23, black) and an even distribution from the 10 extrapolation nodes (grey).

Figure 3.2: Visualizing the different dataset regions from the FOM solution snapshots.

3.3. LINEAR MODEL ORDER REDUCTION

As introduced in section 2.2.2, basis $\boldsymbol{\Phi}$ is used to perform Galerkin projection. As discussed in section 2.2.4, matrix $\boldsymbol{U}$ is equivalent to the eigenvectors of the covariance matrix of $\boldsymbol{X}$, sorted in how well the column captures the information. This is quantified by the singular values in the $\boldsymbol{\Sigma}$ matrix, where a higher number indicates a better summary of the data. After applying SVD to the sampled snapshots, it still needs to be decided how many modes the basis should consist of, thus how many columns of $\boldsymbol{U}$ are selected for the basis. As the columns are sorted on contribution, the first columns add the most information to the basis and their contribution will go down as more are added. The solutions for optimal linear reduction to different latent dimension sizes are nested, such that SVD only needs to be applied once to a dataset, after which bases of different sizes can be constructed from the same $\boldsymbol{U}$ matrix. In Fig. 3.3 it is seen what the first two modes of the linear basis are, obtained for a dataset of 55 sampled locations. Furthermore, Fig. 3.4 shows the information gained by adding more modes to the basis, for the same dataset. To illustrate the impact of a VAE being able to capture more information per latent dimension, bases of one and two modes only are used.

(a) Mode 1 **(b)** Mode 2

Figure 3.3: The two modes making up the 2D POD basis. The 1D POD basis is simply only mode 1.

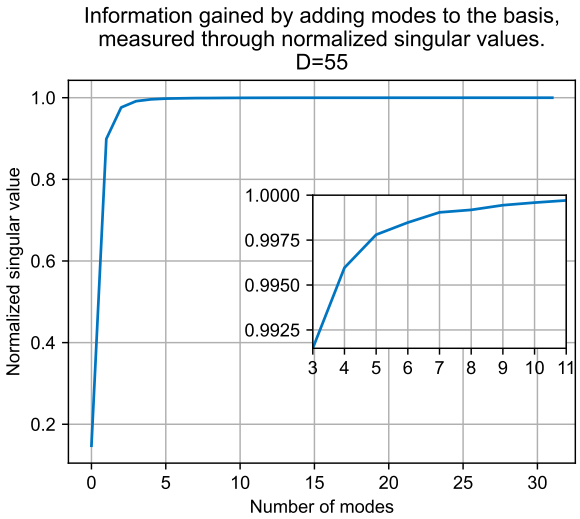


Figure 3.4: The singular values in the $\boldsymbol{\Sigma}$ matrix indicate how much information is captured per mode. For this dataset, over 98% of the information is captured with only two modes.

3.4. NONLINEAR MODEL ORDER REDUCTION

The nonlinear bases are constructed from the same datasets as the linear bases. This is done in two steps. First, the VAE is trained for a latent dimension of $Z = 1$ or $Z = 2$. Different VAEs are trained, based on the hyperparameter ranges, as discussed in this section. All VAE models are trained for 20000 epochs on the DelftBlue supercomputer. From a trained VAE, a gradient is extracted for a given state, from which the basis can be constructed, as is more elaborated on in 3.4.2.

3.4.1. Architecture

When setting up a VAE, like any neural network, there are many choices to be made about its non-trainable parameters: the hyperparameters. The choices of these hyperparameters influence the performance of the VAE and the ROM that depends on it, and thus they need to be made carefully. Machine learning trackers can help with this, such as Weights and Biases [32], which was used for this research. Through 'sweeps' the user can specify a range of hyperparameters to train the VAE for. A sweep can be over multiple hyperparameter ranges at once and their performance is stored in an online workspace, where the correlations between the hyperparameter values and the validation loss (or any other metric the user would like to be tracked) are calculated. This gives clear insights into the influence of different hyperparameters on the performance of the model.

As the relationship between the performance of the autoencoder and the accuracy of the dependent reduced FEM simulation is not known and most likely not straightforward, part of the hyperparameters are not set, but instead used in a range to investigate their influence on the VAE-ROM results. These hyperparameters are the KLD-weight (used for regularization), the latent space dimension and the activation function. The dataset size is not a hyperparameter, as it does not influence how well the model is able to learn itself. However, the dataset size does influence the overall performance of the model and is thus not categorized separately. The ranges of these values can be found in table 3.2, with the initialisms that are used to indicate these parameters. From these parameter ranges, 180 different configurations are made to train 180 different VAEs. Next to these variable parameters, the general architecture is set, as well as the batchsize and learning rate. These values were chosen based on the analysis as performed with the Weights and Biases tracker and can also be found in table 3.2. A visualisation of the architecture of the VAEs is depicted in Fig. 3.5. Each VAE is trained for 20000 epochs.

Table 3.2: (Hyper)parameters for the different Variational Autoencoders

Variable			Fixed	
Dataset size	D	$[10, 20, 30, 40, 50, 55]$	Hidden layers	2
Latent dimension	Z	$[1, 2]$	Hidden layer dimension	64
KLD-weight	K	$[0, 1e^{-8}, 1e^{-6}, 1e^{-4}, 1e^{-2}]$	Learning rate	$1e^{-3}$
Activation function	A	ReLU, ELU, Sigmoid	Minibatch size	64

3.4.2. Obtaining the basis

For a linear basis, SVD is applied to the selected snapshots, after which the first r of columns of matrix $\boldsymbol{U}$ are selected to obtain a basis that will reduce to a latent space with dimension r. For a nonlinear basis, more steps are involved, which are also found in lines 3-11 of Algorithm 1. In the solver, for every time step the state $\boldsymbol{u}$ is propagated forward through the trained VAE, obtaining $\boldsymbol{\mu}$ and the reconstruction of the state $\tilde{\boldsymbol{u}}$. Note that $\boldsymbol{\mu}$ denotes the mean of the distribution from which latent variable $\boldsymbol{z}$ is sampled. The decoder of the VAE then maps from latent variable $\boldsymbol{z}$ to reconstruction of the state $\tilde{\boldsymbol{u}}$. To link all this to the basis $\boldsymbol{\Phi} \in \mathbb{R}^{n \times r}$, the following holds for the linear case:

$$\tilde{\boldsymbol{x}} = \boldsymbol{\Phi} \cdot \boldsymbol{z} \rightarrow \boldsymbol{\Phi} = \frac{\partial \tilde{\boldsymbol{x}}}{\partial \boldsymbol{z}}. \tag{3.1}$$

Even in a nonlinear case, the second part of Eq. 3.1 still holds, but one important alteration is made. Latent variable $\boldsymbol{z}$ is sampled from the distribution $\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, which adds a layer of stochastisity to

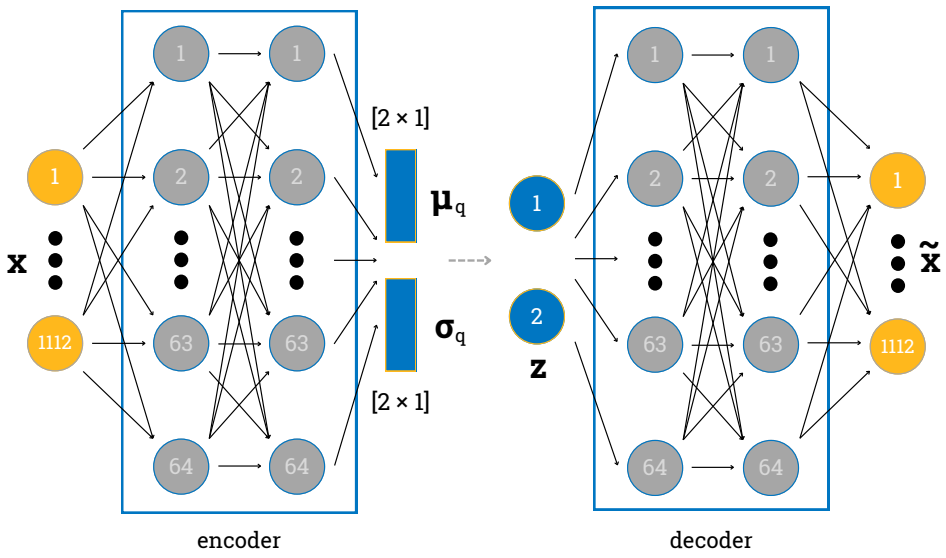


Figure 3.5: The architecture of the Variational Autoencoder, which is trained for different hyperparameter ranges, as specified in table 3.2. For clarity of the image, it is here depicted with a latent space dimension of 2, this could also be 1.

the model which is useful during training, but adds noise when extracting a basis from it. Therefore, to acquire the basis, backpropagation is done to $\boldsymbol{\mu}$:

$$gradient = \frac{\partial \tilde{\boldsymbol{x}}}{\partial \boldsymbol{\mu}}. \tag{3.2}$$

This gradient is not yet the final basis. During development of the VAE-ROM algorithm, two more steps were found to be necessary to get a useful basis. First, to secure a starting position of the solver at that time step, the state for which the gradient is obtained is added as a first column to the gradient. Secondly, SVD is applied on this combined matrix to guarantee a basis in which the modes are orthogonal to each other. From this decomposition, $\boldsymbol{U}$ is then taken as the basis $\boldsymbol{\Phi}$.

Since the gradient used to construct $\boldsymbol{\Phi}$ is based on the current state of the solver, the basis is able to evolve with the model, which is another advantage of using this technique over a POD-based one. When at the start of a time step the basis is updated, the current state of the solver is the result of the previous step, and thus still based on the basis of the previous step. To keep the translation between the full order and reduced order state consistent within a step, the state is reduced and then projected back into full space, both with the just updated basis. The time step $n = 0$ state is a zero-filled state, which does not provide the VAE with any useful information, as all trained locations start with this, and it will not be able to construct a valid basis from this state. Therefore, for the first time step, the basis used is a linear basis $\boldsymbol{\Phi}_l$, constructed from only the time step $n = 1$ snapshots of the training dataset.

3.5. MODIFIED ARC LENGTH MODULE

When looking at a reduced order FEM solver, the stiffness matrix is one of the solver components that gets projected into latent space, resulting in a reduced stiffness matrix. Where before the stiffness matrix $\boldsymbol{K}$ (Eq. 2.8) had a size of $[n \times n]$, where n are the number of DOF in the system, the reduced matrix has a size of $[r \times r]$, where $r < n$, and enforcing the Dirichlet conditions becomes difficult. Thus, even though displacement control could be used in terms of convergence, there are practical limitations to using a displacement controlled iteration scheme when working with a ROM.

An arc length enforced displacement controlled scheme gives a solution here. Instead of enforcing the imposed deformations through altering the stiffness matrix $\boldsymbol{K}$, they are enforced by the constraint equation g, based on the current full order state, instead of on the reduced solver components, as introduced in section 2.1.3. An added benefit of working with a ROM, is that homogeneous Dirichlet

boundary conditions do not need to be reimposed on the solver, since these are learned from the provided FOM snapshots. The constraint equation can be simplified such that it only enforces displacements at the node where the load is placed. The simplified constraint equation looks as follows:

$$g = \sum_{i=1}^{N^p} \Delta u_i^p - \Delta u_i^s. \tag{3.3}$$

The increase in displacement at node i of the current FEM model state, is indicated with Δu_i^s, which should be equal to the prescribed displacement at that same node, Δu_i^p, for all nodes with prescribed displacements N^p. The algorithm for the (reduced) FEM solver is implemented using the pyJive finite element library in Python [33] and is found in Algorithm 1. While the actual code can run FOM as well ROM, the algorithm shows implementation for ROM only. The subscript $_r$ indicates the reduced version of a matrix or vector. For example, stiffness matrix $\boldsymbol{K}$:

$$\boldsymbol{K}_r = \boldsymbol{\Phi}^T \boldsymbol{K} \boldsymbol{\Phi}. \tag{3.4}$$

For the state, or current displacement field, the notation as in line 11 of Algorithm 1 is used. With this, the standard arc length algorithm as found in Appendix B is largely executed in reduced space, with communication between full order and reduced space every time the state is updated. Having said that, two additional alterations are made. The first concerns $\boldsymbol{h}$ (Eq. 2.14): to be able to apply this derivative in reduced space, the following is done:

$$\mathbf{h}^T = \frac{\partial g}{\partial \boldsymbol{\alpha}} = \frac{\partial g}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \boldsymbol{\alpha}} = \frac{\partial g}{\partial \mathbf{u}} \boldsymbol{\Phi} \tag{3.5}$$

Secondly, because Eq. 3.3 does not depend on load factor λ, s from Eq. 2.14 is equal to zero, meaning it is not found in the load factor updates in lines 18 and 28 of Algorithm 1.

Algorithm 1: DispArclenModule - ROM

Require *Nonlinear relation* $\mathbf{f}_{int}(\mathbf{u})$ *with* $\mathbf{K}(\mathbf{u}) = \dfrac{\partial \mathbf{f}_{int}}{\partial \mathbf{u}}$, *constraint equation* g *and* $\dfrac{\partial g}{\partial \mathbf{u}}$, $\boldsymbol{\Phi}$ *for POD-based or* $\boldsymbol{\Phi}_l$ *for VAE-based.*

Initialize $n = 0$, $\mathbf{u}^0 = \mathbf{0}$, $\lambda = 0$, $\hat{f}$

while $n <$ *number of time steps* **do**

  if *VAE and* $n = 0$ **then**

    Use for the first time step the linear basis: $\boldsymbol{\Phi} = \boldsymbol{\Phi}_l$

  if *VAE and* $n > 0$ **then**

    Pass state $\mathbf{u}$ through VAE to obtain $\tilde{\mathbf{u}}$ and $\boldsymbol{\mu}$

    Obtain gradient: $\dfrac{\partial \tilde{\mathbf{u}}}{\partial \boldsymbol{\mu}}$

    Combine state and gradient and apply SVD: $\mathbf{U}, \mathbf{S}, \mathbf{V}^T = \left[\mathbf{u}, \dfrac{\partial \tilde{\mathbf{u}}}{\partial \boldsymbol{\mu}}\right]$

    Update basis: $\boldsymbol{\Phi} = \mathbf{U}$

    Update latent and real state accordingly: $\boldsymbol{\alpha} = \boldsymbol{\Phi}^T \mathbf{u}$ and $\mathbf{u} = \boldsymbol{\Phi}\boldsymbol{\alpha}$

  Assemble $\mathbf{K}$ and $\mathbf{f}_{\text{int}}$

  Reduce $\mathbf{f}_{\text{int}}$, $\hat{f}$, and $\mathbf{K}$

  $\mathbf{f}_{\text{ext0\_r}} = \hat{\mathbf{f}}_r \lambda$

  Calculate reduced residual: $\mathbf{r}_{\text{r}} = \mathbf{f}_{\text{ext0\_r}} - \mathbf{f}_{\text{int\_r}}$

  Solve $\mathbf{K}_{\text{r}} \Delta\boldsymbol{\alpha}^{\text{I}} = r_{\text{r}}$ and $\mathbf{K}_{\text{r}} \Delta\boldsymbol{\alpha}^{\text{II}} = \hat{f}_{\text{r}}$

  Get g and $\mathbf{h}^T = \dfrac{\partial g}{\partial \mathbf{u}} \dfrac{\partial \mathbf{u}}{\partial \boldsymbol{\alpha}} = \dfrac{\partial g}{\partial \mathbf{u}} \boldsymbol{\Phi}$

  Solve $\Delta\lambda = \dfrac{g + \boldsymbol{h}^T \Delta\boldsymbol{\alpha}^I}{\boldsymbol{h}^T \Delta\boldsymbol{\alpha}^{II}}$

  $\Delta\boldsymbol{\alpha} = \Delta\lambda \Delta\boldsymbol{\alpha}^{II}$

  Update reduced state: $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}$

  Project back to real space: $\mathbf{u} = \boldsymbol{\Phi}\boldsymbol{\alpha}$

  repeat

    Reduce $\mathbf{f}_{\text{int}}$ and $\mathbf{K}$

    Construct $\boldsymbol{f}_{ext\_r} = \boldsymbol{f}_{ext0\_r} + \Delta\lambda \hat{\boldsymbol{f}}_r$

    Calculate reduced residual: $\boldsymbol{r}_{\text{r}} = \boldsymbol{f}_{\text{ext0\_r}} - \boldsymbol{f}_{\text{int\_r}}$

    Solve $\mathbf{K}_{\text{r}} d\boldsymbol{\alpha}^{\text{I}} = \boldsymbol{r}_{\text{r}}$ and $\mathbf{K}_{\text{r}} \Delta\boldsymbol{\alpha}^{\text{II}} = \hat{\boldsymbol{f}}_{\text{r}}$

    Get g

    Solve $d\lambda = \dfrac{g + \boldsymbol{h}^T d\boldsymbol{\alpha}^I}{\boldsymbol{h}^T d\boldsymbol{\alpha}^{II}}$

    $d\boldsymbol{\alpha} = d\boldsymbol{\alpha}^I + d\lambda d\boldsymbol{\alpha}^{II}$

    Update reduced state: $\boldsymbol{\alpha} = \boldsymbol{\alpha} + d\boldsymbol{\alpha}$

    Project back to real space: $\mathbf{u} = \boldsymbol{\Phi}\boldsymbol{\alpha}$

    $\Delta\lambda = \Delta\lambda + d\lambda$

  until *relative error* $>$ *tolerance or* $|g| >$ *tolerance*;

  Save $\lambda = \lambda + \Delta\lambda$ for this timestep

  Commit and check commit

4

PERFORMANCE OF POD-BASED REDUCED ORDER MODELS

Before the results of the VAE-based ROMs are analysed, the performance of the POD-based ROMs is visualized in this chapter. The chapter starts by looking at the importance of providing relevant information for the basis. The influence of adding information to the basis through the number of datapoints is shown, and the general performance of the different POD-ROMs is analysed.

4.1. POD-ROM TRAINED ON A SINGLE LOAD LOCATION

To illustrate the consequence of the information contained in the basis used for the reduced model, Fig. 4.1 shows the mean squared error (MSE), calculated as

$$MSE = \frac{1}{N_{DOF}} \sum_{n=1}^{N_{DOF}} (u_{i,FOM} - u_{i,ROM})^2, \tag{4.1}$$

for the last time step states of $Z = 1$ and $Z = 2$ POD-ROMs, trained on the middle load location: number 108. Here, N_{DOF} are the number of degrees of freedom in displacement field $\boldsymbol{u}$. The original FOM contains $N_{DOF} = 1112$, which is thus reduced here to only one or two DOFs. As a consequence of this reduction, the FEM simulations for this case run 1.37 and 1.18 times faster when reduced to $Z = 1$ and $Z = 2$, respectively. When looking at the performance of the POD-ROMs for a load case close to the middle of the beam, the MSE is really small. When the POD-ROM needs to provide results closely related to cases it is familiar with, it performs really well, seen in Fig. 4.2: there is very little difference between the FOM and ROM results.

However, the error grows drastically when the load location moves away from the information that the reduced model has. Fig. 4.3 shows the result for location 138, far away from the middle of the beam, performing poorly: the shape of the ROM displacement field mirrors that of the beam loaded in the middle, which is the only shape the basis of the reduced model learned. This mismatch in shape is seen as well in the corresponding load-displacement curve, where a higher load is required for the same displacement at the location of the load for the ROM in comparison to the FOM.

The comparison between these cases shows the two sides of the metaphorical reduction-coin: if the reduction basis contains enough relevant information, it is able to solve the FEM simulation in reduced space very well, but if the information is too specified, it cannot do the same for new cases. For these new cases, the basis is misinformed.

4.2. LEARNING CURVE

In Fig. 3.3 it was shown how information is retained as displacement shapes in the modes of the basis, of which the relevance is quantified by the singular values found in the $\boldsymbol{\Sigma}$ matrix, plotted cumulatively

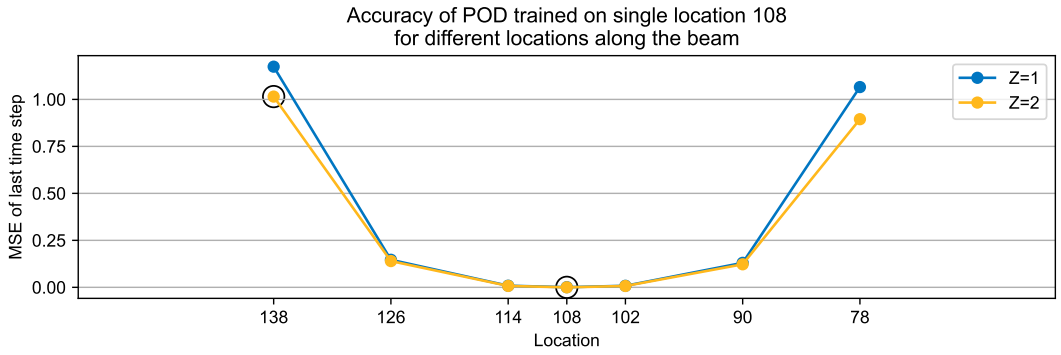
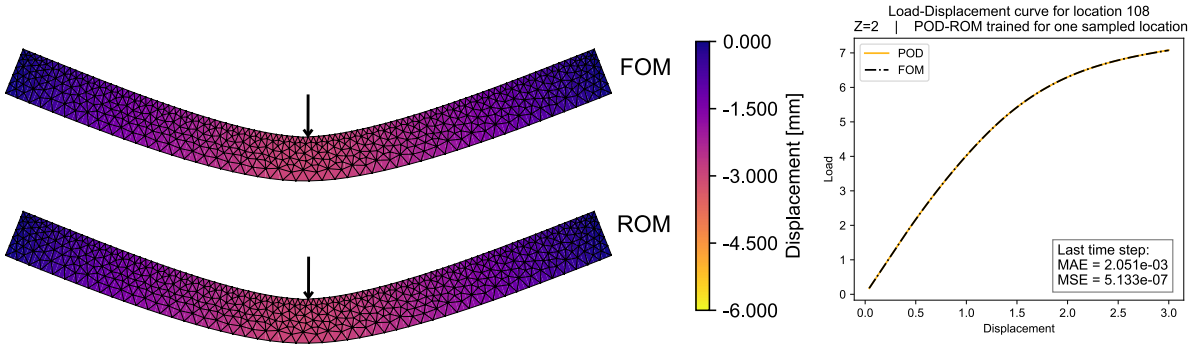


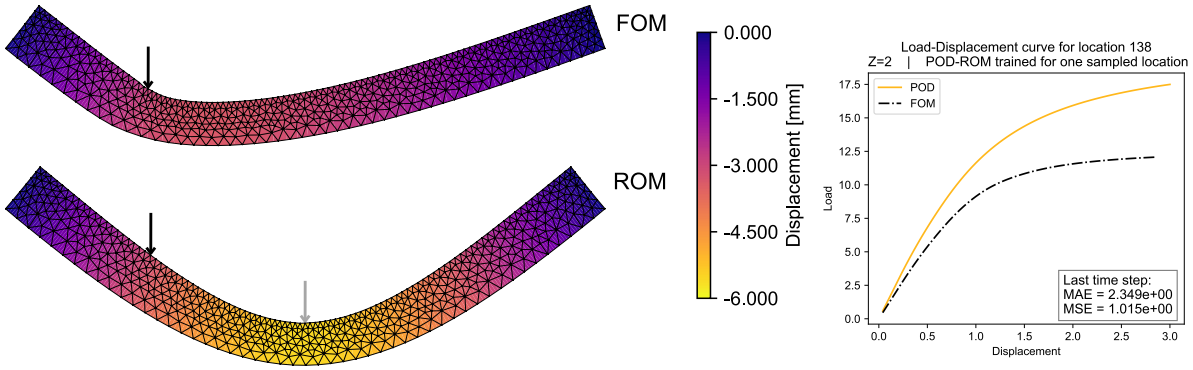
Figure 4.1: MSE of the last time step of the POD-ROM trained solely on location 108. The indicated cases are shown in Fig. 4.2 and 4.3 below.



(a) ROM predicting the same load case the basis was constructed from, compared to FOM

(b) Load-Displacement graph

Figure 4.2: Effect of fitted basis.



(a) ROM predicting for a different load case than it was trained for, compared to FOM. The load used for training is indicated with the light grey arrow.

(b) Load-Displacement graph

Figure 4.3: Effect of misinformed basis.

in Fig. 3.4. For POD-ROMs working with a reduced latent dimension of $Z = 1$ and $Z = 2$, the learning curve when increasing the size of the training dataset is presented in Fig. 4.4. Six validation locations are chosen, equally distributed over the sampling region, of which the MSE of the last time step state between the ROM and FOM results is averaged. This is done for 20 different sampling draws, from which the mean is calculated, and the standard deviation over these 20 draws. As expected, adding more data to the basis gives better performing ROMs with a smaller variation in results, but it is notable how the benefit of adding more sampled locations stagnates from a dataset of $D = 20$, which is equivalent to 1500 datapoints.

Clearly visible in this figure, is also the advantage of adding more modes to the basis, as the jump in performance between $Z = 1$ and $Z = 2$ is significant. The reach of this advantage is shown in Fig. 4.5, where for the largest possible dataset, $D = 55$ with 4125 datapoints, more modes keep being added. Adding modes, or columns, to the linear basis stops being beneficial from $Z = 12$, although this stagnation already starts at $Z = 6$, since the step from six to eight modes does not notably improve the result. These observations are in line with the singular values found in the singular value decomposition, as shown in Fig. 3.4, where with six modes the information retained is already above 99.75%.

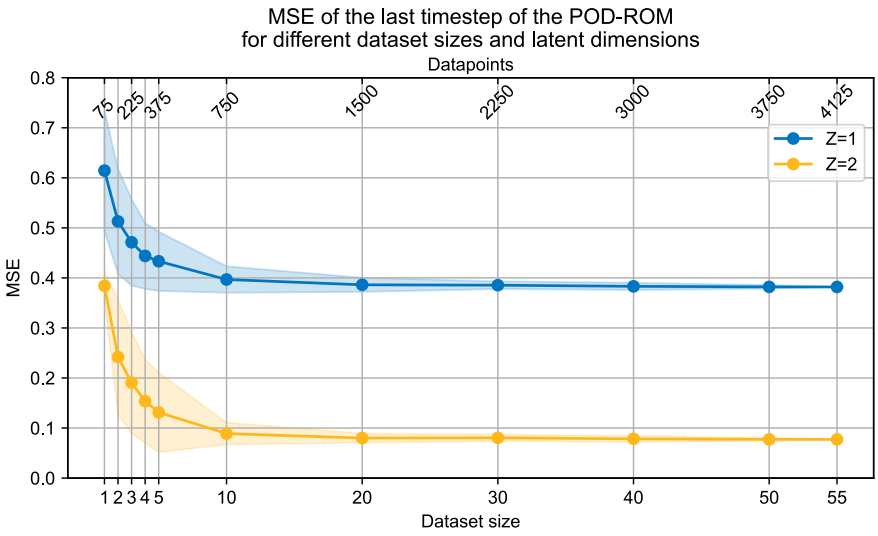


Figure 4.4: Learning curve for POD-based ROMs.

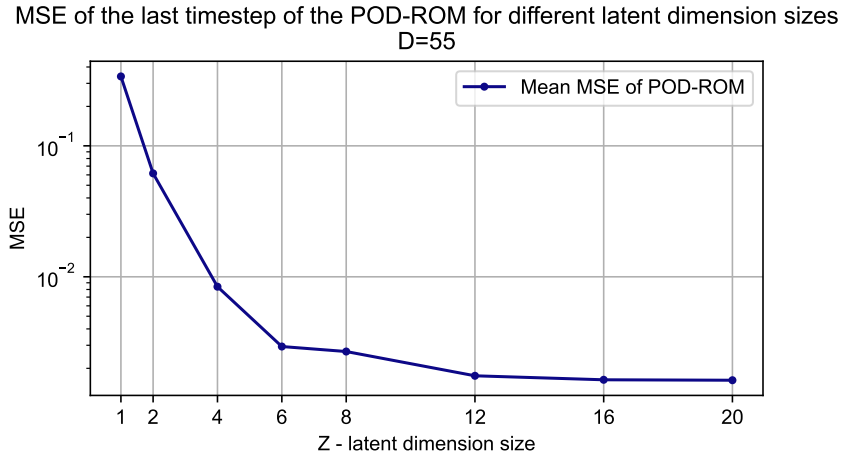
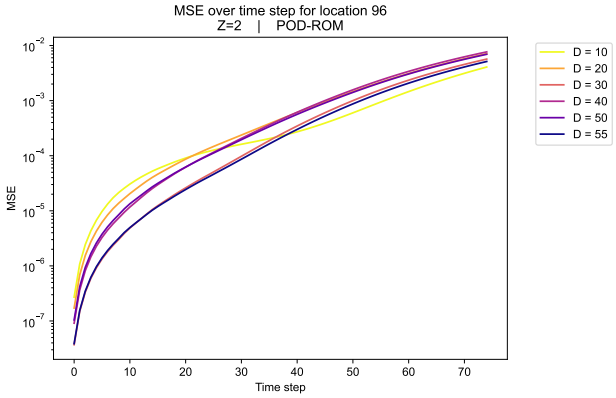


Figure 4.5: For the largest dataset size, extended learning curve for POD-based ROMs over latent dimension sizes.

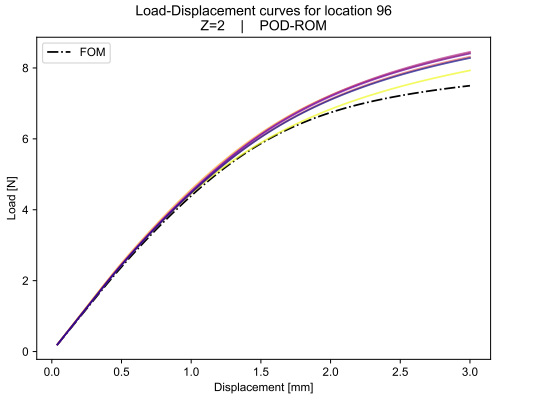
4.3. PERFORMANCE

From the learning curve, it can already be discerned that the errors of the $Z = 1$ and $Z = 2$ POD-ROMs can reach errors of around 0.38 and 0.08 respectively. Be that as it may, these numbers are averaged first over the full displacement field of the last time step, then over the six locations, and over the 20 different dataset draws. For one of these draws (random seed 23, of which the test locations are shown in Fig. 3.2b) two load cases are compared for different dataset sizes. Then, for each dataset size, the performance for each load case is compared.

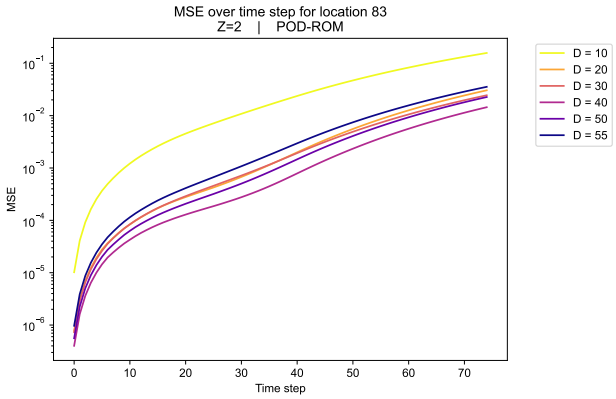
The MSE per time step and the load-displacement curves of two load cases, both $Z = 2$, are shown in Fig. 4.6: location 96 and location 83. Considering the reduced models only use two modes in their bases, these results are quite good. For these two cases, notable is the difference in performance for the $D = 10$ POD-ROM: it outperforms the reduced models based on larger datasets on location 96, but it performs significantly worse on location 83, even though it is not that far away from location 96 on the beam. Relating this to the behaviour shown in Fig. 4.1, a sampled dataset size of $D = 10$ does not perform consistently for the different load locations on the beam: it performs very well in the regions it has information from, but predicts the wrong values outside of these places. The POD-ROMs of which the basis has more information, so based on larger datasets, performs more consistent over the different load cases. To emphasize, using a larger dataset to construct the basis does not by definition improve the result for that location, as discussed for the result of the $D = 10$ ROM on location 96, where the MSE increases for the POD-ROMs based on larger datasets.



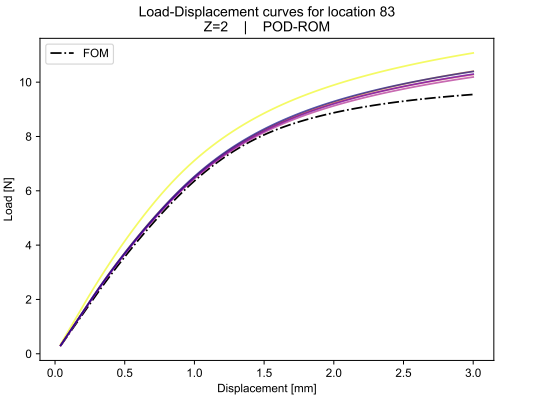
(a) MSE of POD-ROMs for every time step.



(b) Load-displacement curves of POD-ROMs with latent dimension $Z = 2$, compared to the full order solution (FOM).



(c) MSE of POD-ROMs for every time step.

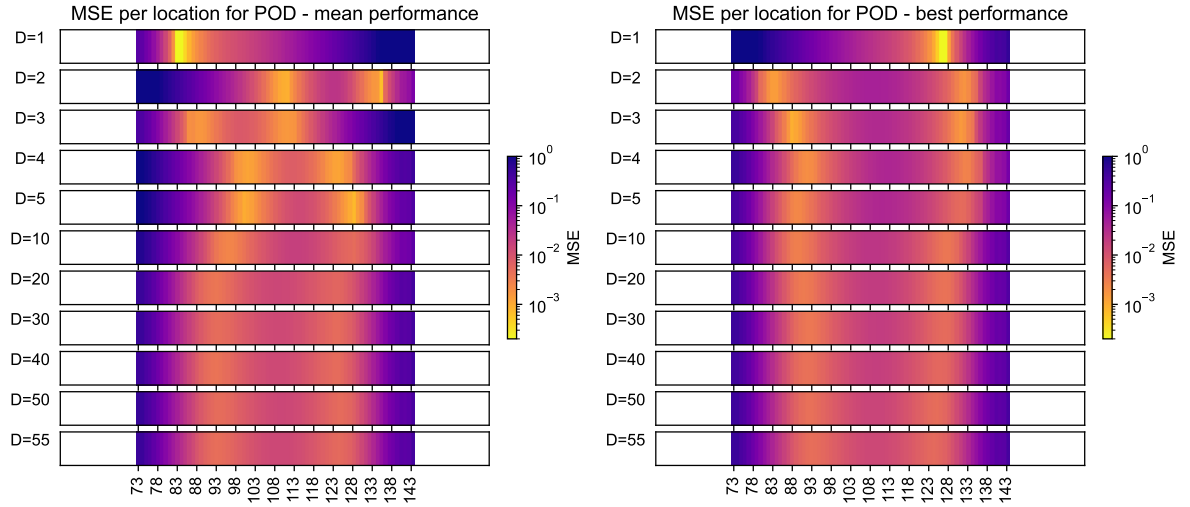


(d) Load-displacement curves of POD-ROMs with latent dimension $Z = 2$, compared to the full order solution (FOM).

Figure 4.6: Comparing POD-ROMs of different dataset sizes for location 83 and 96 as displayed in Fig. 3.2b.

Sampling larger datasets for constructing the POD basis leads to an overall better performance, with a smaller variance in the results. Not only is this the case for the variance within different draws of datasets, as discussed and shown in the learning curve of Fig. 4.4, but also within the load cases on the beam. Fig. 4.7 shows the results of all load cases along the beam. Every bar in the figure has its

own dataset draw: Fig. 4.7a is based on the dataset that performs most similar to the mean value as shown in the learning curve, and Fig. 4.7b displays the best possible dataset draw for that dataset size. Where for $D = 1$ it is very clear what location is sampled from the high accuracy at that location, this gets less clear for increasing dataset sizes. For the larger datasets, it is once again shown how the best result within a beam might decrease in performance, but the general performance in the beam improves. This improvement does stagnate: for the mean performance, this is seen from $D = 20$, and for the best performance already from $D = 5$. Other than that, there is no large distinction in the quality of the draws, especially from a dataset size of $D = 20$ and up.



(a) Using the training dataset draw closest to the average performance as indicated in Fig. 4.4, for every dataset size.

(b) Using the training dataset draw with the best performance as indicated in Fig. 4.4, for every dataset size.

Figure 4.7: Performance of POD-ROM for different dataset sizes, visualized as last time step MSE when loaded at the indicated location. The POD-ROMs in this figure have $Z = 2$.

5

ANALYSIS OF VAE-BASED REDUCED ORDER MODEL RESULTS

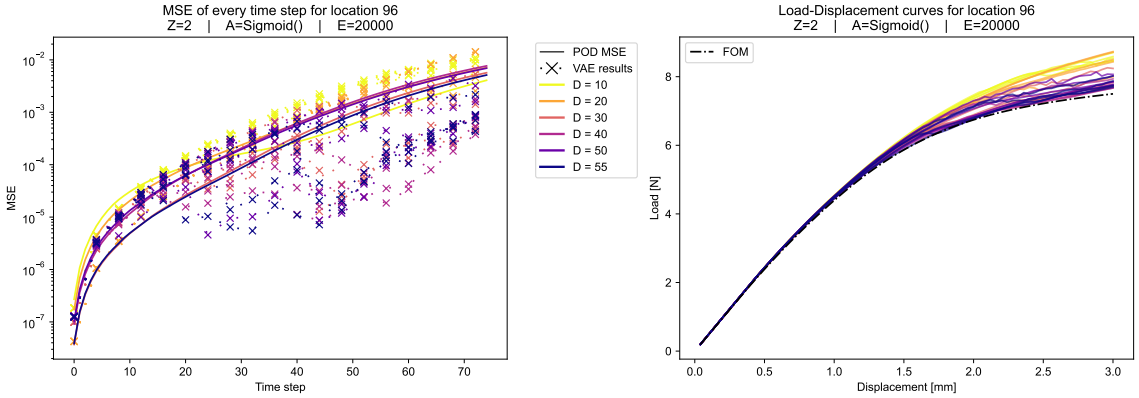
This chapter starts off with the general performance of the VAE-ROMs. To link the results of these reduced order models to the VAEs they are based on, the MSE of the former and the validation loss of the latter are compared. Expanding on this interaction, the chapter continues to investigate how the choices made in the training of the VAE influence the VAE-ROM results. This starts off with looking at the learned manifold, by showing the influence of a faulty state on the quality of the output, in combination with the influence of the regularization through the KLD-weight. The influence of the dataset size and number of training epochs is discussed, ending with a look at the influence of the time step size in the FEM simulation.

5.1. PERFORMANCE OF THE VAE-BASED REDUCED ORDER FINITE ELEMENT MODELS

This chapter will start off with the general performance of the VAE-based Reduced Order Finite Element Models (VAE-ROMs), in order to show the configurations where VAE-ROM performs well and where it does not. To do this, the performance of the VAE-ROMs is compared to their POD counterparts, based on the same latent dimension size on the MSE as calculated in Eq. 4.1. Next to this, a threshold is applied to the VAE-based ROMs, which gives a more absolute performance check instead of a relative one: if a VAE-based ROM is better than its POD counterparts, does it mean it is actually good?

To compare the VAE- and POD-ROMs, they are compared on MSE of the last time step, as that is generally the place of highest interest and of highest error, see also Fig. 5.1a below. In order for a VAE-ROM to be classified as better, its results should have a smaller MSE than those of the POD-ROM, on every test location. The models here are trained for 20K epochs. An example case is shown in Fig. 5.1, which compares different reduced models with each other and to the full order solution. Within the criterion as just described, there is also the matter of test locations within and outside the sampling range, as explained in Fig. 3.2. For test locations within the sampling range, 107 of the 180 VAE-ROM models outperformed their POD counterparts, where outside this range, 134 of such models were found. Thus, inside range 59%, and outside range 74% of the VAE-ROMs outperformed the POD-ROMs, which can also be seen in Fig. 5.2. All 107 models are included in the set of 134. This indicates that the VAE-based ROMs perform better in extrapolation than POD-based ROMs, as more of the VAE-ROMs outperform their POD counterparts in these regions.

In the pool of VAE-ROMs that outperform POD for the same level of reduction, the configurations of (hyper)parameters are inspected. No specific value of any of the hyperparameters stand out as having a high contribution. The different latent dimension sizes, KLD-weights and activation functions, as specified in Table 3.2, contribute more or less equally to these well performing models. For the latent dimension size, it is important to note that the $Z = 2$ VAE-ROMs produced results with a lower MSE than for $Z = 1$, but $Z = 1$ VAE-ROMs outperform $Z = 1$ POD-ROMs to the same degree as for $Z = 2$.



(a) MSE of POD-ROMs and VAE-ROMs for every time step. **(b)** Load-displacement curves of the VAE-ROMs, compared to the full order solution (FOM).

Figure 5.1: Comparing different ROMs for location 96 as displayed in Fig. 3.2b.

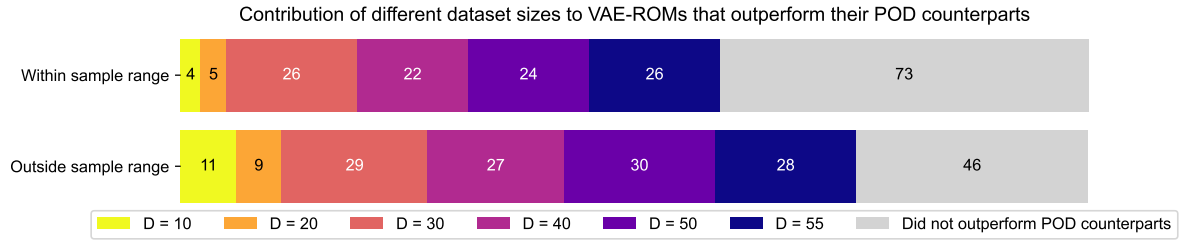
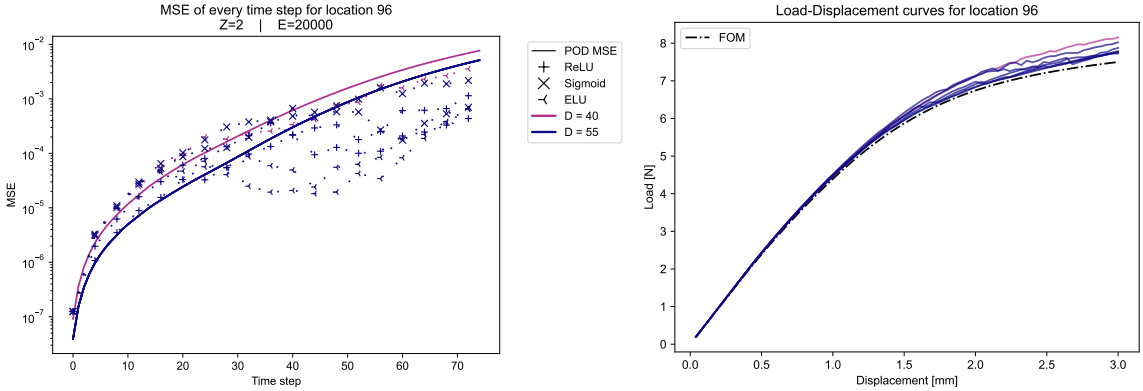


Figure 5.2: The count of the dataset sizes of the VAE-ROMs outperforming their POD counterparts, for test locations inside and outside the sampling range, as shown in Fig. 3.2b.

Table 5.1: Hyperparameters of the VAEs used in the best performing VAE-ROMs.

D	Z	K	A
40	2	$1e^{-2}$	ELU
55	2	0	ReLU
		0	ELU
		$1e^{-8}$	ELU
		$1e^{-6}$	ReLU
		$1e^{-4}$	Sigmoid
		$1e^{-2}$	Sigmoid



(a) MSE of POD-ROMs and VAE-ROMs for every time step. **(b)** Load-displacement curves of the VAE-ROMs, compared to the full order solution (FOM).

Figure 5.3: Results of the VAE-ROMs performing within the threshold of $MSE < 0.1$.

Of the dataset sizes used for training, the dataset sizes of $D = 10$ and $D = 20$ do have a notable lower contribution to the VAE-ROMs that outperform their POD counterparts. The contribution of the different dataset sizes can be found in Fig. 5.2, in reference to all the different configurations of (hyper)parameters, 180 in total.

To be able to look at absolute performance, instead of only a relative one, a threshold can be given for what would be a 'good' performing model. The threshold is put at $MSE \leq 0.1$, on which the models as selected as 'better' will be tested, meaning that on all locations as displayed in Fig. 3.2b the MSE of the last time step cannot exceed 0.1. When applying this threshold, seven VAE-based models are left, of which the specifications are found in table 5.1. Part of the results of these VAE-ROMs is plotted in Fig. 5.3. More results with activation function A=Sigmoid() and latent dimension size $Z = 2$ can be found in Appendix A, including the threshold as introduced here.

5.2. INFLUENCE OF THE DATASET SIZE

In order to apply reduced order modelling, snapshots need to be obtained from the FOM to set up the ROM, which is considered the most computationally heavy part: if this was not computationally costly, then applying ROM would not have added value. Therefore, the quantity of data needed to properly inform the ROM such that it performs as desired, is ideally as small as possible. In section 5.1 it is already found that the higher range of datasets produce the best performing VAE-ROM results, and this is supported with Fig. 5.4. Per dataset size, twenty different random shuffles are obtained to sample the specified number of locations from the sampling range, which are then used as training set. The validation set is constructed from six evenly distributed sampling locations, to be representative of the full dataset. The last step MSE (see Eq. 2.20) of these different random splits are then averaged over the six validation locations and the mean over the twenty draws is plotted, with the standard deviation. Overall, the behaviour shown in this figure is consistent with expectations, where more data improves the quality of the results and lowers the variance. The result and the method are consistent with Fig. 4.4, which shows the same analysis for POD-based ROM.

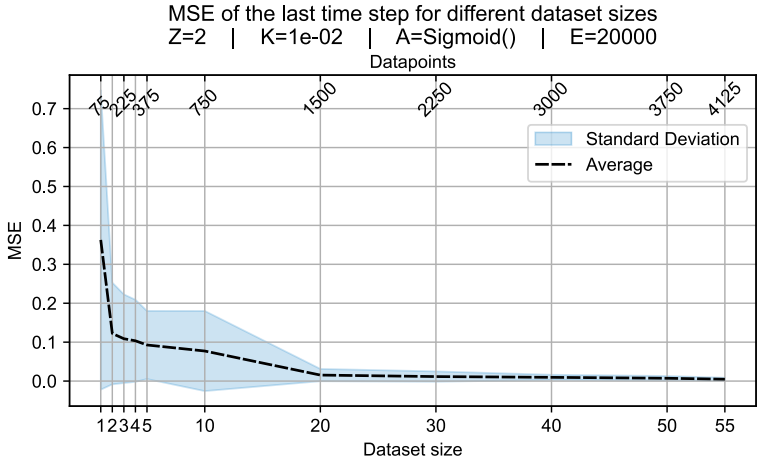
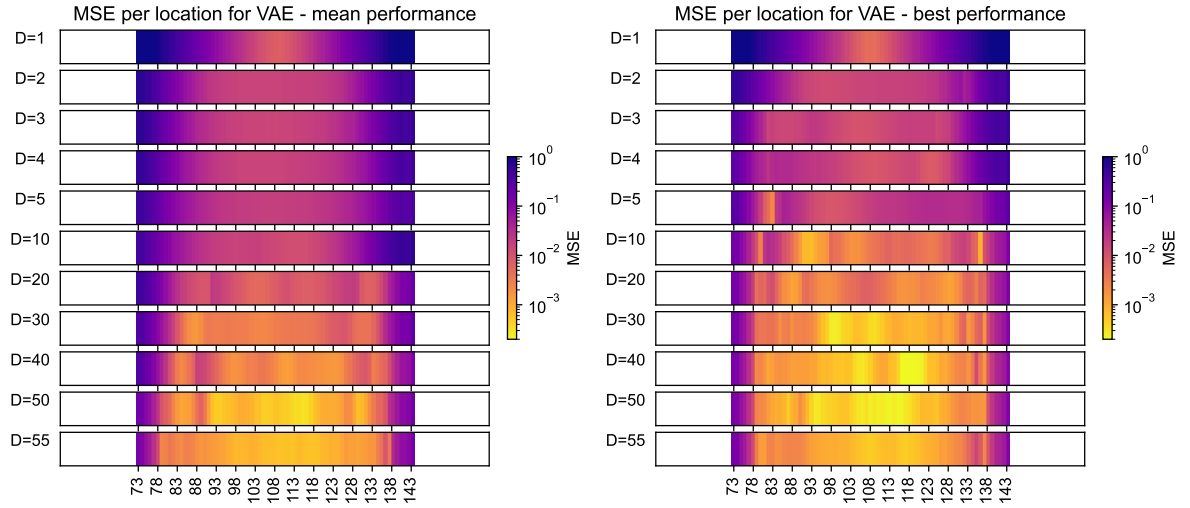


Figure 5.4: Influence of the dataset size on the VAE-ROM MSE and the variance for different locations sampled.

To unpack the many values that the graph of Fig. 5.4 consists of, the same analysis done in Fig. 4.7 for the POD-ROM results, is done here for the VAE-ROM results, of which the result is shown in Fig. 5.5. It is seen here that the results are quite consistent over the different load cases in the beam. Where in the POD-ROM results it was clear where the smaller dataset models were sampled, this is not easily seen in the VAE-ROM results: here the gradient over the performance of the different load cases on the beam is gradual, without clear outliers. There is also a notable increase in performance between the average performance of each dataset and the best performance, based on which load locations are sampled for training the VAE. This distinction was less notable for the POD-ROMs, where also the variance in the learning curve is less pronounced. Comparing these results to the extended POD-ROM learning curve of Fig. 4.5, the POD-based ROM starts to outperform the $D = 55$ VAE-ROM result of Fig. 5.4 from $Z = 6$.

As seen in the results displayed in Fig. 5.2 and 5.5, the VAEs with smaller datasets do not perform as well as those trained with more data. This is also reflected in the models of table 5.1, where only ROMs based on VAEs with more data make it above the specified threshold. From this is clear that the VAE requires many sampled locations, which is in line with observations in [34] and [35], of which the latter also looks into ways to mitigate this.



(a) Using the training dataset draw closest to the average performance as indicated in Fig. 5.4, for every dataset size.

(b) Using the training dataset draw with the best performance as indicated in Fig. 5.4, for every dataset size.

Figure 5.5: Performance of VAE-ROM for different dataset sizes, visualized as last time step MSE when loaded at the indicated location. The VAE-ROMs in this figure have a latent dimension size of $Z = 2$.

5.3. Relation between validation loss and VAE-ROM errors

As described in section 3.4.1, hyperparameters can be selected using a machine learning tracker software to optimize for a small validation loss. However, when applying the VAE inside a reduced FEM model, it is not guaranteed that the validation loss will accurately predict the performance of the VAE-ROM. For this, the validation losses of VAEs trained for 20K epochs are compared to the MSE of the last time step of the corresponding VAE-ROM, averaged over the test locations. When looking at Fig. 5.6, it can be seen that the validation loss associated with the reconstruction capabilities of the VAE is not a good indicator for the performance of the VAE-ROM.

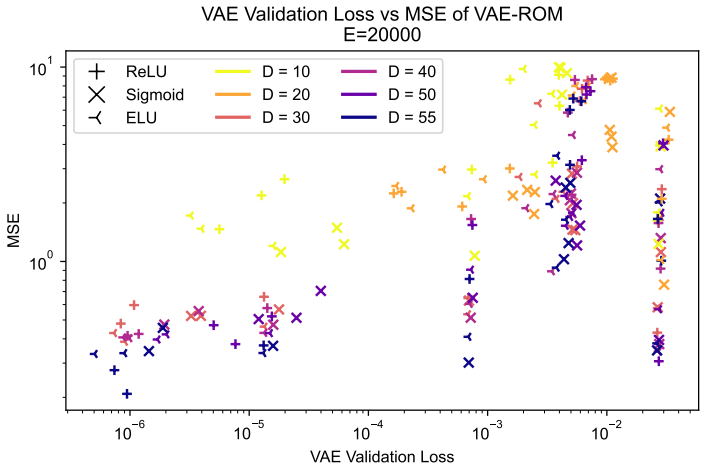
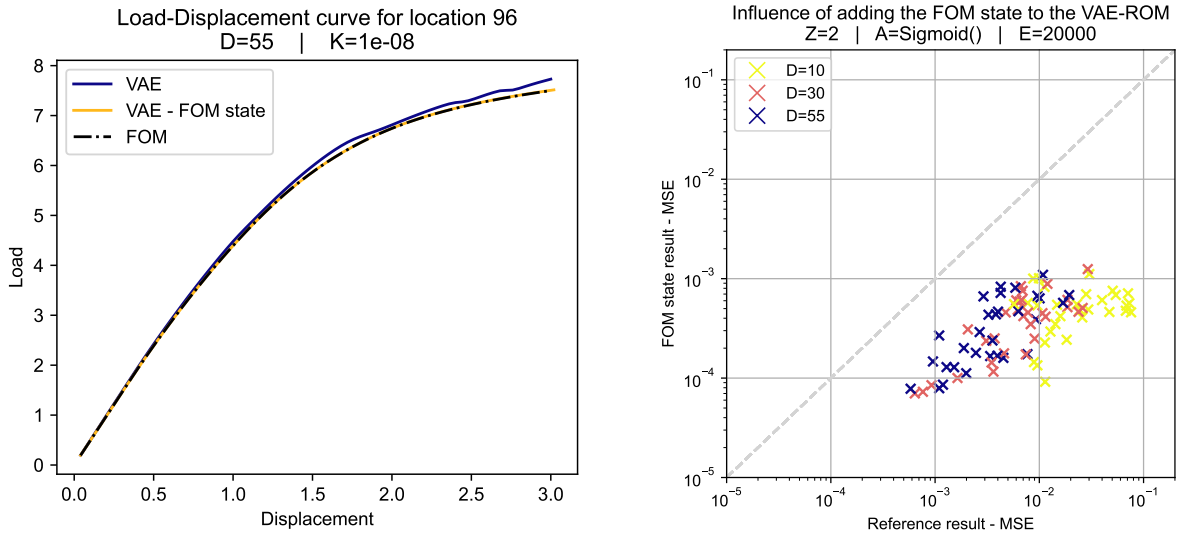


Figure 5.6: Comparison of the validation loss of a VAE and the mean squared error of the last time step of the corresponding VAE-based ROM, showing no clear trend. The results of all different values for Z and K are included in this plot.

5.4. INFLUENCE OF THE LEARNED MANIFOLD

In Algorithm 1 for the reduced order finite element model, the basis used for reduction is dependent on the location on the manifold, determined by the state of the solver. The state passes through the VAE, after which the first part of the basis is obtained with backpropagation, to which the state is then added as the first column. However, since the solution at the end of a time step is not the exact solution due to information lost during reduction, the location on the manifold is also not exact: a faulty location. In order to distinguish what part of the error in the ROM solutions is caused by this faulty location and what part by a faulty manifold, the FE algorithm is altered to at every time step take the FOM solution to determine the basis, after which the algorithm will continue as before. A faulty manifold in this case means that the reduction and reconstruction by the VAE introduces errors.



(a) Load-Displacement curves for one of the values plotted on the right.

(b) Scatter plot of the last time step MSE, results of all possible values for K are included.

Figure 5.7: Showing the effect of putting in the FOM state at every time step in the VAE-ROM

From Fig. 5.7 it can be seen how adding the FOM state to every time step to obtain the basis lowers the MSE of the last time step. The grey diagonal in the right plot indicates the boundary where the values with and without the FOM state added would be equal: values below this line indicate that adding the FOM state improves the results of the VAE-ROM, while any value on the line would indicate that adding the FOM state has no influence on the VAE-ROM result. This means that differences in results between the FOM and VAE-ROM results are not only caused by the information loss during reduction, but also by the quality of the states when the reduction takes place.

The effect of errors in the state is not the same for every VAE used, but depends on how much the latent space is regularized. Regulation of the latent space is done by the KLD-weight, as explained in section 2.3.3. The influence of this regularization is visualized in Figs 5.8 and 5.9. The first of these shows that the more the latent space is regularized, meaning a higher KLD-weight, the less the quality of the input state has an influence on the VAE-ROM output error. However, with that the results move closer to the grey dashed line, they also move up in the plot: a faulty state has less influence, but the general result is worse.

This observation can be associated with the visualisation of a 2D latent space in Fig. 5.9. The latent space visualisation is obtained by passing the FOM state of every time step through a VAE (dataset size of 55 sampled locations, activation function Sigmoid and trained for 20K epochs) for different KLD-weights and for five different locations. Note that this is not a representation of the latent space as present in the ROM solver, as there the current state is added to the basis, resulting in a 3D latent space for a VAE trained for a latent space of $Z = 2$. The figure shows how, for an increasing regularisation, the latent space moves from clear defined lines for each location to scattered values, resembling a normal distribution. This scattering starts at the earlier time steps and at the outer edges of the beam, as

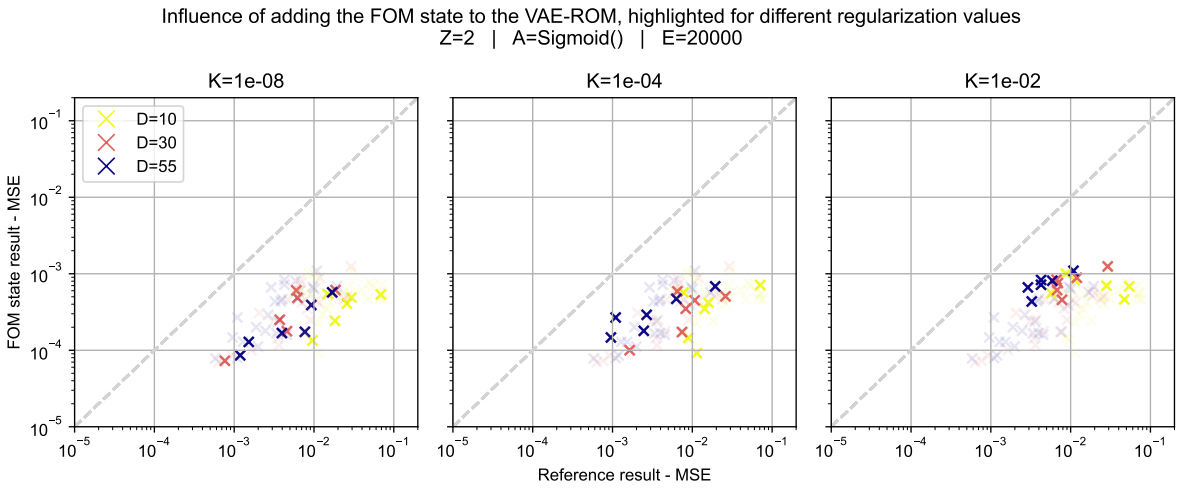
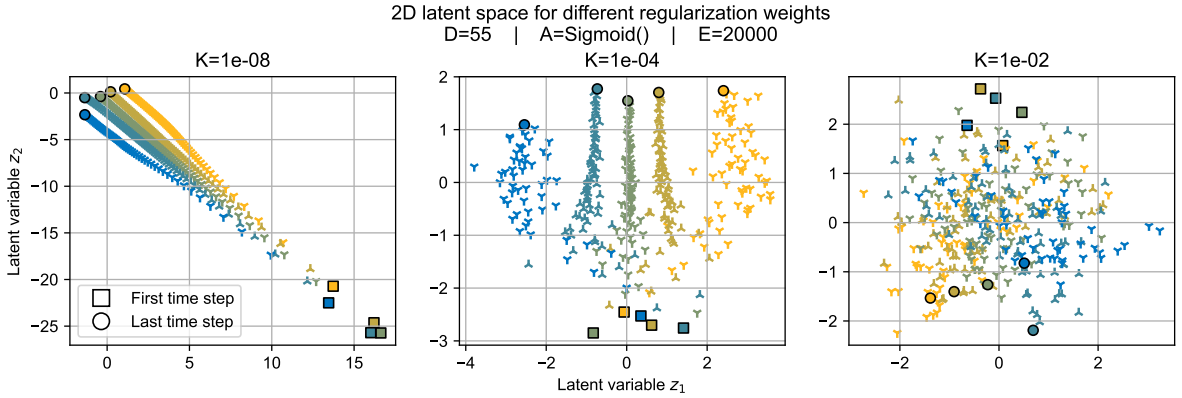
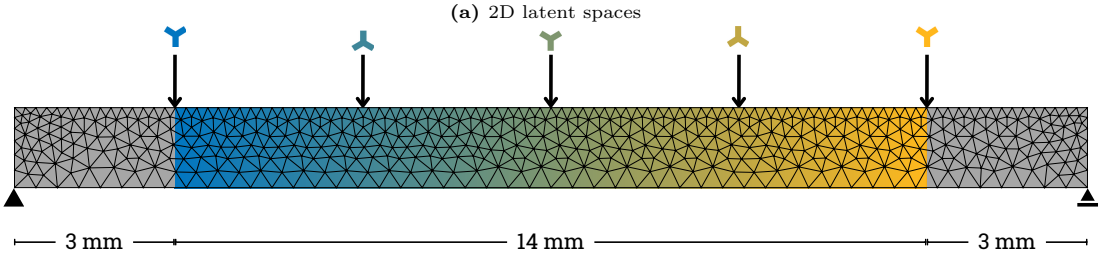


Figure 5.8: Showing the effect of adding the FOM state at every time step in the VAE-ROM, highlighted for different regularization weights.



(a) 2D latent spaces



(b) Locations on the beam

Figure 5.9: Showing a 2D latent space representation of five locations along the beam, for different regularization weights.

these are the areas with more uncertainty in the manifold: either it was not specifically trained for those locations, or the state can still belong to either location because it is still close to a zero-state. When comparing the 2D interpretation of the manifold to the influence of adding the FOM state to models with different regularization values, they are in line with each other: with a more regularized space, errors in the state will sooner be seen as noise by the VAE than in a less regularized space. For the most regularized model, it can be questioned what the VAE used in the model has learned that can add value to the reduced simulation. Adaptability to the state and thus the load location of the case at hand is an important advantage of using a VAE to reduce the simulation, and this latent space seems to have no discrimination for the different load cases. To explore this, simulations were run where the results of VAE-ROMs using a KLD-weight of K=1e-02 were compared to the results that did not use the VAE, but just the initial linear basis. In this analysis, the ROM that used the regularized VAE still performed better than the one without, even though the latent space representation might not be interpreted as distinctive.

5.5. INFLUENCE OF NUMBER OF TRAINING EPOCHS

Every time the VAE has seen the full training dataset, an epoch has passed in training the model. An indicator that sufficient epochs have passed in training, is a stable validation loss: if this does not improve significantly, training can be stopped and the model with the best validation loss can be used. As already discussed and shown in section 5.3, the validation loss obtained during training of the VAE is not a good indicator for the performance of the VAE-ROM. This point is once again supported by the behaviour shown in Fig. 5.10. This figure shows in the top part how the MSE of the VAE-ROM results changes for the number of epochs trained. The mean trend of the results for different dataset sizes is indicated by the dashed lines. With these mean trends, it is clear how the results keep improving for training up to 10K epochs, after which some results keep improving and others get worse. Do note that, in contrary to what is shown in section 5.2, these datasets are based on one single draw of sampling locations and not averaged on multiple random samplings. In the bottom graph, the minimum validation loss for every 200 epochs of training is displayed, showing that the VAEs reach a minimum value relatively fast, after which it does not reach lower levels. The behaviours shown in this figure again show that the validation loss during training of the VAE is not a good indicator for the performance of the VAE-ROM, seen also in the great spread within the VAE-ROM results. Next to that, it also shows how the validation loss on its own is not sufficient to determine whether the VAE is well-trained.

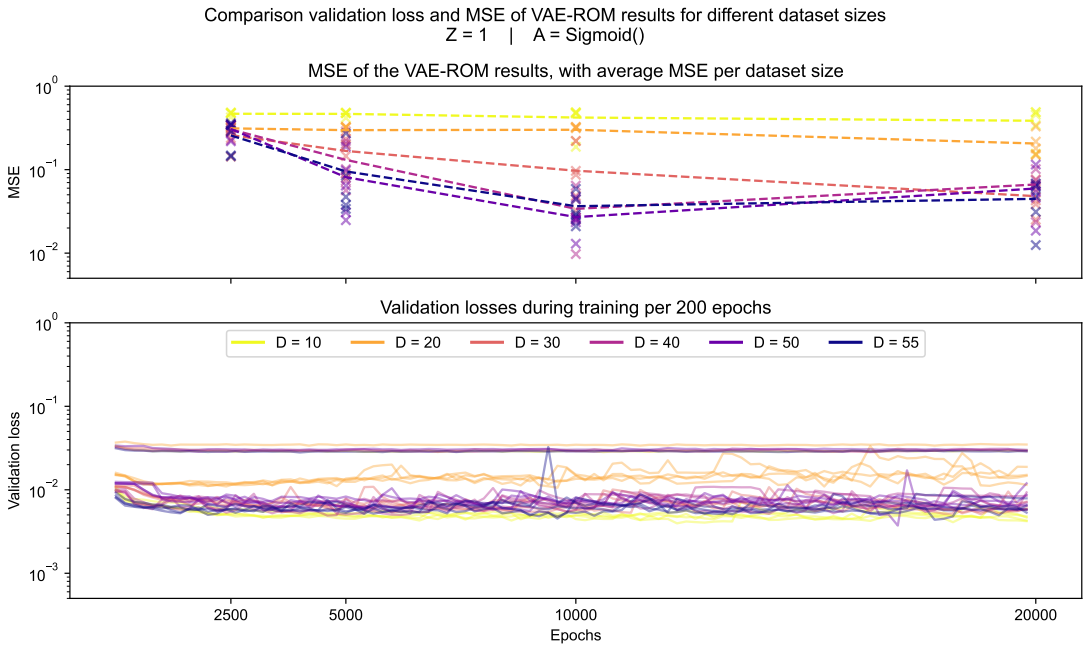


Figure 5.10: MSE of VAE-ROM results and validation loss curves of VAEs of latent dimension Z=1 and with activation function Sigmoid, for different dataset sizes and regularization weights. The top plot shows the MSE. For clarity, the lowest validation loss per 200 epochs is plotted in the bottom graph.

5.6. INFLUENCE OF TIME STEP SIZE

The size of the time step in the nonlinear FEM simulation is of more importance than usual when working with a VAE-ROM, where the basis is updated every time step. A larger time steps means that the basis will get updated fewer times during the simulation. An indication of how this influences the results can be found in Fig. 5.11. The difference in jaggedness of the curves of the different step sizes is seen, although it is more prominent in the more regularized models. The jumps in the graphs correspond to the updates of the basis, and while the 0.04 curve is often able to jump back to the course that the 0.02 curve is on, the 0.2 step size curve is less successful in doing so. In this, the phenomenon as described in Section 5.4 can be recognized, as with larger time steps, there is likely to be a more erroneous state fed to the VAE to extract the gradient from, leading to a faulty basis.

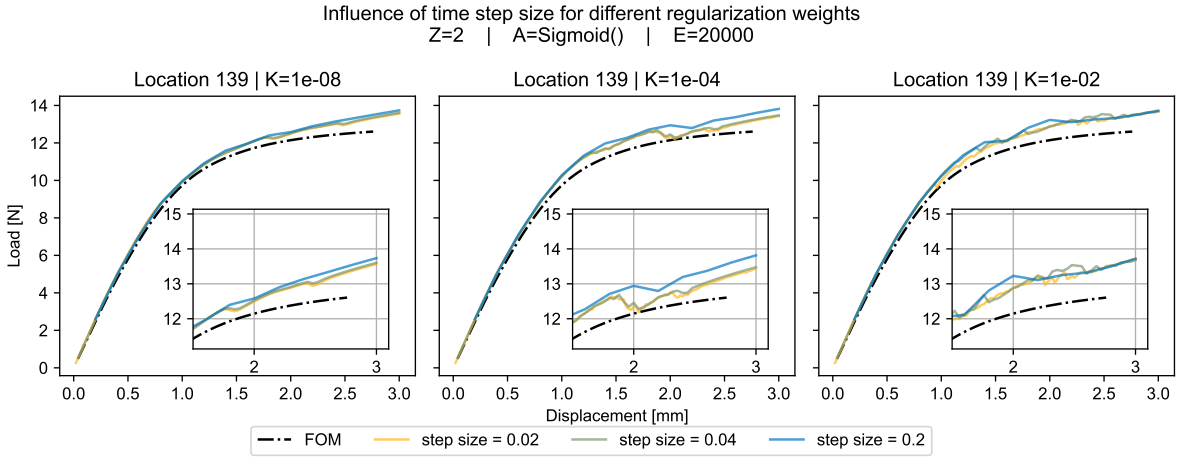


Figure 5.11: Load-Displacement curves of location 139, for three different time step sizes.

6

CONCLUSIONS AND RECOMMENDATIONS

This thesis sought to explore the possibilities of improving on the Proper Orthogonal Decomposition-based model order reduction techniques in computational mechanics, specifically using a Variational Autoencoder-based strategy. Literature review into existing POD-based strategy was the starting point, while simultaneously generating data for the case study. From this dataset, linear bases were extracted using singular value decomposition (SVD). After finding a suiting neural network architecture tailored to the dataset, Variational Autoencoders (VAEs) were trained for a range of (hyper)parameters, making use of the DelftBlue supercomputer. These VAEs are able to find nonlinear patterns in the dataset. To be able to integrate both the linear and nonlinear techniques into a Finite Element solver, a displacement-dependent arc length algorithm was developed, allowing the user to switch between the two reduced techniques, as well as a full order method, by altering just a few arguments. Results obtained are analysed and compared between the POD- and VAE-based techniques, but also between different (hyper)parameters within the VAE-ROM results. The analyses conducted were executed to gain insights to be able to answer the main research question of the thesis, which asks if a VAE-based strategy is an improvement on POD-based ROM. This chapter summarizes the conclusions to this main question, structured to the subquestions as posed in the introduction, after which recommendations on this work and future work are given.

6.1. CONCLUSIONS

In order for there to be an improvement on POD-based ROMs found in VAE-based techniques, clarity is needed on where the limitations lie of this linear technique in relation to nonlinear equilibrium analysis, which is what the first subquestion focussed on. Two main limitations of POD-ROMs are found in their global linearity. The global reduction of the data does not allow for space or time specific patterns in the dataset to be captured: it simply cannot distinguish these. This is most clearly seen in the example of constructing the linear basis from only one sampled location: the information that the learned displacement shape is specific to the location of the load is not registered, and the POD-ROM is unable to adapt for a different load location. The linear way in which the reduction is executed through SVD cannot represent nonlinear patterns. This is a major limitation, as in complex models, where model order reduction would be most beneficial, their highly nonlinear patterns are often what makes them computational heavy. POD-based ROMs need substantial reinforcement to be of use in these cases. To specify this for the context of nonlinear equilibrium analysis in the field of structural and material mechanics, limitations for the linear techniques are prominent in nonlinear problems with local stress or strain concentrations, such as crack propagation problems.

The second phase then poses the question if introducing nonlinearity in the reduction through a VAE improves the quality of the ROM. Results found that for test cases within the sampling range on the simply supported beam of the case study, a majority of the VAE-based ROMs outperformed their POD counterparts. This trend was even stronger for the test cases where the reduced models had to extrapolate: the model with linear bases reached high errors and, when looking over the full range of test locations, produced less accurate results than the best VAE models.

In total, 180 different configurations of (hyper)parameters were used to train as many VAEs. The influence of these different parameters on the performance of the VAE-ROMs was analysed as well. Mirroring the POD results, the best results were obtained using a VAE trained with a latent space size of $Z = 2$, but VAE-ROMs reducing to $Z = 1$ were also able to outperform $Z = 1$ POD-ROM counterparts. Of the dataset sizes used for training, the dataset sizes of $D = 10$ and $D = 20$ do have a notable lower contribution to the VAE-ROMs that outperform their POD counterparts. The fact that the performance of these models based on less data is worse, is in line with the learning curve of the VAE-ROM for different dataset sizes, as well as the performance for all load locations along the beam, for different dataset sizes. These both show that in order for the VAE-ROM to perform well, the VAE needs many sampled locations. Looking at the different KLD regularization weights and the activation functions used, all of them contribute quite evenly to the pool of successful VAE-based models, meaning there is no optimum level of regularization for the VAEs in this context, nor a single best activation function. However, with a more regularized model, the quality of the results is less influenced by errors in the state. This is in line with the effect that a higher KLD-weight has on the structure of the latent space. The connection between the validation loss in training and the errors of the VAE-ROM results was investigated, because a VAE needs to be trained and deemed accurate enough to use in a VAE-ROM. The validation loss is a poor indicator for the quality of the online results, which was indicated not just by a simple scatter plot, but also when looking at the influence of the number of training epochs. Here it was seen how for a relatively stable validation loss, the results of the VAE-ROM can still improve. Finally, it was seen how even at the last part of the implementation of the VAE-ROM, the choice of time step size has an influence on the quality of the result, similar to regular FEM, but with the additional consideration that finer step size corresponds with more basis updates within the simulation.

6.2. Notes, limitations and recommendations

Developing an algorithm for which the integrated VAE produced logical and accurate results which outperformed the POD-based model cost many iterations: it took more than simply introducing matrix multiplications to reduce the stiffness matrix and force vectors. Part of this was due to the low complexity level of the case study: even though the simply supported beam was loaded into plasticity, the number of degrees of freedom were not exceedingly large and there were no stress or strain concentrations involved. This made the results of the POD-ROM very accurate for the fact that they only use one or two modes in the basis. Next to that, the performance of the VAE only gets up to par for the large datasets, which is a problem more observed for VAEs. Combining these two observations, this specific case study would likely be more efficiently reduced by adding three to four additional modes to the POD basis, which would result in still a significant reduction, but would omit the need to develop a VAE.

Still, from the results it is concluded that the VAE-ROM can be successful, but the right context needs to be found. In order to do this, further research can be done to map out exactly where the limitations of POD lie in the context of mechanical equilibrium analysis. To apply the VAE further in nonlinear model order reduction, three main points can be taken from this work. Firstly, the architecture of the VAE is calibrated for the data, resulting in a shallow VAE with only the minimum number of hidden layers needed to obtain nonlinear reduction. For more complex datasets, deeper VAEs might be more suitable, which need to be calibrated accordingly. Secondly, the displacement-dependent arc length algorithm developed can be easily adapted and applied to a number of projection-based Finite Element models. Thirdly and lastly, because it is found that the validation loss during the training of the VAE is not a good indication of the performance in a VAE-based ROM, adaptations can be made to the training procedure to either periodically check the VAE-ROM error during training, or even to calculate the validation loss through the VAE-ROM during training directly.

REFERENCES

1. Phillips, T. R. F., Heaney, C. E., Smith, P. N. & Pain, C. C. An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion. *International Journal for Numerical Methods in Engineering* **122,** 3780–3811. ISSN: 1097-0207. https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6681 (2021).
2. Rathinam, M. & Petzold, L. R. A New Look at Proper Orthogonal Decomposition. *SIAM Journal on Numerical Analysis* **41.** Publisher: Society for Industrial and Applied Mathematics, 1893–1925. ISSN: 0036-1429. https://epubs.siam.org/doi/abs/10.1137/S0036142901389049 (Jan. 2003).
3. Kerfriden, P., Gosselet, P., Adhikari, S., Bordas, S. & Passieux, J.-C. POD-based model order reduction for the simulation of strong nonlinear evolutions in structures: Application to damage propagation. *IOP Conference Series: Materials Science and Engineering* **10,** 012165. ISSN: 1757-899X. https://dx.doi.org/10.1088/1757-899X/10/1/012165 (June 2010).
4. Eivazi, H., Veisi, H., Naderi, M. H. & Esfahanian, V. Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids* **32,** 105104. ISSN: 1070-6631, 1089-7666. https://pubs.aip.org/pof/article/32/10/105104/1060329/Deep-neural-networks-for-nonlinear-model-order (Oct. 1, 2020).
5. Lee, K. & Carlberg, K. *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders* June 5, 2019. arXiv: 1812.08373[cs]. http://arxiv.org/abs/1812.08373.
6. Akkari, N., Casenave, F., Hachem, E. & Ryckelynck, D. A Bayesian Nonlinear Reduced Order Modeling Using Variational AutoEncoders. *Fluids* **7.** Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, 334. ISSN: 2311-5521. https://www.mdpi.com/2311-5521/7/10/334 (Oct. 2022).
7. Kudela, J. & Matousek, R. Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Computing* **26,** 13709–13733. ISSN: 1433-7479. https://doi.org/10.1007/s00500-022-07362-8 (2024) (Dec. 1, 2022).
8. Shinde, K., Itier, V., Mennesson, J., Vasiukov, D. & Shakoor, M. Dimensionality reduction through convolutional autoencoders for fracture patterns prediction. *Applied Mathematical Modelling* **114,** 94–113. ISSN: 0307-904X. https://www.sciencedirect.com/science/article/pii/S0307904X22004541 (Feb. 1, 2023).
9. Rocha, I. B. C. M. Numerical and Experimental Investigation of Hygrothermal Aging in Laminated Composites. https://repository.tudelft.nl/islandora/object/uuid%3A0eab23c7-9ba4-4d27-91ee-58f9f140dd34 (2019).
10. Aragón, A. M. & Duarte, C. A. in *Fundamentals of Enriched Finite Element Methods* (eds Aragón, A. M. & Duarte, C. A.) 1–10 (Elsevier, Jan. 1, 2024). ISBN: 978-0-323-85515-0. https://www.sciencedirect.com/science/article/pii/B9780323855150000076.
11. Tekkaya, A. E. & Soyarslan, C. in *CIRP Encyclopedia of Production Engineering* (eds Laperrière, L. & Reinhart, G.) 508–514 (Springer, Berlin, Heidelberg, 2014). ISBN: 978-3-642-20617-7. https://doi.org/10.1007/978-3-642-20617-7_16699.
12. Zienkiewicz, O. & Taylor, R. *The finite element method, fourth edition; volume 2: solid and fluid mechanics, dynamics and non-linearity* (1991).
13. Crisfield, M. in *Non-Linear Finite Element Analysis of Solids and Structures, Essentials* (Wiley, Chichester, 2000). ISBN: 978-0-471-97059-0. https://web-p-ebscohost-com.tudelft.idm.oclc.org/ehost/ebookviewer/ebook/bmxlYmtfXzE3OTAwX19BTg2?sid=5fe3ddbc-4c54-496a-85a1-b699c32a1ae2@redis&vid=0&format=EB&lpid=lp_266-2&rid=0.

14. Riks, E. An incremental approach to the solution of snapping and buckling problems. *International Journal of Solids and Structures* **15,** 529–551. ISSN: 0020-7683. https://www.sciencedirect.com/science/article/pii/0020768379900817 (Jan. 1, 1979).
15. Ghattas, O. & Willcox, K. Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numerica* **30,** 445–554. ISSN: 0962-4929, 1474-0508. https://www.cambridge.org/core/journals/acta-numerica/article/learning-physicsbased-models-from-data-perspectives-from-inverse-problems-and-model-reduction/C072A4B417F8C3873ED75C1D63BBB31D (May 2021).
16. Wang, J. *Geometric Structure of High-Dimensional Data and Dimensionality Reduction* ISBN: 978-3-642-27496-1 978-3-642-27497-8. http://link.springer.com/10.1007/978-3-642-27497-8 (Springer, Berlin, Heidelberg, 2011).
17. Cassel, K. W. *Projection-Based Model Reduction Using Asymptotic Basis Functions* in *Computational Science – ICCS 2019* (eds Rodrigues, J. M. F. *et al.*) (Springer International Publishing, Cham, 2019), 465–478. ISBN: 978-3-030-22747-0.
18. Liang, Y. C. *et al.* PROPER ORTHOGONAL DECOMPOSITION AND ITS APPLICATIONS—PART I: THEORY. *Journal of Sound and Vibration* **252,** 527–544. ISSN: 0022-460X. https://www.sciencedirect.com/science/article/pii/S0022460X01940416 (May 2, 2002).
19. Candes, E. J., Li, X., Ma, Y. & Wright, J. *Robust Principal Component Analysis?* Dec. 18, 2009. arXiv: 0912.3599[cs,math]. http://arxiv.org/abs/0912.3599.
20. Chandrasekaran, V., Sanghavi, S., Parrilo, P. A. & Willsky, A. S. Rank-Sparsity Incoherence for Matrix Decomposition. *SIAM Journal on Optimization* **21,** 572–596. ISSN: 1052-6234, 1095-7189. arXiv: 0906.2220[math,stat]. http://arxiv.org/abs/0906.2220 (Apr. 2011).
21. Brunton, S. L. & Kutz, J. N. *Data Driven Science & Engineering* (2017).
22. Plaut, E. *From Principal Subspaces to Principal Components with Linear Autoencoders* Dec. 28, 2018. arXiv: 1804.10253[cs,stat]. http://arxiv.org/abs/1804.10253.
23. Al-Digeil, M. *et al. PCA-Boosted Autoencoders for Nonlinear Dimensionality Reduction in Low Data Regimes* May 23, 2022. arXiv: 2205.11673[cs]. http://arxiv.org/abs/2205.11673.
24. Wu, L. & Noels, L. Recurrent Neural Networks (RNNs) with dimensionality reduction and break down in computational mechanics; application to multi-scale localization step. *Computer Methods in Applied Mechanics and Engineering* **390,** 114476. ISSN: 0045-7825. https://www.sciencedirect.com/science/article/pii/S0045782521006940 (Feb. 15, 2022).
25. Kim, Y., Choi, Y., Widemann, D. & Zohdi, T. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics* **451,** 110841. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/S0021999121007361 (Feb. 15, 2022).
26. Kerschen, G., Golinval, J.-c., VAKAKIS, A. F. & BERGMAN, L. A. The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview. *Nonlinear Dynamics* **41,** 147–169. ISSN: 1573-269X. https://doi.org/10.1007/s11071-005-2803-2 (2024) (Aug. 1, 2005).
27. Lote, S., B, P. K. & Patrer, D. Neural networks for machine learning applications. *World Journal of Advanced Research and Reviews* **6.** Last Modified: 2024-07-08T15:50+05:30 Publisher: World Journal of Advanced Research and Reviews, 270–282. ISSN: 2581-9615. https://wjarr.com/content/neural-networks-machine-learning-applications (2020).
28. Bishop, C. M. & Nasrabadi, N. M. *Pattern recognition and machine learning* **4** (Springer, 2006).
29. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* Dec. 10, 2022. arXiv: 1312.6114[cs,stat]. http://arxiv.org/abs/1312.6114.
30. Rivera, M. *How to train your VAE* in *2024 IEEE International Conference on Image Processing (ICIP)* (IEEE, 2024), 3882–3888.
31. Tjensvoll, K. Optimizing the Reduced Basis Construction for Reduced-Order Mechanical Models: Automatic and efficient load case selection using Bayesian machine learning. https://repository.tudelft.nl/islandora/object/uuid%3A3e4a2119-0507-48ca-8aea-7fdbc3d98a64 (2019).

32. *Weights & Biases: The AI Developer Platform* Weights & Biases. https://wandb.ai/site/ (2025).

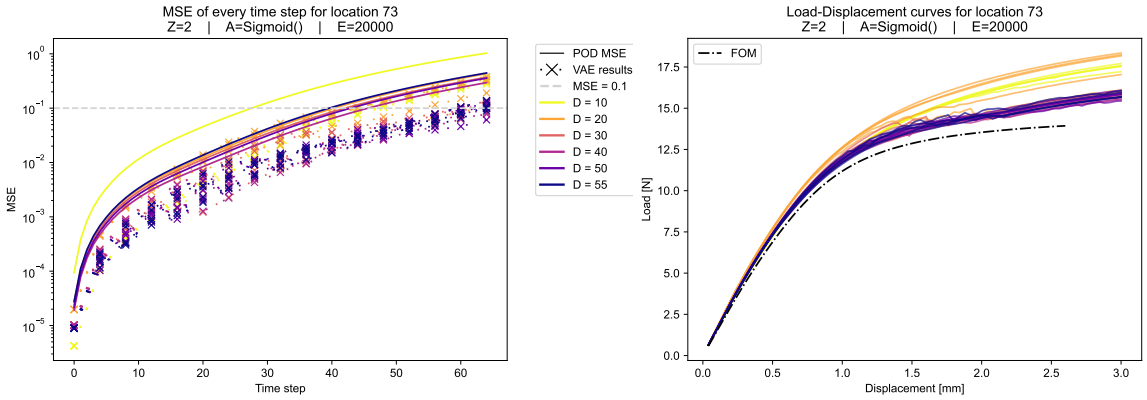
33. Colomés, O., Rocha, I., van der Meer, F. & Leseur, M. *Finite Elements in Civil Engineering and Geosciences* https://interactivetextbooks.citg.tudelft.nl/computational-modelling/intro.html (2024).

34. Scheunemann, L. & Faust, E. *A manifold learning approach to nonlinear model order reduction of quasi-static problems in solid mechanics* Aug. 22, 2024. arXiv: 2408.12415[cs]. http://arxiv.org/abs/2408.12415.

35. Chadebec, C. & Allassonnière, S. *A Geometric Perspective on Variational Autoencoders* Nov. 3, 2022. arXiv: 2209.07370[cs,stat]. http://arxiv.org/abs/2209.07370.

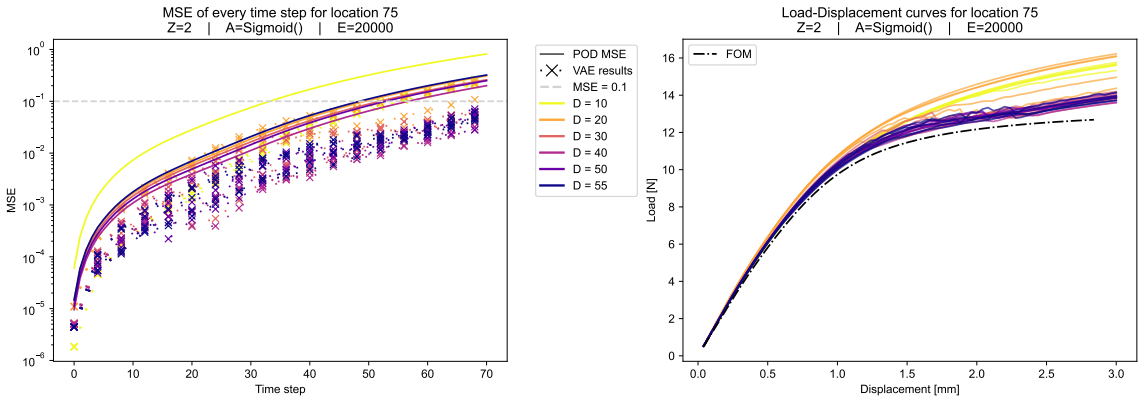
A

ADDITIONAL VAE-ROM RESULTS



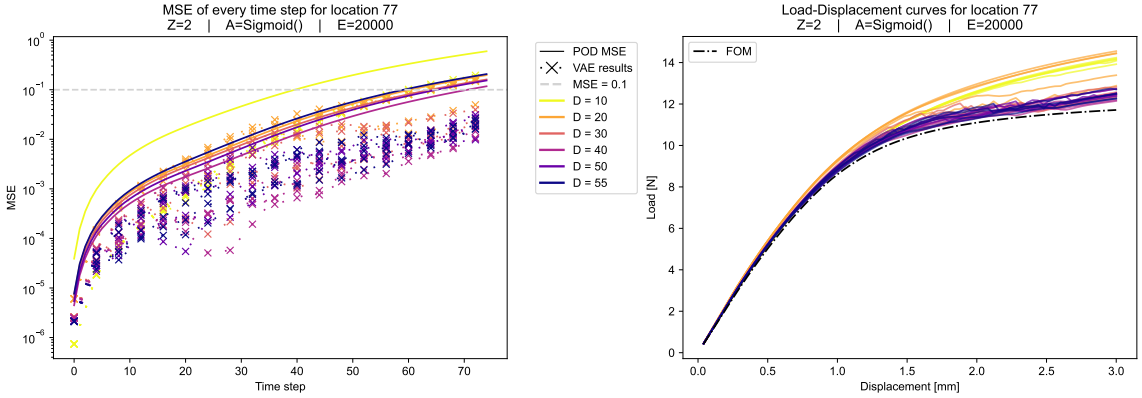
(a) **(b)**

Figure A.1: Selection of results for location 73.



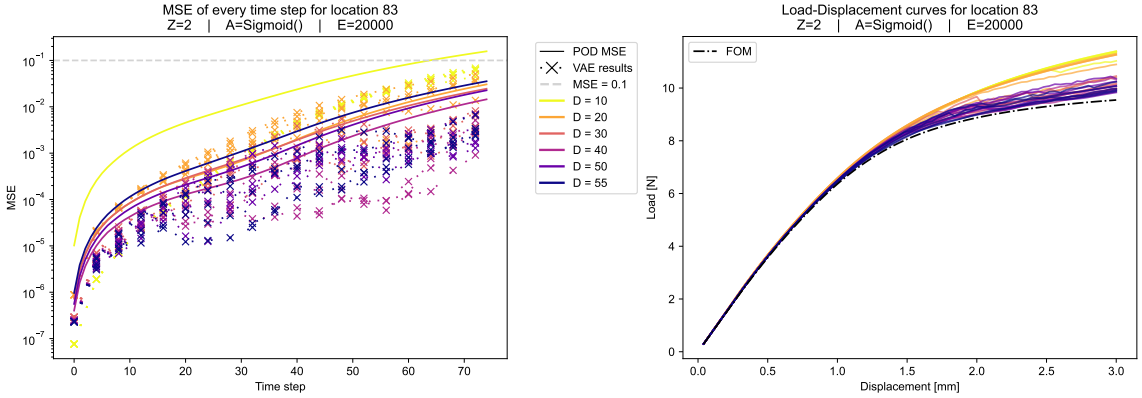
(a) **(b)**

Figure A.2: Selection of results for location 75.



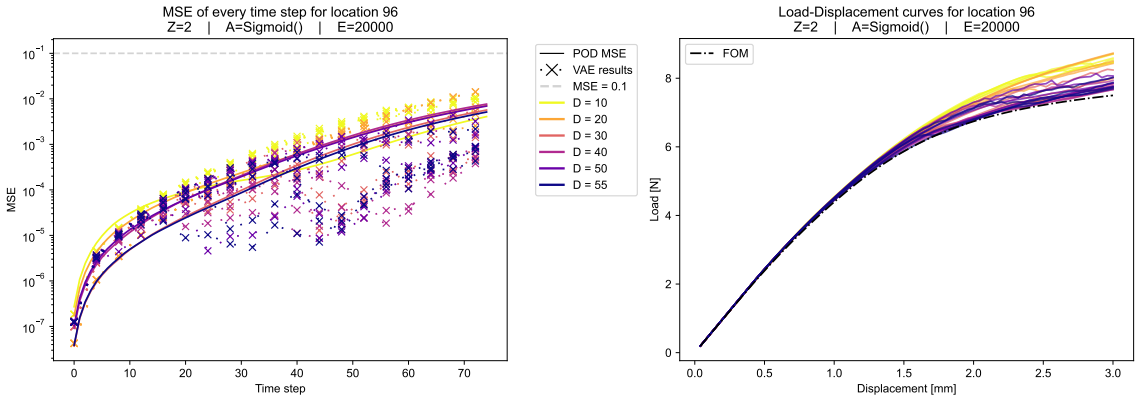
(a) **(b)**

Figure A.3: Selection of results for location 77.



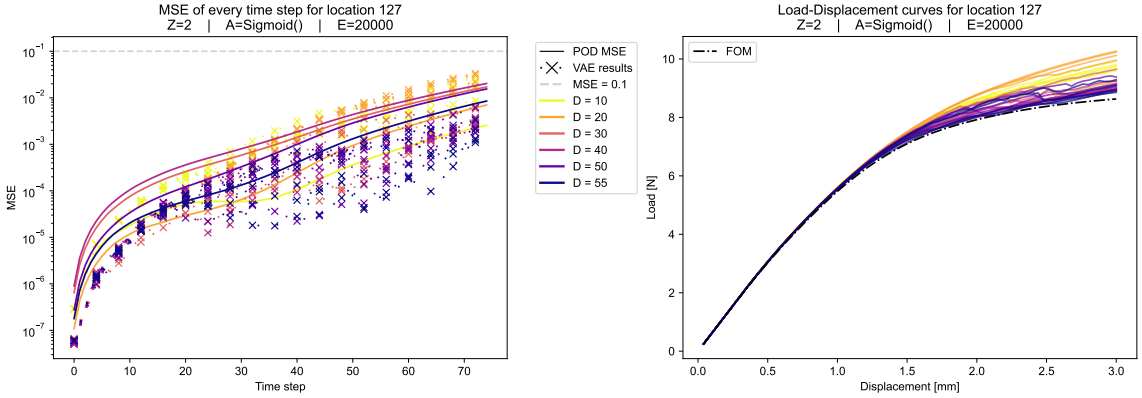
(a) **(b)**

Figure A.4: Selection of results for location 83.



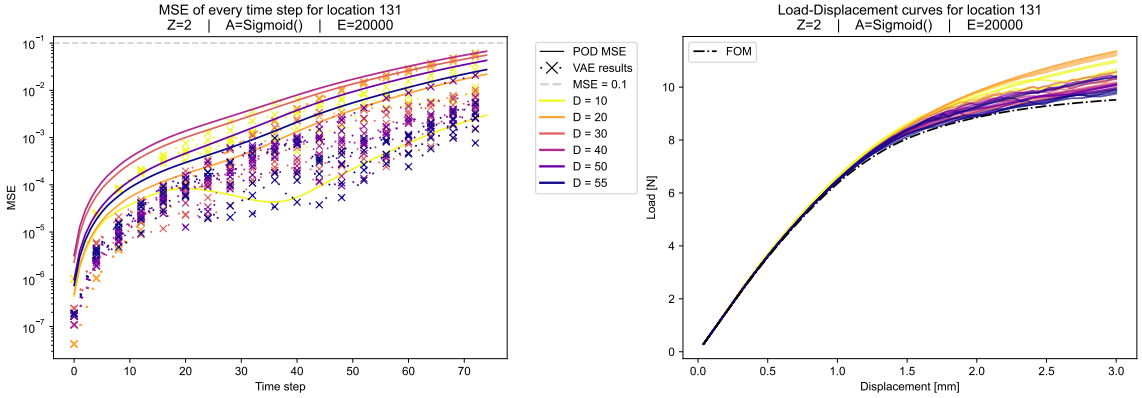
(a) **(b)**

Figure A.5: Selection of results for location 96.



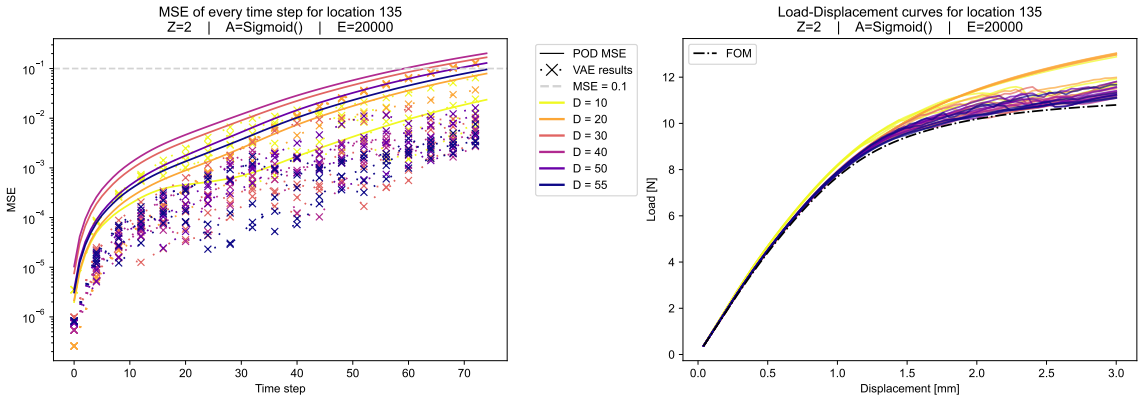
(a) **(b)**

Figure A.6: Selection of results for location 127.



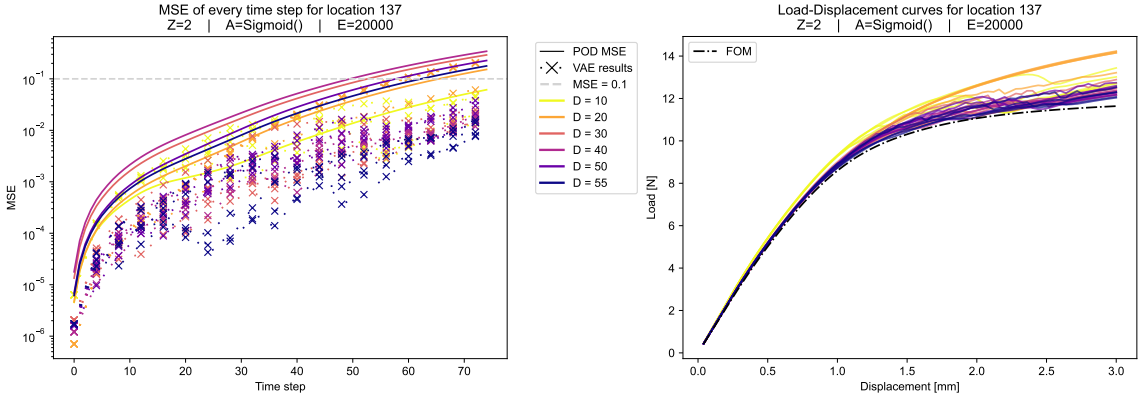
(a) **(b)**

Figure A.7: Selection of results for location 131.



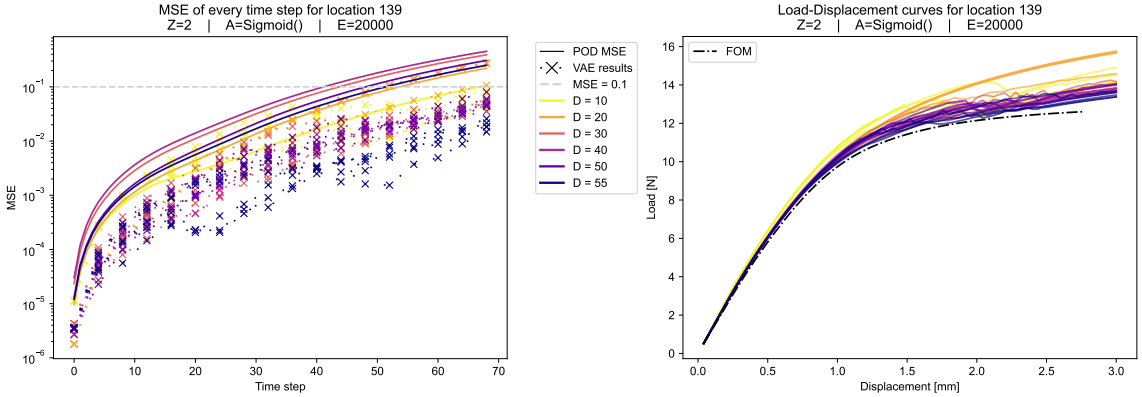
(a) **(b)**

Figure A.8: Selection of results for location 135.



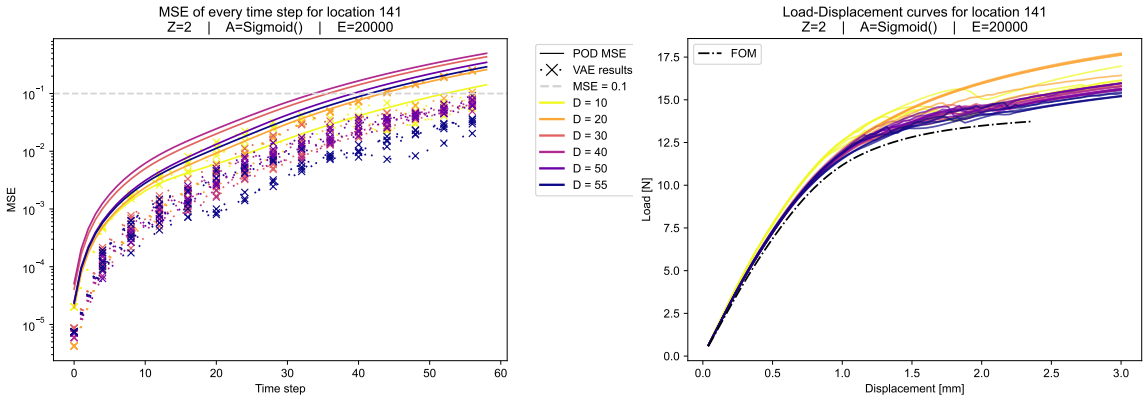
(a) **(b)**

Figure A.9: Selection of results for location 137.



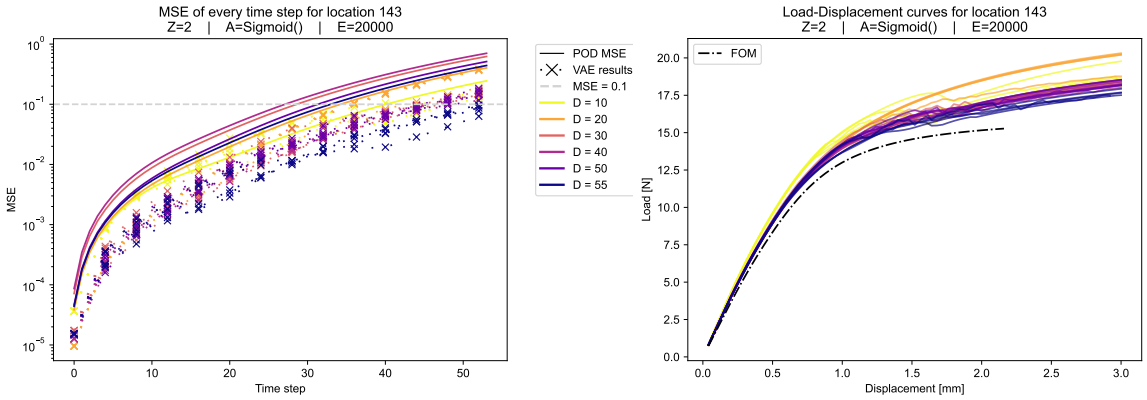
(a) **(b)**

Figure A.10: Selection of results for location 139.



(a) **(b)**

Figure A.11: Selection of results for location 141.



(a) **(b)**

Figure A.12: Selection of results for location 143.

B

ARC LENGTH ALGORITHM

Algorithm 2: Arc length control

Require *Nonlinear relation* $\mathbf{f}_{int}(\mathbf{u})$ *with* $\mathbf{K}(\mathbf{u}) = \frac{\partial \mathbf{f}_{int}}{\partial \mathbf{u}}$, *constraint equation* g
Initialize $n = 0$, $\mathbf{u}^0 = 0$
while $n <$ *number of time steps* **do**
    Initialize new solution at old one: $\mathbf{u}^{n+1} = \mathbf{u}^n$
    Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}(\mathbf{u}^{n+1})$, $\mathbf{K}^{n+1}(\mathbf{u}^{n+1})$
    repeat
        Solve two linear systems of equations: $\mathbf{K}^{n+1}\Delta\mathbf{u}^I = \mathbf{r}$, $\mathbf{K}^{n+1}\Delta\mathbf{u}^{II} = \hat{\mathbf{f}}$
        Solve for load increment: $\Delta\lambda = \dfrac{g + \boldsymbol{h}^T\Delta\boldsymbol{\alpha}^I}{s + \boldsymbol{h}^T\Delta\boldsymbol{\alpha}^{II}}$
        Update solution: $\mathbf{u}^{n+1} = \mathbf{u}^{n+1} + \Delta\mathbf{u}^I + \Delta\lambda\Delta\mathbf{u}^{II}$
        Update load factor: $\lambda^{n+1} = \lambda^{n+1} + \Delta\lambda$
        Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}(\mathbf{u}^{n+1})$, $\mathbf{K}^{n+1}(\mathbf{u}^{n+1})$
        Evaluate residual: $\mathbf{r} = \lambda^{n+1}\hat{\mathbf{f}} - \mathbf{f}_{\text{int}}^{n+1}$
    until $|\mathbf{r}| <$ *tolerance*;
    $n = n + 1$