PhD-FSTM-2022-049
The Faculty of Science, Technology and Medicine

Faculty of Applied Sciences
Department of Aerospace and Mechanical Engineering

DISSERTATION

Defence held on 25/04/2022 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG
EN SCIENCES DE L'INGÉNIEUR

AND

DOCTEUR DE L'UNIVERSITÉ DE LIÈGE
EN SCIENCES DE L'INGÉNIEUR

by

Soumianarayanan Vijayaraghavan
Born on 30 January 1992 in Chennai, India

MACHINE LEARNING FOR PROJECTION-BASED MODEL-ORDER-REDUCTION OF ELASTOPLASTICITY

Dissertation defence committee

Prof. Andreas Zilian, Chairman
Professor, Université du Luxembourg

Dr Ling Wu, Vice-Chairwoman
Research Scientist, Université de Liège

Prof. Stéphane P.A. Bordas, Dissertation Supervisor
Professor, Université du Luxembourg

Prof. Ludovic Noels, Dissertation Co-Supervisor
Professor, Université de Liège

Prof. David Ryckelynck, Member
Professor, Mines ParisTech

Prof. Christian Duriez, Member
Professor, INRIA-Lille

Machine learning for projection-based model-order-reduction of elastoplasticity

Thesis by
Soumianarayanan Vijayaraghavan

In partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

UNIVERSITY OF LUXEMBOURG
Esch-sur-Alzette, Luxembourg

and

UNIVERSITY OF LIÈGE
Liège, Belgium

April 2022

ABSTRACT

Projection-based model-order-reduction (MOR) accelerates computations of physical systems in case the same computation must be performed many times for different load parameters (e.g. parameters, geometries, initial conditions, boundary conditions). It therefore finds its use in application domains such as inverse modelling, optimization, uncertainty quantification and computational homogenization. Projection-based MOR uses the solutions of an initial set of (training/offline) computations to construct the solutions of the remaining (online) computations. For finite element computations of hyperelastic solids, projection-based MOR is accurate and fast. However, for finite element computations of hyperelastoplastic solids, conventional projection-based MOR is far from accurate and fast. This thesis explores different numerical approaches to improve projection-based MOR for hyperelastoplastic finite element simulations. The first investigated innovation focuses on enhancing the interpolation employed in projection-based MOR with an additional interpolation associated with a coarse finite element discretization. Because inconsistent results are obtained with this approach, the second innovation focuses on equipping the projection-based MOR with a neural network. This substantially accelerates the online computations, and although the reported accuracy can be argued to be reasonable, it is definitely not excellent. To this end, the third innovation investigates the use of machine learning to adaptively select the interpolation functions of projection-based MOR during the course of a simulation.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to everyone who has provided their assistance, guidance and support over the duration of my PhD.

First, I would like to thank Prof.Stephane Bordas for providing me the opportunity to pursue my doctoral study. In-spite of his busy schedule, he was always available for discussions, and paved a great path for my research work and also for my career. I thank prof.Bordas for his patience in listening to my problems and uplifting me a numerous times. I appreciate the informality of our relationship leading to a relaxed working environment.

I would like to thank Prof.Ludovic Noels for all his support and valuable suggestions. Every meeting with Prof.Noels has lead to an improvement in my PhD.

I am extremely grateful to my daily supervisor Dr.Lars Beex, without him my PhD thesis would have not been possible. Dr Beex guided me in every step of the research. I am thankful for his patience in clearly teaching me the technical concepts and also for his outstanding ideas, that has tremendously contributed my thesis. I greatly acknowledge his humbleness in correcting all my mistakes in both programming and writing.

I would also like to thank Dr Ling Wu for her kind help in Neural Network implementation, and other fruitful discussions on machine learning concepts. I am grateful to Prof. Andreas Zilian for accepting to be the chair of my defence. I greatly thank the external examiners Professors David Ryckelynck and Christian Duriez for the time they spent to review my thesis.

I thank my office mates Diego, Chris and Raphael for all the office ideas and support. I especially thank Hussein, for guiding me through all the annoying questions and extremely supporting me both personally and professionally. I would also like to thank Nandha, Juan and Kevin for the amazing office support I had at the University of Liege.

I would like to thank my parents for their extreme support and love, without their understanding and sacrifices, I would have not reached the stage I am right now. Finally, I thank my wife, Aishwarya, for her constant support, especially during the last year of my doctoral study.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Model-order-reduction

Model-order-reduction (MOR) encompasses numerical methods that accelerate time-consuming computations. Whereas surrogate models such as response surfaces and Kriging replace the input-output relation of the computation of interest by a fast alternative, MOR only modifies the computation in order to accelerate it. Consequently, more results often remain available for post-processing using MOR, whereas surrogate models only provide the output for which they are trained [83]. In the field of numerical predictions of physical systems such as finite element (FE) simulations, two MOR categories may be distinguished: *a posteriori* methods and *a priori* methods.

A posteriori MOR involves precomputing numerous responses of the physical system of interest in advance and utilizing the precomputed solutions to accelerate the subsequent simulations that remain. *A posteriori* MOR is thus only useful if the same physical system must be simulated numerous times, each time with different load parameters (e.g. material parameters, boundary conditions, geometries). Consequently, *a posteriori* MOR finds its use in applications such as inverse modelling, optimization, uncertainty quantification and computational homogenization.

On the other hand, *a priori* MOR does not require any precomputations to be performed and hence, it can be used the very first time the physical system of interest is simulated. Consequently, *a priori* MOR is more widely applicable than *a posteriori* MOR, since *a posteriori* MOR is only useful if the same type of computation must be performed numerous times. On the other hand, *a priori* MOR often yields smaller accelerations.

One type of *a priori* MOR is proper generalized decomposition [57, 15]. It enriches the approximation of the computation's solution per iteration, thereby approaching the exact result as more iterations are computed. Another type of *a priori* MOR is the quasicontinuum method, which superimposes a finite element interpolation, including associated quadrature points (i.e. 'summation' or 'sampling' in quasicontinuum terminology, or 'hyperreduction' in projection-based MOR terminology) over an atomistic [5], spring [6] or beam [14] lattice. Clearly, *a posteriori* MOR and *a priori* MOR both have their pros and cons.

1.2 Projection-based model-order-reduction: interpolation & hyperreduction

This thesis focuses on *a posteriori* MOR such as the proper-orthogonal-decomposition (POD) method [42, 43, 52, 11] and the reduced-basis (RB) method [59, 60]. Both approaches involve an 'offline' training stage, from which solutions are harvested to construct the solutions of the future computations (i.e. in the 'online' stage). The difference between the POD method and the RB method concerns the manner in which the training solutions are handled to construct the solutions of future computations. In the POD method, the precomputed solutions are decomposed using singular value decomposition (SVD) in order to extract the most dominant characteristics of the training solutions. In the method of RB on the other hand, precomputed training solutions are directly employed (after orthonormalisation) to construct the solutions of future computations. Similar as in the POD method, the ensemble of selected training solutions in the RB method should enclose the characteristic features of the possible solutions of future simulations. This is often performed using a greedy algorithm. In both the POD and RB method, the precomputed solutions yield orthonormalized vectors which together reconstruct the solutions of future computations. These vectors are referred to as *basis functions* or *modes*.

In projection-based MOR such as the POD and RB method, each basis function comes with its own degree of freedom (i.e. 'coefficient'), all of which must be (simultaneously) computed during an online computation. Consequently, it is essential for the speed of the online computations to minimize the number of employed basis functions and hence, the number of degrees of freedom. In the case of non-linear computations however, besides the time needed to *solve* the linearized governing equations, another bottleneck is present: the time to *construct* the linearized governing equations. In implicit non-linear computations based on Newton's method, the linearized governing equations (i.e the column with first-order derivatives and the matrix with second-order derivatives) must be constructed for each iteration, for each increment. Whereas the interpolation of the basis functions reduces the number of equations and hence, the time to solve the systems of linear equations, it does not reduce the time to construct the linearized governing equations.

Approaches to reduce the time to construct the linearized governing equations in projection-based MOR are often referred to as 'hyperreduction'. Hyperreduction involves the sampling of quantities at only a limited number of spatial locations to approximate the first-order and second-order derivatives of the system of interest.

These quantities may be the derivatives themselves, as in the discrete empirical interpolation method [12, 13, 58, 62], or quadrature points [68, 66, 24, 53, 35].

1.3 Elasticity versus elastoplasticity

Projection-based MOR using interpolation and hyperreduction as described above has successfully been used to accelerate FE simulations of hyperelastic solids [40, 56]. The use of only a small number of basis functions yields excellent results for hyperelastic FE computations. Consequently, the associated systems of linear equations are small and fast to solve. Hyperreduction using the discrete empirical interpolation method is furthermore highly accurate using only a small number of sampled derivatives, thanks to the spatially smooth fields of the derivatives associated with (hyper)elasticity.

Projection-based MOR using interpolation and hyperreduction as described above is substantially less trivial to accelerate FE simulations of hyperelastoplastic solids [29]. The reason is twofold. First, elastoplastic FE computations require many more basis functions to obtain a similar accuracy as for hyperelastic FE computations. This is illustrated in Fig. 1.1, in which the same FE model is considered with a hyperelastic and a hyperelastoplastic material description. The singular values decay substantially faster for hyperelasticity than for hyperelastoplasticity. Because many more basis functions are required for elastoplasticity, the number of equations (i.e. of degrees of freedom) remains relatively large and hence, solving the system of linear equations (once per iteration) requires more time.

Second, hyperreduction is substantially less trivial to be effectively exploited. This has two causes. First, the fact that many basis functions are required entails that more quantities need to be sampled to accurately approximate the linearized governing equations. Second, elastoplasticity entails that the spatial fields of derivatives and second-order derivatives are non-smooth and the spatial domain in which smoothness is lacking changes for each online computation.

The first aim of this thesis is therefore to devise a projection-based MOR for elastoplasticity such that substantially less basis functions are required. The intent is not necessarily to reduce the number of degrees of freedom, but to ensure that hyperreduction can be applied more effectively, i.e. to ensure that less quantities need to be sampled to approximate the linearized governing equations. Reducing the number of basis functions is investigated in this thesis by using the additional interpolation of an additional, rather coarse FE discretization. Since each basis function interpolates over the entire domain, and the additional FEs only have

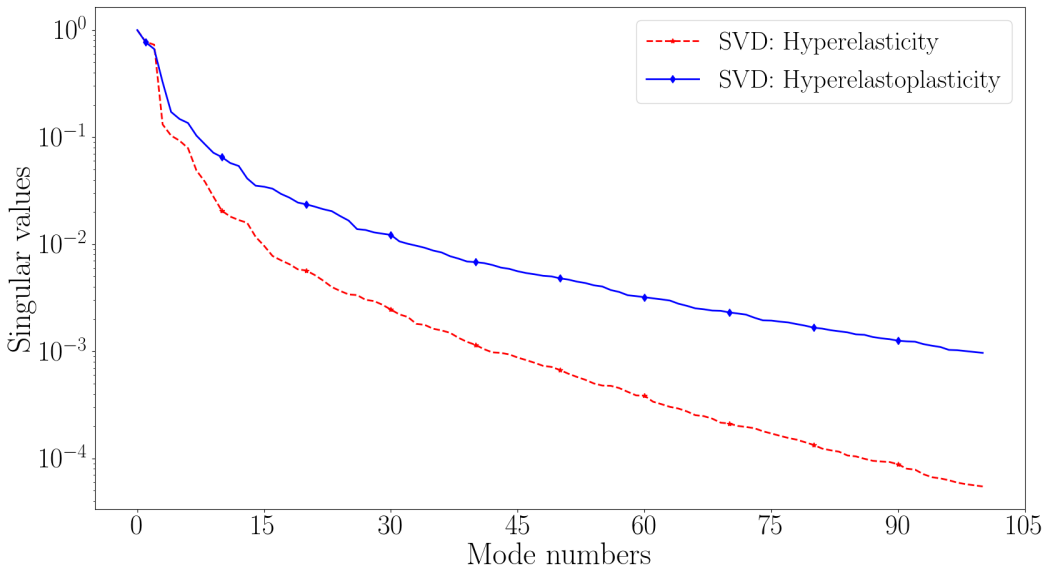


Figure 1.1: The singular values for two the same quasi-static FE computations: one with a hyperelastic constitutive model (red curve) and one with a hyperelastoplastic constitutive model (blue curve). Except for the constitutive model, all other aspects are the same (e.g. mechanical parameters, geometry, FE discretization, boundary conditions). The decay of the singular values is substantially faster for the hyperelastic configuration than for the hyperelastoplastic configuration. This indicates that substantially less basis functions (i.e. modes) need to be used for the hyperelastic case than for the hyperelastoplastic case if projection-based ROM is to be applied.

local support, the interpolation of the basis functions is referred to as the *global* interpolation and that of the additional FE discretization as the *local* interpolation.

1.4 Neural network acceleration of projection-based model-order-reduction

Because the local/global MOR approach mentioned above does not yield consistent results (as will be explained in detail in the next chapter), another projection-based MOR approach is also exploited and developed in this thesis. The ansatz of this approach is to let a neural network emulate the coefficients of the basis functions for each increment. This implies that no iterative solution process is required to compute the basis function coefficients. This not only avoids the need to solve a system of linear equations (once per iteration), it also avoids the construction of the matrix with second order derivatives (i.e. the stiffness matrix) and the column with first order derivatives (i.e. the force column) per iteration. The only issue that remains is the construction of the force column once per increment. Because constructing the force column once per increment does not require much time, hyperreduction does not need to be applied to achieve a signifant acceleration.

Projection-based MOR with neural networks to rapidly predict the basis function coefficients is not new. Several of such frameworks were developed for fluid problems [45, 81, 7], just like the first projection-based MOR to be developed [73]. Also in the field of solid mechanics, such frameworks have recently been proposed for hyperelastic computations [80, 9, 17]. However, no such approaches have been constructed for projection-based MOR for elastoplastic FE computations.

The second aim of this thesis is therefore to devise a projection-based MOR for elastoplasticity where the basis function coefficients are emulated by a neural network. The scientific challenges are to ensure that the framework can accurately treat the path-dependency of elastoplasticity and that appropriate input parameters are selected for the neural network. As will become clear later in the thesis, the treatment of the path-dependency in the framework is strived by using *recurrent* neural networks as the type of neural network.

Because the number of basis function coefficients has not much influence on the computational times of the aforementioned (neural-network-accelerated) projection-based MOR, a large number can be used. However, even if a large number of 100 basis functions is employed, the accuracy of the neural-network-accelerated projection-based MOR is acceptable, but not perfect. This relative inaccuracy does not originate from the neural network, but from the projection-based MOR itself, which is also without neural network acceleration relative inaccurate due to the non-ellipticity of the partial differential equations (see Fig.1.1 again).

1.5 Machine learning for adaptive basis selection

To ensure that projection-based MOR itself is more accurate, several studies have proposed to group the training solutions in clusters (based on some appropriate quantities). In such approaches, one set of basis functions is identified for each cluster independently. During the course of an online computation, the same quantity that was used to group the training solutions parametrises the current configuration (i.e. increment) and is consequently used to decide from which cluster the set of basis functions must be used.

In [19, 21], the authors themselves grouped the training computations in clusters based on the time. Since time also elapses during the course of the online computations, a different set of basis functions is used during the online computations, yielding an approach in which the basis functions are adaptively changed during an online simulation.

Although in some cases the clustering for adaptive selection of the basis function can be performed manually, it is inaccurate and impractical in most scenarios: for instance if several quantities must be used (instead of just one like the time), and/or if some manifold is more suited than an Euclidian space. For this reason, machine learning has recently been proposed to perform the clustering; in particular unsupervised learning [79, 2, 58, 29, 51, 10]. However, no studies have investigated this for elastoplasticity.

The third aim of this thesis is therefore to propose a projection-based MOR for elastoplasticity with an adaptive selection of the basis functions based on machine learning. The start is made with the unsupervised k-means clustering of [79, 2], but as will become clear in Chapter 4, every time the basis functions are changed during the course of a simulation, the results are momentarily highly inaccurate. For this reason, instead of k-means clustering, DBSCAN is also investigated. DBSCAN indeed clusters the training solutions such that less changes of basis functions are needed in case of monotonic elastoplastic simulations, but the significant inaccuracy that occurs when the basis functions are changed remains. For this reason, several approaches are introduced in Chapter 4 which intend to change the basis functions more smoothly, but they do not yield a satisfying accuracy. A new approach is therefore proposed in Chapter 4 based on k nearest neighbour (k-NN) searching. In this approach the basis functions potentially change each iteration. The results are highly accurate for only a few basis functions.

1.6 Aims and innovations

The three aforementioned aims of this thesis are the three main innovations of this thesis and are summarized as follows:

- to formulate a projection-based MOR for elastoplasticity to reduce the number of basis functions by introducing an additional local interpolation using coarse FE discretizations. The ultimate intent is not to reduce the number of degrees of freedom in the online simulations, but to reduce the efforts to approximate the linearized governing equations (i.e. the force column and stiffness matrix).
- to investigate if a neural network is able to emulate the coefficients of the basis functions of projection-based MOR for elastoplasticity, such that both (1) many basis functions can be used, and (2) the neural network can account for the path-dependency of elastoplasticity.

- to investigate if machine learning can be used to adaptively change the basis functions during the course of an online elastoplastic projection-based MOR simulation.

1.7 Outline

The remainder of this thesis consists of four more chapters. Chapter 2 discusses the projection-based MOR in which a reduction of the number of basis functions is strived by combining the global interpolation of the basis functions with an additional interpolation associated with a course FE discretization (Innovation 1). Because the local/global approach of chapter 2 does not yield consistent results, a different approach is the focus of Chapter 3. The approach of Chapter 3 uses neural networks to emulate the basis function coefficients (Innovation 2). To further improve the neural-network-accelerated projection-based MOR of Chapter 3, Chapter 4 discusses machine learning to adaptively select the basis functions during the course of a simulation (Innovation 3). Finally, Chapter 5 presents conclusions and identifies potential avenues for future work.

1.8 Additional notes

It must be noted that projection-based MOR frameworks exist in which large plastic deformations (or damage) occur in a small part of the domain [42, 41, 61]. In those frameworks, MOR is used for the largest part of the domain that deforms elastically and the original FE discretization in the small domain in which dissipation occurs remains in place. This is different from the scenarios considered in this thesis, in which large plastic deformations occur in the entire domain.

It it also worth to note that both the local and global elastoplastic deformations that occur in the simulations considered in this thesis, as well as the fluctuations of these deformations, are considerably larger than those considered in [79]. This poses a larger challenge for projection-based MOR.

The hyperelastoplastic problem to which all the introduced projection-based MOR approaches are applied are 2D periodic representative volume elements, as useful for computational homogenization. However, true multiscale simulations are not performed, amongst others because the approaches are not limited to computational homogenization. In fact, they can in principle be used for any type of application where an elastoplastic model is exposed to uncertain boundary conditions.

The following three chapters in which the scientific and technological work and novelties are described in detail are extracted from three manuscripts that have been, or are in the process of being submitted for publication in peer-reviewed journals. Consequently, some repetition in the chapters is present.

Chapter 2

LOCAL INTERPOLATION TO AID PROJECTION-BASED MODEL-ORDER-REDUCTION FOR ELASTOPLASTICITY

ABSTRACT

Projection-based model-order reduction approaches are accurate and fast for finite element simulations with hyperelastic constitutive models: the use of only a limited number of global basis functions generally provides an excellent accuracy. In contrast to (hyper)elasticity, (hyper)elastoplasticity require many global basis functions to achieve a reasonable accuracy. The fact that many basis functions are required for hyperelastoplasticity is not only problematic because a relatively large number of degrees of freedom remains, it also entails the need for many integration points in the online simulations (the stress update must be iteratively computed at each quadrature point in each iteration). This chapter investigates two approaches to keep the number of global basis functions of finitely plastically deforming finite element simulations small - ultimately in order to keep the number of reduced quadrature points of the hyperreduction small. This is performed by combining the global basis functions with local interpolation functions. For the two approaches combining the global and local interpolation, two manners are investigated to identify the global basis functions. Although the results are clearly improved, a truly consistent trend is lacking for the time being. Hence, several avenues can be taken to improve the frameworks.

2.1 Introduction

Often in engineering practice, the same mechanical model must be computed numerous times, but with different sets of material parameters, geometries, and/or boundary conditions. Examples are optimization problems, inverse problems, forward uncertainty propagation, and nested multiscale approaches. To speed up the computations of such problems, one can revert to a computational approach that rapidly emulates the output parameters of interest. Response surface models are one such approach that has been studied persistently [76, 31, 30]. Neural networks, currently popular in computational mechanics [83, 26, 36, 69, 70], can also be used to rapidly predict the required output. One may also employ *a priori* [57, 15] or *a posteriori* [8] model-order-reduction (MOR).

Each of these approaches comes with its own advantages and disadvantages. The construction of response surface models is for instance not trivial if the Euclidean space is insufficient to formulate a response surface. Neural networks are fast at the 'online/prediction stage', but require many 'offline training simulations' (i.e. direct numerical simulations - DNS) to achieve a good accuracy in the online stage. *a posteriori* MOR has the advantage that many local results of the online simulations remain available, whilst complex geometries and periodic boundary conditions can easily be treated. A disadvantage is that offline training simulations are required.

This chapter focuses on *a posteriori* projection-based model order reduction. Projection-based MOR utilizes solutions of so-called training simulations to construct global basis functions in order to interpolate kinematic variables. They use either a representative set of orthonormalized training solutions as the basis functions (i.e. the method of 'reduced basis' [59, 60]), or apply singular value decomposition to the training solutions, and use the left-singular vectors associated with the highest singular values as basis functions (i.e. the method of 'Proper Orthogonal Decomposition' - POD. [48, 42, 43, 52, 11]).

The global basis functions in POD-based MOR interpolate the kinematic variables to reduce the number of degrees of freedom in the online simulations. This accelerates the process to *solve* the governing equations, but it does not reduce the time required to *construct* the governing equations - which is important for non-linear models, as the governing equations must be solved and constructed numerous times. In those cases, to alleviate the computational burden, a reduced set of quadrature points is selected (directly or indirectly, often referred to as 'hyperreduction' [38, 12, 67, 66, 35]).

Projection-based MOR only needs a limited number of basis functions for finite element computations of hyperelastic solids [62]). However, for simulations of elastoplasticity, a high number of basis functions is required to obtain an acceptable accuracy in the online simulations. The difference in their working can be deduced from Fig. 1.1, in which the largest singular values are presented for the same finite element problem described by hyperelasticity and by hyperelastoplasticity. The singular values clearly decay substantially faster for hyperelasticity than for hyperelastoplasticity.

The requirement of a large number of basis functions for hyperelastoplasticity has not only the disadvantage that the number of degrees of freedom (DoFs) must remain relatively large to achieve an acceptable accuracy, it also increases the required number of reduced quadrature points (at which the stress update must be iteratively computed, i.e. the efficiency of the hyperreduction). The increase of the number of reduced quadrature points for an increase of the number of basis functions is illustrated in Fig. 2.1, in which the hyperreduction strategy of [35] is applied to the elastoplastic model of interest in this thesis. The explanation for the increase of the number of reduced integration points for an increase of the number of global basis functions is that the spatial fluctuations in consecutive basis functions increase.

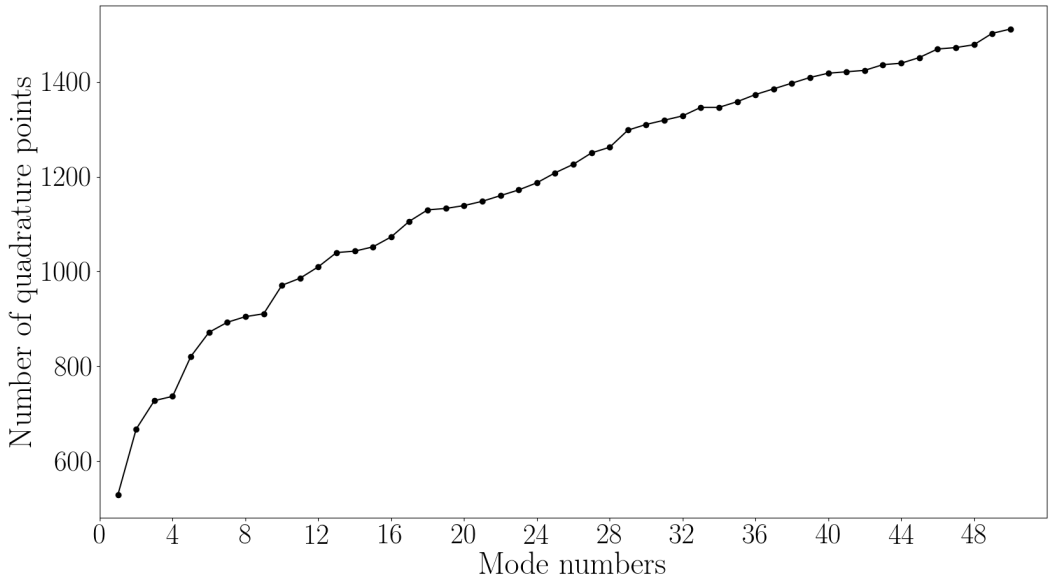


Figure 2.1: The number of required (reduced) quadrature points as a function of the number of global bases for the hyperreduction of [35].

Hence, the number of basis functions for hyperelastoplasticity must remain relatively small in order to substantially reduce the number of reduced integration points. Therefore, this chapter aims to reduce the number of basis functions, by binding together the global interpolation of the conventional basis functions with a local interpolation.

The global/local interpolation schemes are similar to the approach in [20], but with an important difference. In the current chapter, the local and global interpolations are combined additively, whereas [20] uses them multiplicatively. The issue with the multiplicative use of the two interpolations [20] is that the effective basis functions change from iteration to iteration (and from increment to increment). This results in a limited reduction of the number of quadrature points if the hyperreduction of [35] is employed (which is our final goal, although not treated in the current chapter).

It is also worth noting that the local/global approaches are different from the substructuring method in [61]. [61] employs a spatially concurrent projection-based MOR approach to distinguish between regions with elastoplastic deformation (where the original FE interpolation is employed) and regions with purely elastic deformation (where the global interpolation of projection-based MOR is used to reduce the number of DoFs). In the current chapter however, elastoplastic deformation occurs in the entire modeling domain.

In this chapter, two local/global interpolations are investigated. They are applied to a hyperelastoplastic representative volume element (RVE) with stiff elastic particles, exposed to monotonic loading. In the next section, the DNS are concisely discussed. Section 2.3 describes a standard POD-based MOR, and the two local/global interpolation approaches are discussed in section 2.4. The results are presented in section 2.5, where the results of the two local/global interpolation schemes are compared with each other and with those of a conventional POD-based MOR for different numbers of global basis functions and different local refinements. A short conclusion and outlook are presented in section 2.6.

2.2 Direct Numerical Simulations

The plane strain simulations employ bilinear quadrilateral (four node) finite elements with four Gauss quadrature points. An F-bar method is utilized to alleviate locking due to the incompressibility of the plastic deformation. In the employed F-bar method, the volume change of the deformation gradient tensor at a quadrature point is replaced with the volume change at the center of the element. The result-

ing deformation gradient tensor, $\bar{\mathbf{F}}$, is multiplicatively decomposed into an elastic (subscript e) and a plastic (subscript p) deformation gradient tensor: $\bar{\mathbf{F}} = \mathbf{F}_e \cdot \mathbf{F}_p$.

The following strain energy density is employed:

$$W = \frac{E(I_e - 3 - 2\ln(J_e))}{4(1+\nu)} + \frac{E\nu(\ln(J_e))^2}{2(1+\nu)(1-2\nu)}, \tag{2.1}$$

where E and ν denote Young's modulus and Poisson's ratio, respectively. Furthermore: $I_e = \mathrm{tr}(\mathbf{F}_e^T \cdot \mathbf{F}_e)$ and $J_e = \det(\mathbf{F}_e)$, where superscript T denotes the transpose. Differentiating the strain energy with respect to $\mathbf{F}_e$ gives a 1st Piola-Kirchhoff stress tensor, $\mathbf{P}_e$: $\mathbf{P}_e = \frac{\partial W}{\partial \mathbf{F}_e}$, which is related to the Mandel stress, $\mathbf{M}$, as $\mathbf{M} = \mathbf{F}_e^T \cdot \mathbf{P}_e$.

The employed yield function reads:

$$y = \sqrt{\frac{3}{2}\mathbf{M}^{dev} : \mathbf{M}^{dev}} - M_0 - h\,\lambda^n, \tag{2.2}$$

where material parameters M_0, h and n denote the initial yield stress, the hardening modulus and an exponential hardening parameter, respectively. Furthermore, $\mathbf{M}^{dev}$ denotes the deviatoric Mandel stress and λ the plastic multiplier. The following associated flow rule is employed:

$$\dot{\mathbf{F}}_p = \dot{\lambda}\frac{\partial\, y}{\partial \mathbf{M}} \cdot \mathbf{F}_p. \tag{2.3}$$

The Karush-Kuhn-Tucker conditions close the constitutive model:

$$y \leq 0, \qquad \dot{\lambda} \geq 0, \qquad y\dot{\lambda} = 0. \tag{2.4}$$

A periodic mesh is employed in the simulations, Dirichlet boundary conditions are used for the four corner nodes, where the displacement values are dictated by the right stretch tensor of the macroscale deformation ($\mathbf{U}^M$, assuming that the RVE is used in a nested multiscale computation), given by:

$$\mathbf{u}_j - \mathbf{u}_i = \left(\mathbf{U}^M - \mathbf{I}\right) \cdot \left(\mathbf{X}_j - \mathbf{X}_i\right), \tag{2.5}$$

where $\mathbf{u}$ and $\mathbf{X}$ denote the displacement vector and reference location of a finite element node, respectively. The subscripts denote the numbers of two corner nodes.

As the displacement of one of the four corner nodes is set to zero (and all reference locations are known), the displacement vectors of the other three corner nodes are completely known, since $\mathbf{U}^M$ is known for each increment.

In case of nodes on the RVE's opposing edges, the above vector equation yields two scalar constraints in a 2D setting (as is the case here). In this thesis, these constraints are enforced using Lagrange multipliers.

The incorporation of periodic boundary conditions using Lagrange multipliers results in the following system of linear equations, which must be constructed and solved for each iteration, at each increment:

$$\begin{bmatrix} \underline{\underline{K}}_{\text{int}}(\underline{u},\underline{z}) & \left(\frac{\partial \underline{\underline{c}}}{\partial \underline{u}}\right)^T \\ \frac{\partial \underline{\underline{c}}}{\partial \underline{u}} & \underline{\underline{0}} \end{bmatrix} \begin{bmatrix} d\underline{u} \\ d\underline{g} \end{bmatrix} = \begin{bmatrix} \underline{f}_{\text{ext}} - \underline{f}_{\text{int}}(\underline{u},\underline{z}) - \underline{g}^T \frac{\partial \underline{c}}{\partial \underline{u}} \\ \underline{c}(\underline{u}) \end{bmatrix}, \tag{2.6}$$

where column $\underline{u}$ collects the displacement components of all FE nodes at an intermediate solution, column $\underline{z}$ the plastic variables at all Gauss quadrature points at an intermediate solution (λ and $\mathbf{F}_p$), column $\underline{g}$ the Lagrange multipliers, column $\underline{c}$ the scalar constraints due to the periodic boundary conditions of Eq. (4.5) ($\underline{c}$ is linear in $\underline{u}$), column $\underline{f}_{\text{ext}}$ the components of the reaction forces, column $\underline{f}_{\text{int}}$ the components of the internal forces ($\underline{f}_{\text{int}}$ depends non-linearly on $\underline{u}$ and $\underline{z}$) and matrix $\underline{\underline{K}}_{\text{int}}$ the derivatives of the internal forces components with respect to the displacement components ($\underline{\underline{K}}_{\text{int}}$ depends non-linearly on $\underline{u}$ and $\underline{z}$). $d\underline{u}$ and $d\underline{g}$ together denote the correction to the intermediate solution given by $\underline{u}$ and $\underline{g}$, that is to be computed each iteration. The new plastic variables are computed together with the new internal forces for each quadrature point, after new solution $\underline{u} + d\underline{u}$ is computed.

2.3 POD-based model order reduction

One of the computational challenges of the DNS is the large number of DoFs involved in the systems of equations (2.6) that need to be computed for each iteration, for each increment. Projection-based MOR overcomes this computational drawback by substantially reducing the number of equations and hence, the number of DoFs. In this thesis, we consider *a posteriori* projection-based MOR, which works based on the idea of precomputing solutions in advance and use the precomputed solutions to speed up the simulations in a later stage.

The precomputations are performed in an offline training stage, during which the parameter-dependent DNS (the parameters in this chapter are formed by the paths

of $\mathbf{U}^M$), is solved for numerous parameter sets of interest. Projection-based MOR assumes that the various precomputed DNS solutions, obtained for different parameters, can be well represented in a lower-dimensional subspace.

To reduce the number of equations and hence, the number of DoFs, the first stage of projection-based MOR is to identify the basis functions ($\underline{\underline{\Phi}}$) that capture the lower dimensional subspace. In the online stage subsequently, all n_u kinematic variables $\underline{u}$ are interpolated using the identified n_b global basis functions according to:

$$\underline{u} \approx \sum_{i=1}^{n_b} \underline{\phi}_i \alpha_i = \underline{\underline{\Phi}}\,\underline{\alpha}, \tag{2.7}$$

where $\underline{\phi}_i$ of length n_u denotes the i^{th} basis function and scalar α_i denotes its associated coefficient that is to be computed for each increment in an online simulation. $\underline{\underline{\Phi}}$ and $\underline{\alpha}$ collect all the global basis functions and their associated coefficients in an $n_u \times n_b$ matrix and a column of length n_b, respectively.

In the subsequent part of this section we discuss the identification of the global basis functions ($\underline{\underline{\Phi}}$) in the offline stage, using the proper orthogonal decomposition method, one of the variants of projection-based MOR. Subsequently, the computation of the associated coefficients ($\underline{\alpha}$) in the online simulations is discussed.

Offline stage: Construction of the global basis

In POD-based MOR, the global basis functions are identified using the method of snapshots introduced in [73]. In this method, training simulations are performed using the DNS for different sets of parameter values. Incremental solutions are stored in a matrix referred to as a snapshot matrix ($\underline{\underline{S}}$) of size $n_u \times n_t$, where n_t denotes the number of training solutions. Therefore, a column in matrix $\underline{\underline{S}}$ corresponds to an incremental solution of a training simulation and one is obviously not obliged to store the solution of all increments.

To obtain $\underline{\underline{\Phi}}$, the snapshot matrix ($\underline{\underline{S}}$) is decomposed into three matrices according to singular value decomposition:

$$\underline{\underline{S}} = \underline{\underline{U}}\,\underline{\underline{\Sigma}}\,\underline{\underline{V}}^T, \tag{2.8}$$

where matrices $\underline{\underline{U}}$ (of size $n_u \times n_t$) and $\underline{\underline{V}}$ (of size $n_t \times n_t$) are the left and the right singular vectors of $\underline{\underline{S}}$, which are orthonormal with respect to each other. Matrix $\underline{\underline{\Sigma}}$

is a diagonal matrix (of size $n_t \times n_t$), with singular values arranged in a decreasing order ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{n_t} \geq 0$). The n_b left singular vectors in $\underline{\underline{U}}$ that correspond to the largest singular values are chosen as the POD basis functions.

An important question is how many basis functions are required to use in the online simulations. On the one hand, one desires to maximize the acceleration of the online simulations, limiting the number of basis functions in order to minimize the number of DoFs. On the other hand, one desires a high accuracy by incorporating a sufficiently large number of basis functions.

A traditional approach to decide the number of basis vectors is based on the singular values. The reason to use singular values as a measure is that, according to the Schmidt-Eckart-Young theorem [1, 22], if the first n_b left singular vectors are considered as basis functions, the projection error of the basis on the training solutions can be evaluated using the $(n_b + 1)^{\text{th}}$ to n_t singular values, given as:

$$\min_{\Phi \in \mathbf{R}^{n_u \times n_b}} \sum_{i=1}^{n_t} \| \underline{s_i} - \underline{\underline{\Phi}}\,\underline{\underline{\Phi}}^T \underline{s_i} \|_2^2 = \sum_{i=n_b+1}^{n_t} \sigma_i^2, \tag{2.9}$$

where $\underline{s_i}$ denotes the i^{th} training solution (i.e. the i^{th} column of $\underline{\underline{S}}$) and $\|.\|_2$ denotes the L^2-norm. It can be inferred from Eq. (2.9) that the sum of squares of singular values, corresponding to the left singular vectors that are not included in the basis, represents the square of the error in the snapshot representation. Therefore, an optimal number of basis functions (n_b) are chosen such that the following error measure (ν) is sufficiently low [42]:

$$\nu_{POD}^2 = \frac{\sum_{i=n_b+1}^{n_t} \sigma_i}{\sum_{i=1}^{n_t} \sigma_i}. \tag{2.10}$$

It is also worth to note that in case $n_u \gg n_t$, a faster way to identify the basis functions is by the application of eigenvalue decomposition. to $\underline{\underline{\hat{S}}} = \underline{\underline{S}}^T \underline{\underline{S}}$ ($\underline{\underline{\hat{S}}}$ is symmetric and of size $n_t n_s \times n_t n_s$), which may be expressed by finding eigenvalues b^i and eigenvectors $\underline{b}^{*i}$ according to:

$$\left(\underline{\underline{\hat{S}}} - b^i \underline{\underline{I}} \right) \underline{b}^{*i} = 0, \tag{2.11}$$

$$\underline{b}^{*iT}\underline{b}^{*j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} = 0, \end{cases} \tag{2.12}$$

where $b^i > b^{i+1}$ and $\underline{\underline{I}}$ is of size $n_t n_s \times n_t n_s$. The matrix with the left singular values can then be computed as:

$$\underline{\underline{\Phi}} = \underline{\underline{S}} \left[\underline{b}^{*1} \quad \underline{b}^{*2} \quad \underline{b}^{*3} \quad \ldots \quad \underline{b}^{*n_b} \right], \tag{2.13}$$

where the eigenvalues of the associated eigenvectors are ordered according to $b^1 > b^2 > \ldots > b^{n_b}$. It may be clear that eigenvalue decomposition is so important that, similarly as for singular value decomposition, practically all numerical software packages have their own dedicated functions to efficiently compute it.

In RVE simulations with periodic boundary conditions, the global basis functions are not directly extracted from the training solutions. Rather, each training solution is additively decomposed into a homogeneous contribution, $\bar{\underline{u}}$, and a fluctuating contribution, $\tilde{\underline{u}}$:

$$\underline{u} = \bar{\underline{u}} + \tilde{\underline{u}}, \tag{2.14}$$

which is graphically illustrated in Fig. 2.2. Because the right stretch tensor of the macroscale deformation, $\mathbf{U}^M$, is the input for RVE simulations in multiscale computations, homogeneous displacement field $\bar{\underline{u}}$, is known and does not need to be computed. The global basis functions are thus only used to interpolate the fluctuating part of the kinematic variables, yielding the following expression:

$$\underline{u} = \underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha}, \tag{2.15}$$

where column $\underline{\beta}$ contains the known components of $\mathbf{U}^M$ (three in 2D simulations) and matrix $\underline{\underline{\Psi}}$ (of size $n_u \times 3$ in 2D) homogeneously interpolates the kinematic variables (see Fig. 2.2).

If the bottom-left corner node of the 2D RVE is the first node of the discretization and we set its reference location to the zero vector and consider it not to displace during a simulation, we express $\underline{\beta}$ as follows:

$$\underline{\beta} = \begin{bmatrix} U^M_{xx} - 1 & U^M_{xy} & U^M_{yy} - 1 \end{bmatrix}^T, \tag{2.16}$$

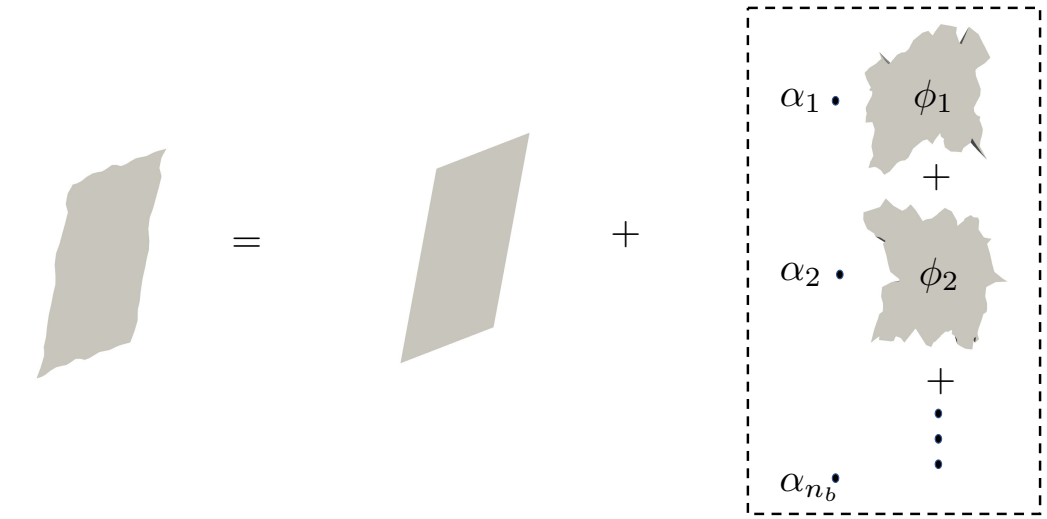


Figure 2.2: Illustration of the additive split of an RVE's displacement field in a homogeneous displacement field and a microstructurally fluctuating displacement field employed in a projection-based MOR. The basis functions ($\underline{\underline{\Phi}}$) are only used for the fluctuating displacement field.

with U^M_{xx}, U^M_{xy} and U^M_{yy} denoting the three independent, known components of $\mathbf{U}^M$. If the horizontal and vertical displacement components of all nodes are stored as follows in $\underline{u}$:

$$\left(\underline{u}\right)_i = \begin{cases} u^x_{\lceil \frac{i}{2} \rceil} & \text{if } i \text{ is odd} \\ u^y_{\lceil \frac{i}{2} \rceil} & \text{if } i \text{ is even} \end{cases}, \tag{2.17}$$

where $\lceil \bullet \rceil$ denotes the ceiling, and $u^x_{\lceil \frac{i}{2} \rceil}$ and $u^y_{\lceil \frac{i}{2} \rceil}$ denote the horizontal and vertical displacement components of node $\lceil \frac{i}{2} \rceil$, respectively, then, row i of $\underline{\underline{\Psi}}$ can be expressed as follows:

$$\left(\underline{\underline{\Psi}}\right)_{i,:} = \begin{cases} \begin{bmatrix} X_{\lceil \frac{i}{2} \rceil} & Y_{\lceil \frac{i}{2} \rceil} & 0 \end{bmatrix} & \text{if } i \text{ is odd} \\ \begin{bmatrix} 0 & X_{\lceil \frac{i}{2} \rceil} & Y_{\lceil \frac{i}{2} \rceil} \end{bmatrix} & \text{if } i \text{ is even} \end{cases}, \tag{2.18}$$

where $X_{\lceil \frac{i}{2} \rceil}$ and $Y_{\lceil \frac{i}{2} \rceil}$ denote the horizontal and vertical component of the reference location of node $\lceil \frac{i}{2} \rceil$, respectively.

The additive decomposition of the kinematic variables entails that SVD is not directly applied to the training solutions. Instead, the homogeneous deformations

are first subtracted from the training solutions, and SVD is applied to the resultant, fluctuating displacement field (i.e. $\underline{\tilde{u}}$). The decomposition produces global basis functions that are themselves periodic and hence, periodicity does not need to be actively enforced in the online simulations.

Online stage: Computing the coefficients

The online stage involves computing the solutions of the reduced order model for new parameter values, that were not a part of the training simulations. In this thesis we use the conventional framework of Galerkin projection method to compute the coefficients ($\underline{\alpha}$) of the basis functions, according to which the solution of $\underline{\alpha}$ is defined as:

$$\underline{\underline{\Phi}}^T \underline{R}(\underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha}, \underline{z}) = \underline{0}, \tag{2.19}$$

where the residual $\underline{R} = \underline{f}_{\text{int}}(\underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha}, \underline{z}) - \underline{f}_{\text{ext}}$.

Linearisation of Eq. (2.19) leads to:

$$(\underline{\underline{\Phi}}^T \underline{\underline{K}}_{\text{int}} \underline{\underline{\Phi}}) d\underline{\alpha} = \underline{\underline{\Phi}}^T \underline{f}_{\text{ext}} - \underline{\underline{\Phi}}^T \underline{f}_{\text{int}} - \underline{\underline{\Phi}}^T \underline{\underline{K}}_{\text{int}} \underline{\underline{\Psi}} d\underline{\beta}. \tag{2.20}$$

where update $d\underline{\beta}$ is known. Both $\underline{f}_{\text{int}}$ and $\underline{\underline{K}}_{\text{int}}$ depend on $\underline{u} = \underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha}$ and $\underline{z}$.

It is worth to recall that although the number of DoFs is reduced relative to that of the DNS (from all n_u displacement components in $\underline{u}$ to the n_b coefficients in $\underline{\alpha}$), the stress update (i.e. history variables λ and $\mathbf{F}_p$) must still be computed for all quadrature points that are present in the DNS. The base of this work is built upon incorporating POD with a hyperreduction strategy of [35], which also reduces the number of quadrature points. However, according to the strategy of [35], increasing the number of POD basis functions increases the number of quadrature points quadrature (look back to Fig. 2.1). Therefore, in the following section, we propose two local/global interpolations that aim to reduce the number of global basis functions for finite plasticity but aim to preserve the accuracy in the online simulations.

Remark The number of load parameters are reduced by considering macroscale right stretch tensor $\mathbf{U}^M$ instead of macroscale deformation gradient tensor $\mathbf{F}^M$. The macroscale right stretch tensor and the macroscale deformation gradient tensor are

related via the following multiplicative decomposition:

$$\mathbf{F}^M = \mathbf{R}^M \cdot \mathbf{U}^M, \tag{2.21}$$

where $\mathbf{R}^M$ denotes the macroscale rotation tensor. When $\mathbf{F}^M$ is known, $\mathbf{U}^M$ can be easily determined by applying an eigenvalue decomposition to the macroscale Green's deformation tensor. In the online simulations, macroscale 1st Piola-Kirchhoff stress tensor $\mathbf{P}^M$ can then be determined as follows:

$$\mathbf{P}^M = \mathbf{R}^M \cdot \bar{\mathbf{P}}^M, \tag{2.22}$$

where $\bar{\mathbf{P}}^M$ denotes the macroscale 1st Piola-Kirchhoff stress tensor that is calculated using the POD-based MOR in the online stage (for $\mathbf{U}^M$ instead of $\mathbf{F}^M$).

2.4 Local/Global interpolation approaches

In this section we discuss the two local/global interpolation approaches that are incorporated with the traditional POD-based MOR. The two approaches employ an additional FE interpolation, on top of the DNS' FE mesh (the discretization of this additional interpolation is refereed to as a grid, to distinguish it from the conventional FE mesh of the DNS). The term local refers to the fact that each kinematic variable associated with the interpolation of the additional grid only affects the displacement field in a part of the RVE. On the other hand, the term global is used for the conventional basis functions of the POD-based MOR, as each kinematic variable associated with each MOR basis function affects the displacement field in the entire RVE (except for the RVE's corners).

The difference between conventional POD-based MOR and the proposed enhancements is thus the following. In conventional POD-based MOR, the global basis functions are used to interpolate the entire fluctuating displacement field ($\underline{\tilde{u}}$), whereas the local/global extensions use both the global basis functions, as well as the local interpolation of the additional grid to interpolate the fluctuating displacement field (cf. Fig. 2.4 and Fig. 2.5).

The presence of an additional local interpolation requires several issues to be treated. First, we investigate the effect of orthonormalizing local and global interpolation functions with respect to each other. Second, we make sure that the local interpolation functions are constructed such that periodicity does not need to be actively enforced in the online simulations (similar as for the global interpolation). The main difficulty is the identification of the global basis functions under the presence of the local interpolation.

We do not investigate the most suited locations for the grid nodes. Instead, we mainly stick to regular, rectangular grid discretizations (see ahead to Fig. 2.4).

Scheme I

In the first scheme, the local interpolation is an additional FE interpolation (see discretized picture in Fig. 2.4). The FE interpolation employs bilinear quadrilateral (four node) elements. The discretization in scheme I is performed such that each element of the grid is a square and hence, the additional local interpolation completely disregards the particles.

The interpolation of scheme I can be expressed as follows:

$$\underline{u} \approx \underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha} + \underline{\underline{\Omega}}\,\underline{\gamma}, \tag{2.23}$$

and hence,

$$\underline{\tilde{u}} \approx \underline{\tilde{u}}_G + \underline{\tilde{u}}_L, \tag{2.24}$$

with

$$\underline{\tilde{u}}_G = \underline{\underline{\Phi}}\,\underline{\alpha}, \qquad\qquad \underline{\tilde{u}}_L = \underline{\underline{\Omega}}\,\underline{\gamma}, \tag{2.25}$$

where $\underline{\underline{\Omega}}$ stores each local interpolation function as a column. $\underline{\underline{\Omega}}$ is of size $n_u \times n_L$ where n_L denotes the number of variables associated with the local FE interpolation. $\underline{\gamma}$ of length n_L denotes the variables of the local interpolation, which must be computed online.

Each node of the additional interpolation grid comes with two variables (one for the horizontal direction and one for the vertical direction in 2D as is the case here), but there are exceptions. First, the corner nodes do not come with any variables. This entails that the local interpolation has no influence on the displacement of the corner nodes. Second, two nodes on opposing edges share the same two variables (one for the horizontal direction and one for the vertical direction). This ensures that the displacement field associated with the local interpolation is automatically periodic and hence, periodicity does not need to be actively prescribed in the online simulations.

As we mainly consider grids with the same number of nodes in horizontal and vertical direction (see Fig. 2.4) and if we denote the number of grid nodes in one direction by n_{gr} (and hence, the total number of grid nodes equals n_{gr}^2), the number

of variables (and hence, the width of $\underline{\underline{\Omega}}$ and the length of $\underline{\gamma}$) reads:

$$n_L = 2\left(n_{gr}^2 - 4 - 2\left(n_{gr} - 2\right)\right). \tag{2.26}$$

We construct local interpolation matrix $\underline{\underline{\Omega}}$ as follows. First, we evaluate the shape function of each grid node at the locations of all the nodes of the DNS mesh and store these shape function evaluations in each column of $\underline{\underline{\Omega}}$. (In the process of storing the shape function evaluations, we keep in mind that each node of the grid comes with two variables). For example, consider a 3×3 gird of scheme I depicted in fig. 2.3, the shape functions N_1, N_2, N_3, and N_4 for the local grid nodes 1,2,3, and 4 are computed for all the DNS mesh nodes that falls inside the current local grid element. In order to compute the shape functions, the local locations ξ_l and η_l of each DNS mesh nodes are first identified, and the shape functions are computed as $N_1 = \frac{1}{4}(1-\xi_l)(1-\eta_l)$, $N_2 = \frac{1}{4}(1+\xi_l)(1-\eta_l)$, $N_3 = \frac{1}{4}(1+\xi_l)(1+\eta_l)$, and $N_4 = \frac{1}{4}(1-\xi_l)(1+\eta_l)$.

At this moment, the sum of all components in each row of $\underline{\underline{\Omega}}$ still equals one. Once constructed, we remove the columns of $\underline{\underline{\Omega}}$ that correspond to the corner nodes of the grid. Then, we merge the columns of nodes on opposite edges (so that these nodes indeed share the same variables).

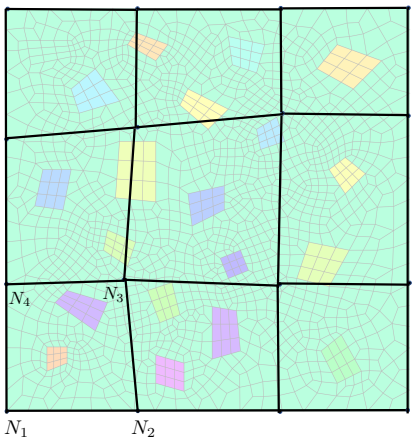


Figure 2.3: Scheme I: Illustration of a 3×3 local grid

Offline stage: Identifying the global basis functions

The identification of the global basis functions is less straightforward than for the conventional POD-based MOR. Several approaches may be used and we investigate two of those.

In the first one, the global basis functions are assumed to capture as much of the fluctuation field as possible and the local interpolation is intended to compensate for the inaccuracies of the global interpolation. In practise, this means that the identification of the global basis functions is exactly the same as for the conventional POD-based MOR of the previous section; first the training solutions are split in homogeneous displacements ($\underline{\bar{u}}$) and fluctuating displacements ($\underline{\tilde{u}}$), and subsequently, the fluctuating displacements are stored in the snapshot matrix ($\underline{\underline{S}}$), from which the global basis functions ($\underline{\underline{\Phi}}$) are extracted using SVD.

In the second approach, the local interpolation captures as much of the fluctuating displacement field as possible, whereas the global basis functions are used to compensate for the inaccuracies of the local interpolation. To this end, we again first split each training solution in a homogeneous displacement field ($\underline{\bar{u}}$) and a fluctuating displacement field ($\underline{\tilde{u}}$). Subsequently, we find the variables of the local interpolation ($\underline{\gamma}$) that best match the fluctuation field of each training solution. This is achieved by solving the following minimization problem:

$$\underline{\gamma}^* = \underset{\underline{\gamma}}{\operatorname{argmin}} \quad \|\underline{\tilde{u}} - \underline{\underline{\Omega}}\,\underline{\gamma}\|_2^2, \tag{2.27}$$

where $\underline{\gamma}^*$ denotes the values of the variables of the local interpolation that best describe the fluctuating displacement field. This minimization problem is solved using Newton's approach, requiring only one iteration as the objective function is quadratic in $\underline{\gamma}$.

The global basis functions are found by storing $\underline{\tilde{u}} - \underline{\underline{\Omega}}\,\underline{\gamma}^*$ of each training solution as a column in matrix $\underline{\underline{S}}$ and applying SVD in the same manner as described in section 2.3.

Online stage: Computing the variables of the local/global basis functions

A relevant change with the traditional POD-based MOR at the online stage is that the DoFs of the new approach involves both the variables ($\underline{\alpha}$) of the basis functions and the variables ($\underline{\gamma}$) of the local interpolation functions. Since both the local and global basis functions are themselves periodic, similar to the global basis functions of conventional POD-based MOR, periodicity is not required to be actively enforced.

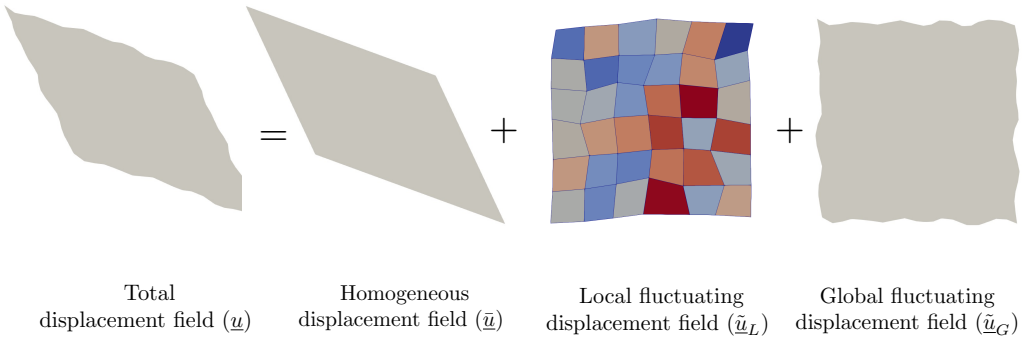


Figure 2.4: Local/Global scheme I: Split of a solution, $\underline{u}$, in to a homogeneous part, $\underline{\bar{u}}$, a locally fluctuating part, $\underline{\tilde{u}}_L$, that is captured by the interpolation of an FE discretization/grid, and a globally fluctuating part $\underline{\tilde{u}}_G$ from which the basis functions are extracted.

For simplicity of notation, the local and global interpolation matrices are stored together as:

$$\underline{\underline{A}} = \left[\underline{\underline{\Phi}} \quad \underline{\underline{\Omega}}\right], \tag{2.28}$$

where $\underline{\underline{A}}$ is thus of size $n_u \times (n_b + n_L)$. The variables of both sets of interpolation functions are gathered in:

$$\underline{a} = \left[\underline{\alpha}^T \quad \underline{\gamma}^T\right]^T. \tag{2.29}$$

Computing the variables using the Galerkin projection framework, leads to the following linearized problem to be solved online:

$$(\underline{\underline{A}}^T \underline{\underline{K}}_{\text{int}} \underline{\underline{A}}) d\underline{a} = \underline{\underline{A}}^T \underline{f}_{\text{ext}} - \underline{\underline{A}}^T \underline{f}_{\text{int}} - \underline{\underline{A}}^T \underline{\underline{K}}_{\text{int}} \underline{\underline{\Psi}} d\underline{\beta}, \tag{2.30}$$

Finally, it is worth mentioning that the conventional POD-based MOR of section 2.3 is recovered if the grid consists of a single element.

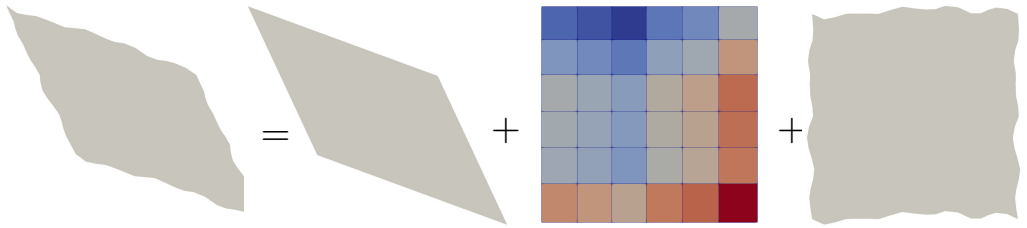
Scheme II

The second local/global interpolation scheme is similar to the scheme I, with the only difference that the grid elements are restricted to deform affinely (i.e. homogeneously) (cf.Fig. 2.4 and 2.5). Restricting the deformation of the grid elements to homogeneous deformation requires the need of an additional matrix in the interpolation, denoted by $\underline{\underline{M}}$ of size $n_L \times n_m$ (where n_m is expressed as given in Eq. (2.33)),

that constrains the deformation of the grid elements to homogeneous deformation. We now express the interpolation as follows:

$$\underline{u} \approx \underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha} + \underline{\underline{\Omega}}\,\underline{\underline{M}}\,\underline{\gamma}, \tag{2.31}$$

where the meaning of the variables in $\underline{\gamma}$ (which is now of length n_m) has changed. In scheme I, the variables in $\underline{\gamma}$ are the DoFs of the grid nodes. In scheme II, the variables are components of the deformation gradient tensors of the grid elements.



Total displacement field ($\underline{u}$) | Homogeneous displacement field ($\bar{\underline{u}}$) | Local fluctuating displacement field ($\tilde{\underline{u}}_L$) | Global fluctuating displacement field ($\tilde{\underline{u}}_G$)

Figure 2.5: Local/Global scheme II: Split of a solution, $\underline{u}$, in to a homogeneous part, $\bar{\underline{u}}$, a locally fluctuating part, $\tilde{\underline{u}}_L$, that is captured by the interpolation of an FE discretization/grid restricted to affine deformations, and a globally fluctuating part $\tilde{\underline{u}}_G$.

The constraints incorporated in scheme II can best be explained using an example, for which we refer the reader to Fig. 2.6, in which a 3×3 interpolation grid is superimposed on the FE mesh of the DNS. In scheme II, each column of grid elements shares the same deformation gradient tensor components F_{xx} and F_{yx} in order to guarantee displacement field compatibility over each column of grid elements. Similarly, each row of grid elements shares the same deformation gradient tensor components F_{yy} and F_{xy} in order to guarantee displacement field compatibility over each row. Finally, the deformation gradient tensor components of the last column and the last row are not variables in order to ensure that the total deformation of the local interpolation is on average the same as the macroscopically applied deformation. In other words, we incorporate the following constraint:

$$\mathbf{U}^M = \frac{1}{n_{EL}} \sum_{i=1}^{n_{EL}} \mathbf{F}^i, \tag{2.32}$$

where $\mathbf{F}^i$ denotes the deformation gradient tensor of the i^{th} grid element. Note that Eq. (2.32) only holds if the reference area/volume of each grid element is the same.

In the 2D cases of this chapter, the number of variables in scheme II, n_m, can thus be expressed in terms of the number of grid nodes in one direction, n_{gr}, as follows:

$$n_m = 4\left(n_{gr} - 2\right). \tag{2.33}$$

Similarly, if we assume that the bottom left corner node of the RVE does not displace and the reference location of the bottom left corner node is set to the zero vector, we can express the meaning of the variables in $\underline{\gamma}$ as follows:

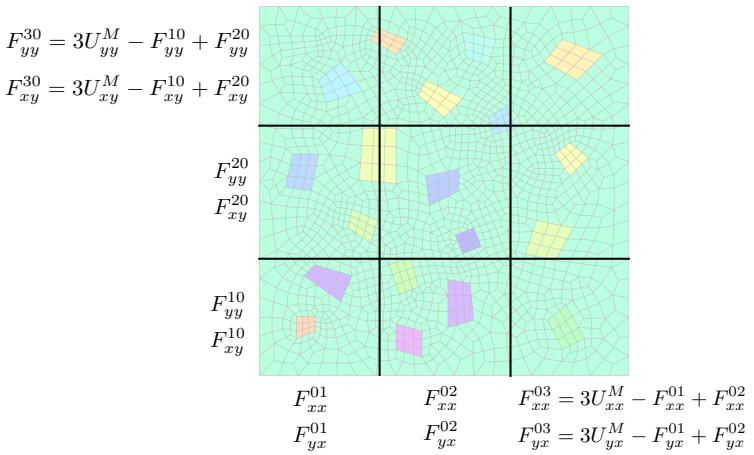


Figure 2.6: Scheme II: Illustration of a 3×3 FE grid with the same number of FEs along the horizontal and vertical directions, also with the same volume (size).

$$\left(\underline{\gamma}\right)_i = \begin{cases} F_{xx}^{0i} - U_{xx}^{M} & \text{if } i \leq n_{gr} - 2 \\ F_{yx}^{(i-n_{gr}-2)0} - U_{xy}^{M} & \text{if } n_{gr} - 2 < i \leq 2n_{gr} - 4 \\ F_{yy}^{(i-2n_{gr}-4)0} - U_{yy}^{M} & \text{if } 2n_{gr} - 4 < i \leq 3n_{gr} - 6 \\ F_{yx}^{(i-3n_{gr}-6)0} - U_{xy}^{M} & \text{if } 3n_{gr} - 6 < i \end{cases}, \tag{2.34}$$

where we refer the reader to Fig. 2.6 for the meaning of the superscripts of the components of $\mathbf{F}$. Associated with the definition of the variables in $\underline{\gamma}$ according to the previous equation, the expression for a component of matrix $\underline{\underline{M}}$ on row i, column j, reads:

$$\left(\underline{\underline{M}}\right)_{i,j} = \begin{cases} H(\hat{X}_{\lceil \frac{i}{2} \rceil} - (j-1)l)l & \text{if } i \text{ is odd } \& \; j \leq n_{gr} - 2 \\ H(\hat{Y}_{\lceil \frac{i}{2} \rceil} - (j-n_{gr}+1)l)l & \text{if } i \text{ is odd } \& \; n_{gr} - 2 < j \leq 2n_{gr} - 4 \\ 0 & \text{if } i \text{ is odd } \& \; 2n_{gr} - 4 < j \\ 0 & \text{if } i \text{ is even } \& \; j \leq 2n_{gr} - 4 \\ H(\hat{Y}_{\lceil \frac{i}{2} \rceil} - (j-2n_{gr}+3)l)l & \text{if } i \text{ is even } \& \; 2n_{gr} - 4 < j \leq 3n_{gr} - 6 \\ H(\hat{X}_{\lceil \frac{i}{2} \rceil} - (j-3n_{gr}+5)l)l & \text{if } i \text{ is even } \& \; 3n_{gr} - 6 < j \end{cases}, \tag{2.35}$$

where the hat on top of X and Y is used to distinguish the reference location vector components of the grid nodes from those of the DNS mesh. Furthermore, $H(\bullet)$ denotes the Heaviside step function (with $H(0) = 0$) and l denotes the length of a grid element (which we consider to be the same in horizontal and vertical direction because the RVE of interest is a square).

Offline stage: Identification of the global basis functions

Similarly as for scheme I, we investigate the same two approaches to identify the global basis functions. In the first approach, we again let the global interpolation capture as much of the fluctuating displacement field as possible and assume that the local interpolation captures the remaining inaccuracies. This entails that the global basis functions are exactly the same as the ones of the conventional POD-based MOR of the previous section.

In the second approach, we again let the local interpolation capture as much of the fluctuating displacement field as possible, whereas the global interpolation is intended to compensate for the remainder of the fluctuating displacement field. To this end, similarly as for scheme I, we solve the following minimization problem for each training solution:

$$\underline{\gamma}^* = \underset{\underline{\gamma}}{\operatorname{argmin}} \quad \|\underline{\tilde{u}} - \underline{\underline{\Omega}}\,\underline{\underline{M}}\,\underline{\gamma}\|_2^2. \tag{2.36}$$

Subsequently, the global basis functions are found by storing $\underline{\tilde{u}} - \underline{\underline{\Omega}}\,\underline{\underline{M}}\,\underline{\gamma}^*$ of each training solution as a column in matrix $\underline{\underline{S}}$ from which the global basis functions are extracted using SVD.

Online stage: Computing the variables of the local/global basis functions

The linearized governing equations of the online simulations are similar to those for the first scheme. The local ($\underline{\gamma}$) and global coefficients ($\underline{\alpha}$) are computed using

Eq. (2.30) at the online stage, with a small difference in the definition of matrix $\underline{\underline{A}}$, which is now written as:

$$\underline{\underline{A}} = \left[\underline{\underline{\Phi}} \quad \underline{\underline{\Omega}}\,\underline{\underline{M}}\right]. \tag{2.37}$$

Equivalent to the scheme I, the conventional POD-based MOR of section 2.3 is recovered if the local FE discretization consists of a single quadrilateral element.

The presence of an additional local interpolation is expected to improve the online computations of conventional POD-based MOR using a less number of basis functions, consequently reducing the required number of quadrature points for the hyperreduction strategy of [35].

2.5 Results and discussion

We investigate the above discussed MOR methods for the RVE shown in Fig. 2.7 which consists of stiff elastic particles in an elastoplastic matrix and is subjected to monotonic loading. The material parameters for the matrix are set to $E = 1$, $\nu = 0.3$, $M_0 = 0.01$, $h = 0.02$ and $m = 1.05$. For the particles, the elastic material parameters are set to $E = 20$, $\nu = 0.3$, while $M_0 = \infty$ ensures that the particles behave purely elastically. Because the matrix deforms mostly plastically and plastic deformation is isochoric, and because the particles deform only minimally relative to the matrix (due to the ratio of Young's moduli), we only consider the application of isochoric macroscale deformations (i.e. $\det(\mathbf{U}^M) = 1$), governed by bounds $0.5 \leq U^M_{11} \leq 1.5$, $0.5 \leq U^M_{22} \leq 1.5$ and $-0.5 \leq U^M_{12} \leq 0.5$.

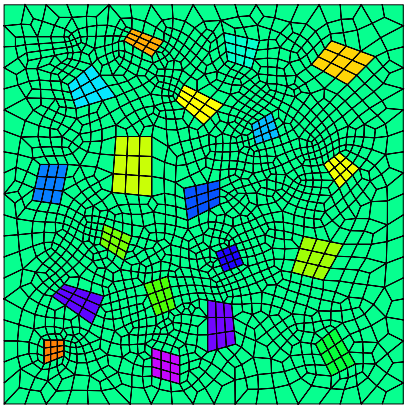


Figure 2.7: The discretized RVE with stiff, elastic particles.

Training data is generated by solving the DNS problem for 36 monotonically increasing load paths shown as red dashed curves in Fig. 2.8. The online predictions of the different MOR approaches are tested using four verification test simulations, shown as black solid lines in Fig. 2.8. All load paths are applied in 1000 load increments. The training solutions, i.e. the nodal displacement values, are extracted after the first and at every 50^{th} load increment of each training simulation.

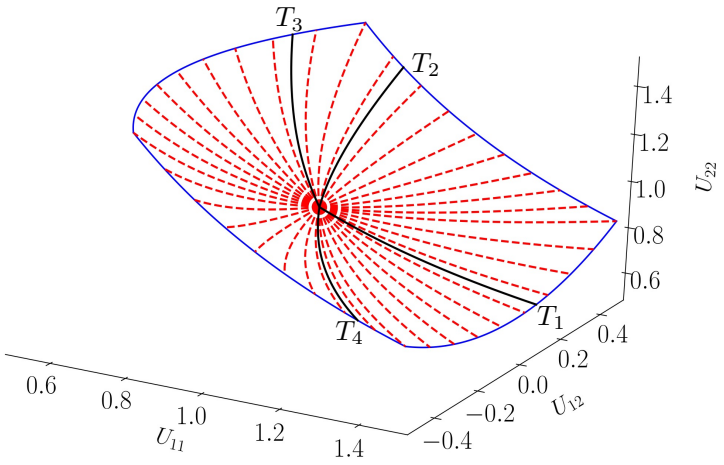


Figure 2.8: 36 training load path as red dashed lines and four verification load paths as black solid lines.

The performances of the local/global schemes are investigated for the following aspects:

1. DNS vs. Conventional POD-based ROM vs. Scheme I vs. Scheme II: which of the two new schemes performs best and how much better than the conventional ROM?

2. The orthonormality of the local interpolation functions: does the presence or absence of the orthonormality of the local interpolation functions with respect to each other, as well as with respect to the global basis functions influence the results?

3. The grid refinement: does a finer grid improve the results?

4. The identification of the global basis functions: do the schemes perform better if the same global basis functions are used as for conventional POD-based ROM, or if the local interpolations capture the majority of the fluctuating displacement field and the global basis functions only compensate for the local interpolation's deficiency?

In order to investigate the impact of the orthonormality of the local interpolation functions with respect to the global basis functions, we only consider scheme I with a 4×4 grid for verification simulation T1. The orthonormalization of the local interpolation functions with respect to each other and the global basis functions is performed using the Gram-Schmidt process. Fig. 2.9 presents the homogenized, in-plane 1$^{\text{st}}$ Piola Kirchhoff stress values predicted using the DNS, the conventional ROM and Scheme I with the same global basis as in the conventional ROM (i.e. assuming that the global basis functions capture most of the flucuation displacement field). The results with orthonormalized and non-orthonormalized local interpolation functions match each other perfectly.

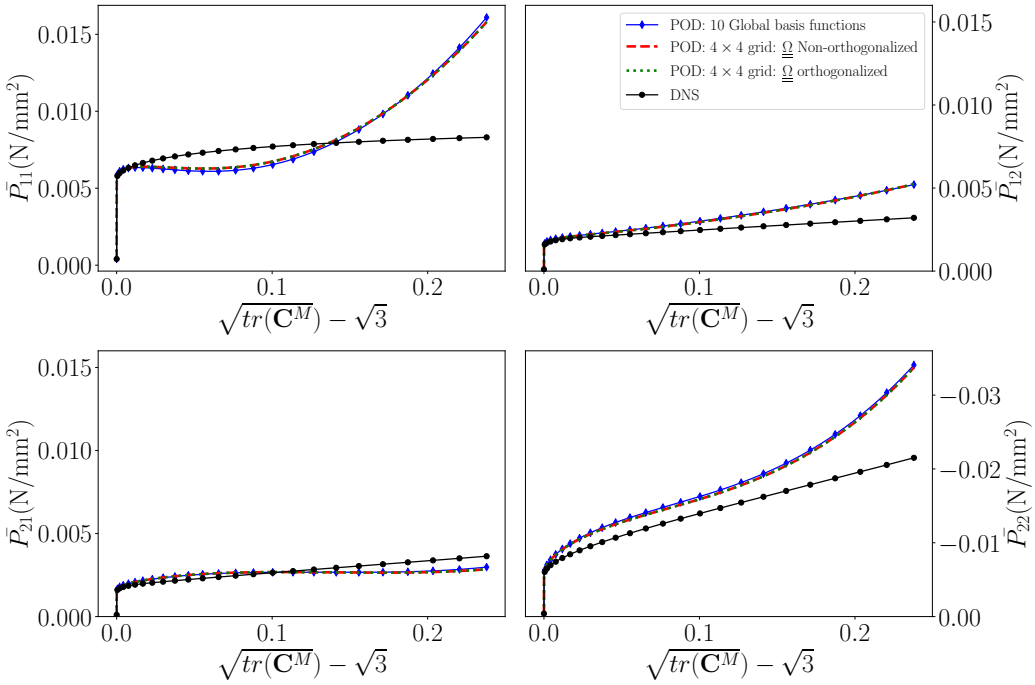


Figure 2.9: Orthonormality investigated for scheme I: using a 4×4 local grid and ten global basis functions, which are the same in the conventional MOR.

Fig. 2.10 again shows the in-plane components of the homogenized 1$^{\text{st}}$ Piola Kirchhoff stress, but with the global basis functions identified assuming that the local interpolation captures most of the fluctuating displacement field (scheme I, 4×4 grid). The orthonormalization of the local interpolation functions is again achieved using the Gram-Schmidt process, before Eq. (2.27) is employed to identify the global basis functions. Again, the results predicted with orthonormalized and non-orthonormalized local interpolation functions are exactly the same.

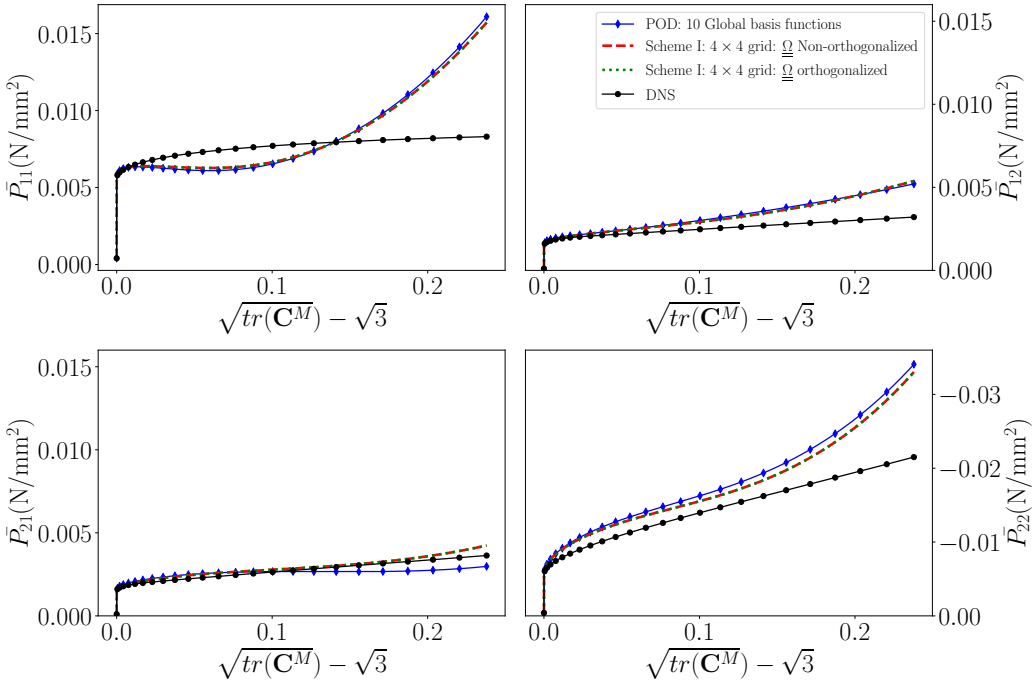


Figure 2.10: Orthonormality investigated for scheme I: using a 4×4 local grid and ten global basis functions, which are identified assuming that the local interpolation captures most of the fluctuating displacement field.

It can be inferred from Fig. 2.9 and 2.10 that orthonormalizing the local interpolation functions with respect to each other and with respect to the global basis functions has no influence on the results.

Scheme I

We now proceed to investigate the performance of scheme I on the basis of the four verification tests for different grid refinements and different numbers of global interpolation functions. We also investigate the two aforementioned approaches to identify the global basis functions: (i) Global basis functions from traditional MOR (which assumes that the local interpolation only corrects for the deficiency of the global interpolation's inaccuracy), and (ii) Global basis functions identified to compensate for the deficiency of the local interpolation in capturing the fluctuating displacement field (using Eq. (2.27)).

For each verification simulation, we measure a relative error of all in-plane components of the 1st Piola Kirchhoff stress at all quadrature points at the final increment:

$$
\begin{aligned}
e = &\left| \frac{P^{incr}_{xx,DNS} - P^{incr}_{xx,MOR}}{P^{incr}_{xx,DNS}} \right| + \\
&\left| \frac{P^{incr}_{xy,DNS} - P^{incr}_{xy,MOR}}{P^{incr}_{xy,DNS}} \right| + \\
&\left| \frac{P^{incr}_{yx,DNS} - P^{incr}_{yx,MOR}}{P^{incr}_{yx,DNS}} \right| + \\
&\left| \frac{P^{incr}_{yy,DNS} - P^{incr}_{yy,MOR}}{P^{incr}_{yy,DNS}} \right|,
\end{aligned}
\tag{2.38}
$$

where subscripts DNS and MOR distinguish the DNS results from the ROM results, respectively. The superscript $incr$ refers to the last converged increment of each simulation. Scheme I and II do not always converge until the end of the verification load paths. In those cases, the error in Eq. (2.38) is computed at the last converged load increment.

The bar charts in Fig. 2.11 summarize all relative errors of scheme I using different grid refinements and different numbers of global interpolation functions, in case the same global basis functions are used as in the conventional MOR. In other words, the global interpolation functions are calibrated such that the global interpolation captures most of the fluctuating displacement field, whereas the local interpolation is aimed to compensate for any inaccuracies. The left bar in each diagram presents the error of the conventional MOR.

It is important to realize that verification simulation 1 failed to converge for the 12×12 grid with 10 global interpolation functions, 20 global basis functions, 30 global basis functions and with 40 global basis functions after increment 495, 463, 392 and 403, respectively. Verification simulation 2 failed to converge for the 2×2 grid with 10 global interpolation functions after increment 957, for the 4×4 grid with 10 global interpolation functions after increment 951 and for the 6×6 grid with 10 global interpolation functions after increment 949. Verification simulation 4 failed to converge for the 4×4 grid and for the 8×8 grid, both with 10 global basis functions, after increment 965 and 989, respectively. Failure to converge is generally caused by the interplay of the local and global interpolation: the deformation in quadrature points can become unphysical ($\det(\mathbf{F}) < 0$). On the other hand, it is also worth to realize that the simulations include substantially large deformations (since the

bounds of the macroscale deformation are $0.5 \leq U^M_{xx} \leq 1.5$, $-0.5 \leq U^M_{xy} \leq 0.5$ and $0.5 \leq U^M_{yy} \leq 1.5$) and hence, the frameworks are tested to their limits.

The results in Fig. 2.11 show that scheme I with the same global basis functions as used in the conventional ROM indeed improves the overall accuracy. Refining the local interpolation grid generally improves the accuracy, but for larger numbers of global interpolation functions the accuracy actually reduces for the most refined grid. If the number of DoFs is of primary interest, Fig. 2.11 demonstrates that it is always better to increase the number of global basis functions, instead of using a small number of global basis functions together with scheme I (with standard global basis functions). However, if the primary interest is to limit the number of global basis functions, scheme I indeed improves the overall accuracy in most cases.

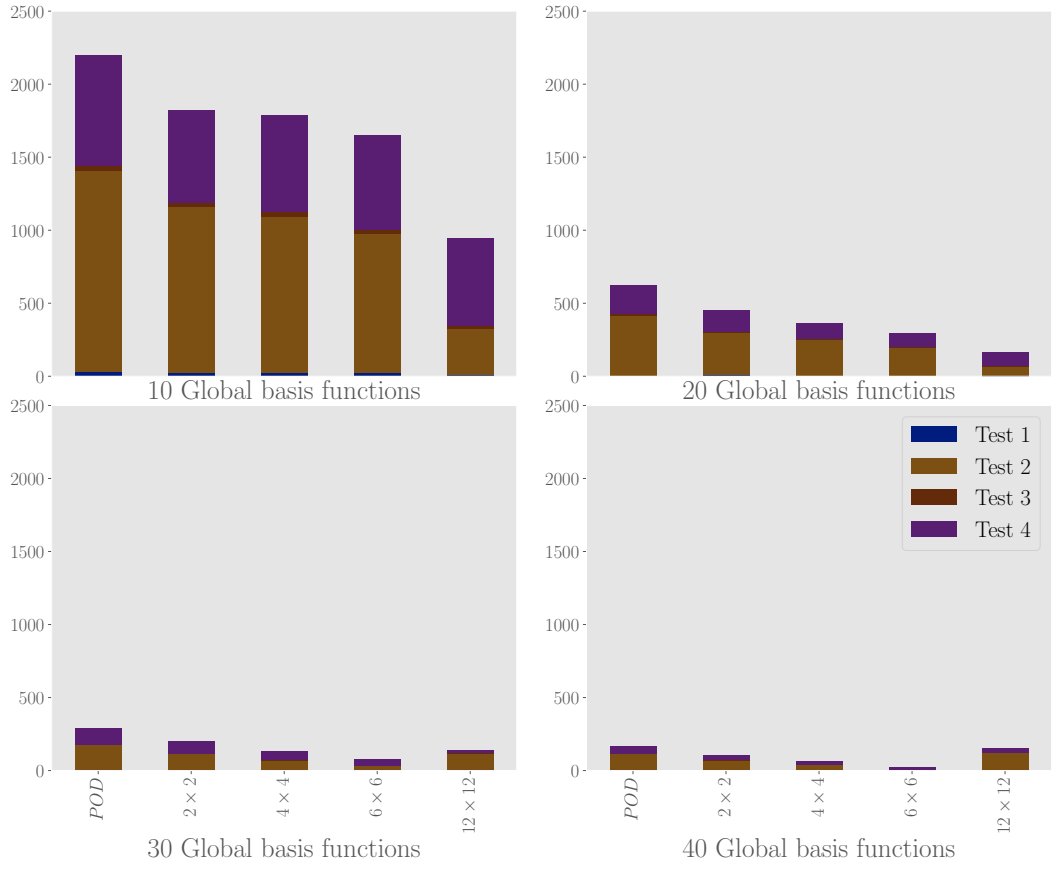


Figure 2.11: Scheme I: total error in 1st Piola Kirchhoff stress at all quadrature points using the same global basis functions as in conventional ROM.

We stick to scheme I, but change the identification of the global basis functions. Instead of the same global basis functions as in the conventional MOR, we first compute the best match of the local interpolation to capture the fluctuating displacement field (according to Eq. (2.27)) and calibrate the global basis functions such that they capture the difference between the locally interpolated fluctuation field and the actual fluctuation field.

Substantially more verification simulations converge with the new calibration of the global basis functions. Only verification simulation 1 fails for the 12×12 grid for all numbers of global basis functions tested (after approximately increment 450) and verification simulation 2 fails for the 4×4 grid and the 6×6 grid (both with 10 global basis functions) after increment 958 and 949, respectively.

The results in Fig. 2.12 generally show similar trends as the results in Fig. 2.11. We see that refinement of the grid generally improves the accuracy, but that there is again a limit to this for the largest numbers of global basis functions tested. One can also notice that the increase of the accuracy is not as consistent as when the standard global basis functions are employed. However, the best increase of the accuracy can be observed for the smallest number of global basis functions, which is the aim of this chapter. Comparing Figs. 2.11 (standard global basis functions) and 2.12 (alternative global basis functions), together with the fact that more simulations converge for the new global basis functions, the global basis functions are favored to be identified considering the treatment according to Eq. (2.27).

Scheme II

The results of scheme II with the use of conventional global basis functions are presented in Fig. 2.13. None of the simulations failed to converge, but it is clear that the accuracy is not increased at all (with a single exception). The fact that the impact of scheme II is less pronounced than that of scheme I is twofold. First, scheme II comes with substantially less DoFs than scheme I, especially for refined grids. Second, the deformation of a grid element in scheme II is governed in a substantially more nonlocal fashion than in scheme I, because each grid element shares the components of its deformation gradient tensor with the row and column in which is located.

When the new calibration of the global basis is used, the results change, as can be seen in Fig. 2.14. In case 10 global basis functions are used, the accuracy is indeed improved, albeit substantially less than for scheme I. In the case of 10 global basis

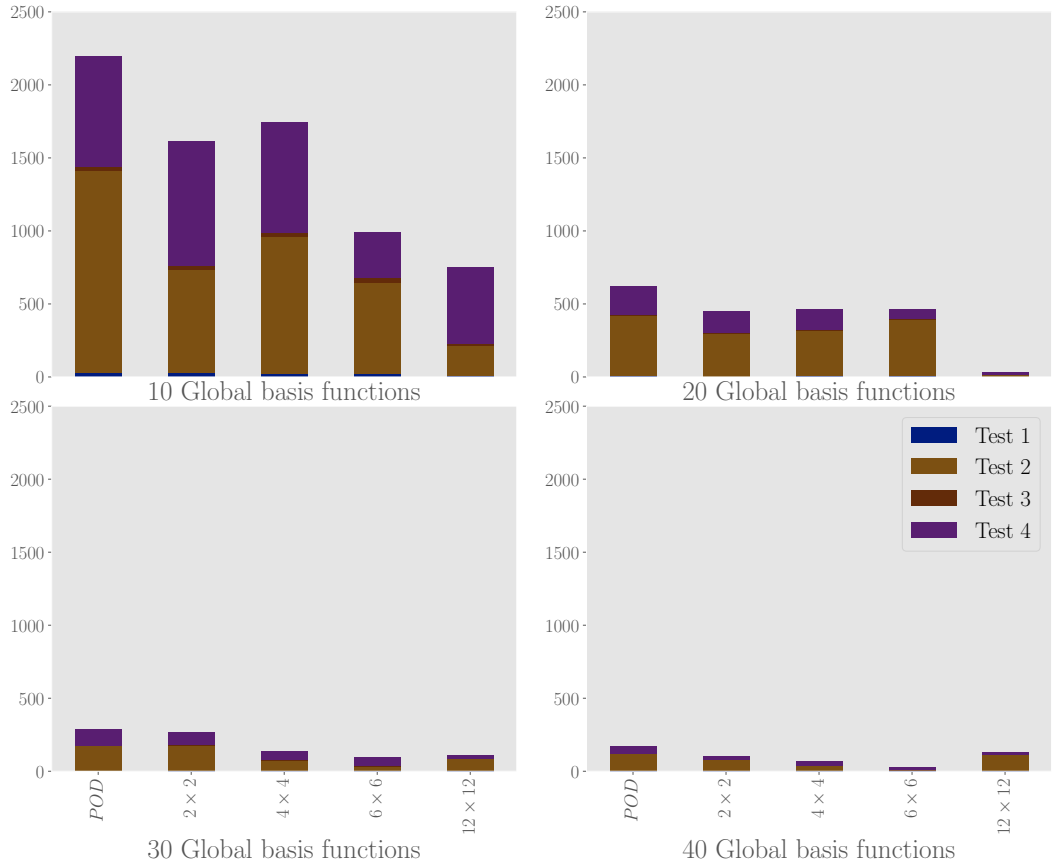


Figure 2.12: Scheme I: total error in 1st Piola Kirchhoff stress at all quadrature points using global basis functions to compensate for the deficiency of the local interpolation to describe the fluctuating displacement field.

functions on the other hand, verification test 4 fails for all grid refinements around increment 985, except for the 2×2 grid. For 30 global basis functions furthermore, the accuracy deteriorates. Comparing the results of scheme I with scheme II, it is clear that scheme I performs substantially better than scheme II.

Conforming mesh

A last illustrative result is presented for the use of the coarse discretization on the left in Fig. 2.15 as the local interpolation. This coarse mesh is different from the previously investigated grids, because its elements match the particles accurately. The coarse, conforming mesh is only investigated for scheme I, because scheme II is restricted to grids with perfectly rectangular elements.

The results obtained for the mesh with ten global basis functions are presented in Fig. 2.16 for verification simulation T2, together with those of the conventional

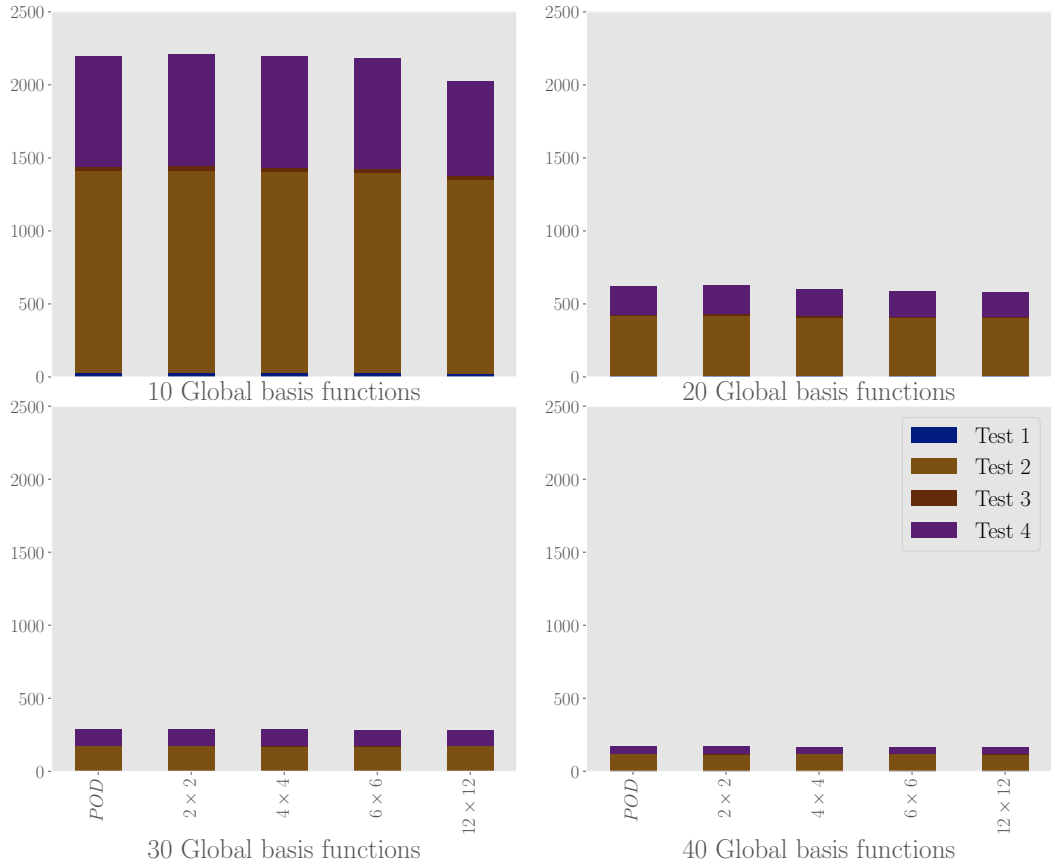


Figure 2.13: Scheme II: total error in 1st Piola Kirchhoff stress at all quadrature points using the same global basis functions as in conventional ROM.

POD-based MOR and the DNS. The results of scheme I for the 6×6 and the 12×12 grid are included in Fig. 2.16 for comparison, because the 6×6 grid comes with 48 DoFs and the 12×12 grid with 240 DoFs, whereas the mesh of Fig. 2.15 with 190 DoFs. In other words, the conforming mesh comes with less DoFs than the 12×12 grid, but with more than the 6×6 grid.

Comparing the different curves in Fig. 2.16 with the naked eye is sufficient to conclude that the conforming mesh outperforms the grids and the conventional POD-based MOR. Nevertheless, the lack of robustness is even more present for this conforming mesh. For instance, the simulations with the conforming mesh for the other three verification simulations (not shown) maximally converged until the 100th increment (out of a 1000 increments). Second, if the elements of the conforming mesh of Fig. 2.15 are split in four so that the new mesh has four times as many elements, divergence is observed for all verification simulations and at earlier increments.

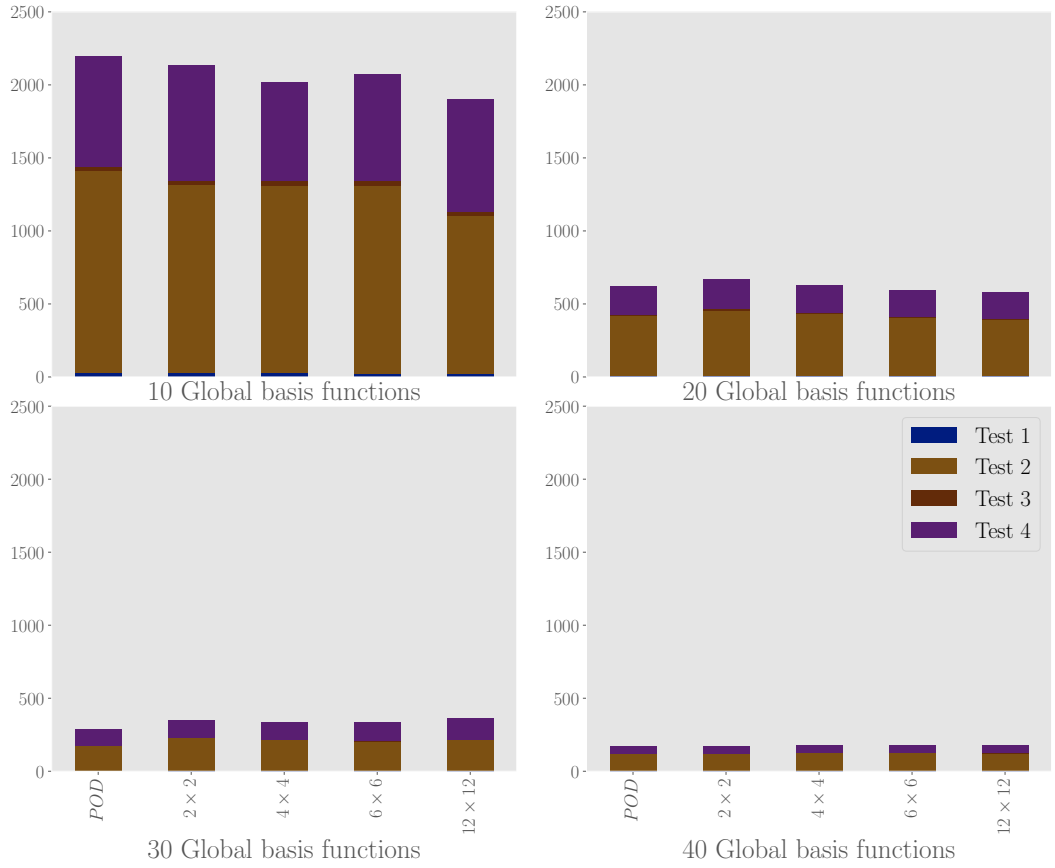


Figure 2.14: Scheme II: total error in 1st Piola Kirchhoff stress at all quadrature points using global basis functions to compensate for the deficiency of the local interpolation to describe the fluctuating displacement field.

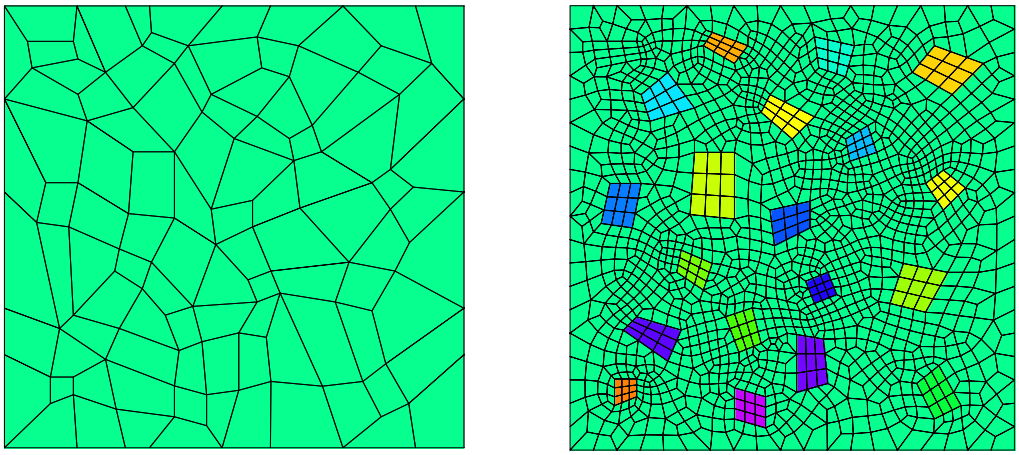


Figure 2.15: A coarse, periodic, conforming mesh on the left and the DNS' FE mesh on the right for comparison.

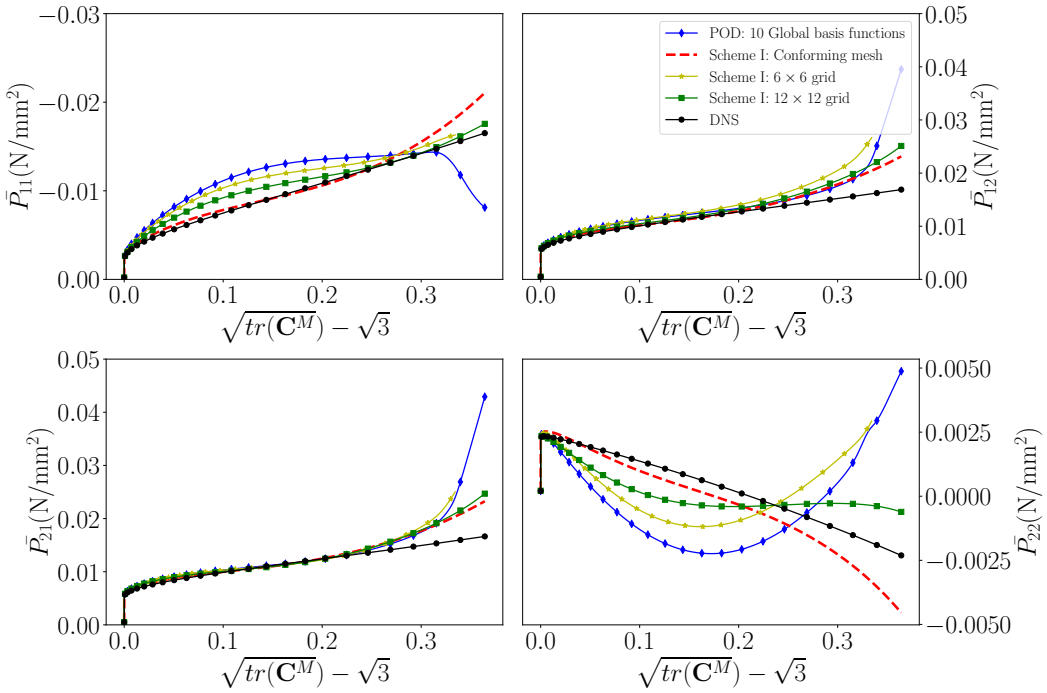


Figure 2.16: The results of scheme I for the conforming mesh of Fig. 2.15 for verification simulation T2, together with the results of the DNS, the conventional MOR and scheme I (with corrected basis functions) for the 6×6 and the 12×12 grid for ten global basis functions.

2.6 Conclusion

Projection-based model order reduction (MOR) for hyperelastoplastic simulations requires large numbers of global basis functions in order to achieve an acceptable accuracy. Large numbers of global basis functions need large numbers of quadrature points (e.g. according to the hyperreduction strategy of [35]), thereby compromizing the speed of the online simulations. Therefore, the goal of this chapter was to devise a MOR approach that reduces the number of global interpolation functions for hyperelastoplastic representative volume element simulations, so that less quadrature points are required when hyperreduction is applied.

To this purpose, the global interpolation achieved using the basis functions of conventional projection-based MOR is combined with the additional interpolation of coarse finite element discretizations. Since the additional interpolations only influence the displacement field in a part of the domain, the additional interpolation is referred to as the *local* interpolation to distinguish it from the *global* interpolation of the standard basis functions. The local interpolation functions are constructed such

that, similar to the global basis functions, they are periodic and hence, periodicity does not need to be actively enforced in the online RVE simulations.

For the two new frameworks, we have considered conventional global basis functions, which assumes that the global interpolation captures the majority of the fluctuating displacement field and that the local interpolation compensates for the global interpolation's inaccuracy. We have also formulated an alternative calibration of the global basis functions, which assumes that the local interpolation captures the majority of the fluctuating displacement field and that the global interpolation compensates for the local interpolation's inaccuracies.

The results demonstrate that the local/global MOR approaches are indeed substantially more accurate than the conventional POD-based MOR (which is not entirely surprising because they introduce more DoFs than the conventional POD-based MOR). Positive is also the fact that the increase of the accuracy of the local/global approaches is more pronounced for small numbers of global (i.e. conventional) basis functions, since the intention is to limit the number of basis functions.

The results also demonstrate that the local/global approaches clearly require more development for use in a robust manner. The main issue is that for some simulations (i.e. sets of load parameters), the new approaches perform extremely well, whereas for other simulations the new approaches diverge before the simulation is finished. This even occurred for substantially coarse grids, where each element interpolated more than enough FE nodes of the underlying, original DNS discretization. On top of that, the lack of robustness was even more pronounced for a mesh that conformed the underlying topology than for perfectly structured meshes that did not follow the underlying topology.

One potential avenue for further improvement may involve the identification of the global basis functions. In this chapter, two approaches of identification were investigated: (1) the conventional one that assumes that the global interpolation will interpolate the majority of the displacement field (and the local interpolation accounts for the deficiencies of the global interpolation), and (2) one that aims to let the local interpolation describe the majority of the displacement field (and the global interpolation accounts for the deficiencies of the local interpolation). Instead of these two extremes of the spectrum, one may formulate types of identification more in the center of the spectrum. How to formulate such an approach, and whether or not such an approach should correct for the amount of plastic deformation occurring in the training solutions, seems not trivial at this moment.

Another avenue for improvement may be found in staggered solution schemes. In the present chapter, a monolithic approach was applied (see e.g. [63]), since we simultaneously solve for the coefficients of the basis functions and for the DoFs of the local interpolation. Staggered approaches, in which one solves for one set of variables in one iteration and for the other set of variables in the next iteration, are completely accepted in fluid-structure simulations (see e.g. [18]) and have shown to provide stability to phase-field damage simulations [74]. They may therefore also help the local/global schemes of the current chapter.

Another approach that may improve the robustness of the presented schemes is increasing the hardening modulus in the stiffness matrix (h in Eq. (2.2)). Such an 'engineering trick' may be considered somewhat ad-hoc, because there is no theory to choose the most appropriate value for this increase. On the other hand, for perfect plasticity in infinitesimal strain settings, this modification is completely accepted to achieve convergence.

Chapter 3

NEURAL-NETWORK ACCELERATION OF PROJECTION-BASED MODEL-ORDER-REDUCTION FOR FINITE PLASTICITY

ABSTRACT

Compared to conventional projection-based model-order-reduction, its neural network acceleration has the advantage that the online simulations are equation-free, meaning that no system of equations needs to be solved iteratively. Consequently, no stiffness matrix needs to be constructed and the stress update needs to be computed only once per increment. In this chapter, a recurrent neural network is developed to accelerate a projection-based model-order-reduction of the elastoplastic mechanical behaviour of an RVE. In contrast to a neural network that merely emulates the relation between the macroscopic deformation (path) and the macroscopic stress, the neural network acceleration of projection-based model-order-reduction preserves all microstructural information, at the price of computing this information once per increment.

3.1 Introduction

Relatively recently, artificial neural networks (ANNs) have been investigated to emulate the relation between the macroscale deformation (path) and the macroscale stress in nested multiscale approaches [28, 72, 77, 25, 33]. Although such ANN-emulations are rapid, all microstructural information is in principle lost (some microstructural information could be included in the ANN emulator [82]). In order to preserve all microstructural information, ANNs can be combined with projection-based model-order-reduction (MOR) [45, 75].

Projection-based MOR is an *a posteriori* method; it utilizes the solutions of training simulations as global basis functions to interpolate kinematic variables of the original direct numerical simulation. It uses either a representative set of orthonormalized training solutions directly as global basis (i.e. the method of 'reduced basis' [59, 60]), or applies singular value decomposition to the training solutions, and uses the basis vectors associated with the largest singular values as global basis functions (i.e. the method of 'Proper Orthogonal Decomposition' - POD [42, 43, 52, 11]).

The global basis in projection-based MOR interpolates the kinematic variables to reduce the number of degrees of freedom in the online simulations. In this chapter, ANNs are used to emulate the values of these remaining degrees of freedom: the coefficients of the global basis functions. This eliminates the need to construct stiffness matrices, since the iterative process to solve for the basis coefficients is avoided. The only issue that remains to be computed once per increment is the stress update in each quadrature point (i.e. the plastic variables in the case of elastoplasticity).

The aim of this chapter is to formulate an ANN-accelerated POD-based MOR for finite plasticity under cyclic and random loading, applied to a representative volume element (RVE). The use of projection-based MOR for systems governed by elastoplasticity requires a large number of global basis functions to achieve an acceptable accuracy (we use 100 basis functions). As ANNs avoid the computation of the basis coefficients, many basis functions can be used and hence, ANN-accelerated projection-based MOR may be considered particularly useful in the context of elastoplastic finite element computations.

Since elastoplasticity includes both reversible and irreversible physics, a suitable ANN must be able to account for the deformation path. Since [54, 32, 83, 28] have shown that the hidden variables in recurrent neural networks (RNNs) are able to account for this (in the context of conventional finite element simulations to compute

inelastic responses), these types of ANNs are also used in the current chapter.

The remainder of this chapter is as follows: in the next section, the architecture and working of ANNs is discussed, then a brief descriptions of feed forward neural networks and recurrent neural networks are provided. Subsequently, the discussed RNN is tested to predict the coefficients of basis functions for an elaso-plastic finite element model. The results are discussed in section 3.3, where the predictions of the RNN-accelerated POD-based MOR are compared with those of the direct numerical simulation (DNS) and the conventional POD-based MOR. A short conclusion is presented in section 3.4.

The DNS are not discussed in this chapter, because they are the same as those of the previous chapter. Similarly, a discussion on conventional POD-based MOR is lacking (to promote conciseness), because it follows the same recipe as discussed in section 2.2.

3.2 ANN-acceleration

In this section, we discuss the RNN that rapidly emulates the basis coefficients in $\underline{\alpha}$ for each increment (the input is $\mathbf{U}^M$). This circumvents the iterative process (Eq. (2.20)) necessary for conventional POD-based MOR.

Introduction to neural networks

A neural network is a combination of numerous neurons. Each neuron receives several input values (O_1^{j-1} to O_k^{j-1} in Fig. 3.1), and outputs a single value (O_n^j) as a function of weighing the input values (w_i), adding a bias to it (b), and inserting the result in an activation function (f) (Appendix A discusses commonly used activation functions in more detail). A collection of neurons are grouped together to form a layer.

The best known ANNs are feed forward deep neural networks (see Fig. 3.2). In deep neural networks, several layers of neurons are placed one after another (see Fig. 3.2). The neurons in subsequent layers are connected with each other [37]. Feed forward networks have a unique relationship between input and output data, and cannot handle sequential information (i.e. a sequence of $\mathbf{U}^M$), as required for path-dependent models such as elastoplasticity [65]. In other words, the output predicted by a feed forward network for a given input does not depend on previous input provided to the network.

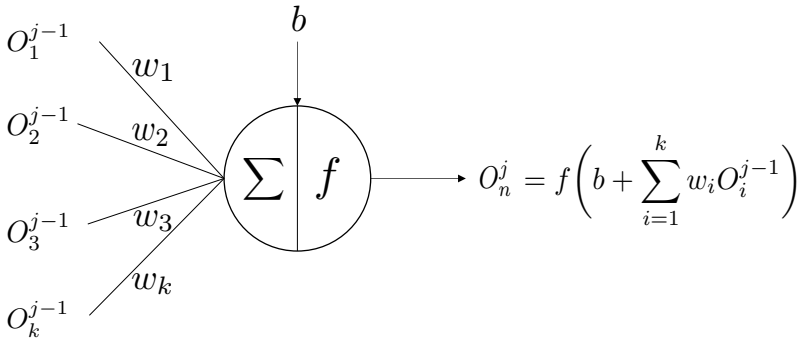


Figure 3.1: A single artificial neuron at layer j. The outputs of previous layer O^{j-1} are the inputs of current layer j.

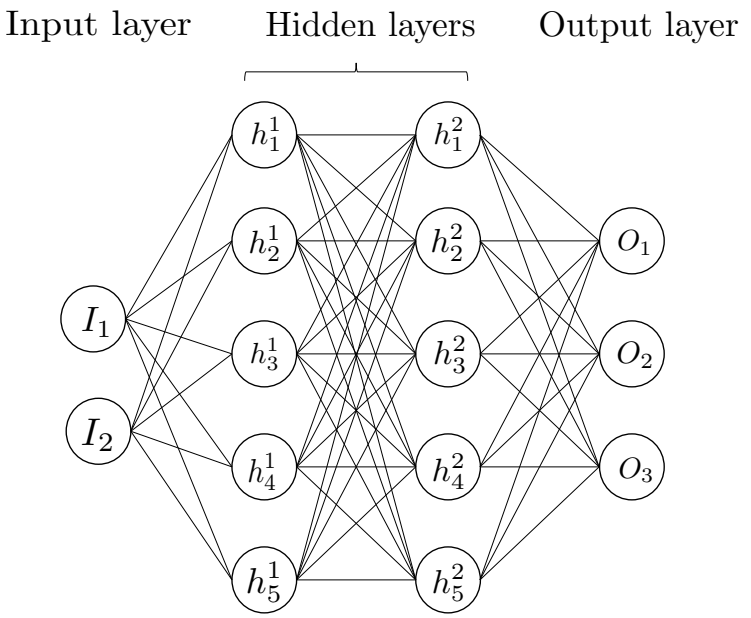


Figure 3.2: A feed forward neural network with three layers: Two hidden layers with five neurons each (h_i^j), two neurons for the input layer and three for the output layer.

On the other hand, recurrent neural networks (RNNs) have the intrinsic feature to handle sequential data. RNNs (Fig. 3.3) employ hidden state variables ('H') as memory elements to store and pass on information from the past to the upcoming prediction. Therefore, the predictions of an RNN at time step 't' are based on the input at time step ('t') and the values of the hidden variables ('H_{t-1}') at the beginning of that time step ('t'). Hidden variables 'H' can be considered as the RNN's way to quantify the past, in analogy to history variables $\underline{z}$ in the DNS and

the conventional POD-based MOR.

Traditional RNNs suffer from the problem of vanishing gradients while handling long sequential data using standard gradient based optimizers in the learning stage [44]. This is due to the fact that the parameters at the beginning of the sequence depend on the gradient of the parameters present later in the sequence. In this process, the derivatives, which take small values, are multiplied several times, resulting in significantly smaller values, explaining the term vanishing gradients. To overcome this problem, a gated RNN, employing Long Short Term Memory (LSTM) [39] or Gated Recurrent Unit (GRU) [16] is generally used.

In this chapter we use an RNN with a Gated Recurrent Unit (GRU). The GRU enables control over the flow of information through its 'hidden state' variables '$\underline{H}$'. It uses an update gate and a reset gate to determine the amount of information to be passed on and to be retained by the hidden variable. The gated structure of a GRU also controls the flow of gradients during learning, such that the parameter update value does not vanish. The reader is referred to appendix B for a detailed description about the working of a LSTM and a GRU unit.

The RNN architecture used in this chapter is shown in Fig. 3.3. The feed forward neural networks (FFN_I and FFN_O) at the input and at the output of the GRU increase the flexibility in the RNN design [83]. The hidden variables 'H' of the GRU unit are initially set to -1, as recommended in [83].

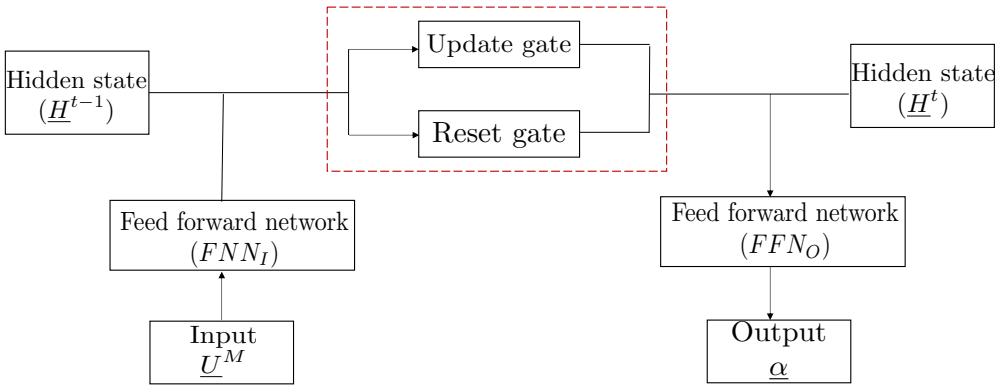


Figure 3.3: Neural network architecure used in this chapter. The red dashed box indicates the GRU.

Learning phase of the recurrent neural network

The learning phase is the term used for the phase in which the parameters of the neural network are identified. This is effectively a least squares minimization problem for which many input-output relations must be available (which are obtained by running the conventional POD-based MOR for many input variations). The minimization is performed using a stochastic steepest descent algorithm [64, 78], that requires the gradient to be constructed for each iteration.

We perform a supervised learning strategy in which the data is passed through the RNN numerous times, where every time the entire data is passed through is referred to as an epoch. The learning stage, i.e. the identification of the weights and biases, is an iterative process in which the following loss function (3.1) is minimized to increase the network's accuracy:

$$L_{\mathrm{MSE}} = \frac{1}{n_t} \sum_{i=1}^{n_t} \|\underline{\tilde{\alpha}}^i - \underline{\alpha}^i\|_2^2, \tag{3.1}$$

where the tilde is used to denote the reference coefficients (i.e. those predicted by the conventional POD-based MOR, which the RNN must replicate).

An epoch has a forward propagation stage in which the information is passed from the input layer to the output layer. At the beginning of learning, the parameters such as weights and biases are initialized randomly.

In the backward propagation stage, the loss function's gradients with respect to the RNN's parameters are calculated starting at the output layer and ending at the input layer. A gradient descent algorithm is used to update the RNN's parameters. The procedure is repeated for thousands of epochs.

The data is fed to the network in multiple mini-batches in order to speed up the training process. Since we have a large data set, updating the gradients for the whole data at once as in the method of batch gradient descent is not computationally efficient. Also, updating the gradients for one data at a time as in the method of stochastic gradient descent will slow down the learning time of the network. Therefore, mini-batch gradient descent is used to update the parameters of the RNN. Each batch consists of a sequence of input-output pairs that are extracted as training solutions at every load increment. The length of sequences is equal for all the batches. In the current chapter, data in each batch corresponds to the solutions of a single training simulation. There are 1000 load increments per simulation, therefore each batch is of length 1000.

3.3 Results

Model setup and data collection

The discretized RVE is portrayed in Fig. 3.4 and is subjected to cyclic and random loading. The material parameters for the elasto-plastic matrix are set to $E = 1$, $\nu = 0.3$, $M_0 = 0.01$, $h = 0.02$ and $m = 1.05$ (see Eq. (2.2)). For the particles, the elastic material parameters are set to $E = 20$, $\nu = 0.3$, while $M_0 = \infty$ ensures that the particles behave purely elastically. Because the matrix deforms mostly plastically and plastic deformation is isochoric, and because the elastic particles deform only minimally relative to the matrix (due to the ratio of Young's moduli), we only consider the application of isochoric macroscale deformations (i.e. $\det(\mathbf{U}^M) = 1$), governed by bounds $0.75 < U^M_{11} < 1.25$, $0.75 < U^M_{22} < 1.25$ and $-0.25 < U^M_{12} < 0.25$. This means that it is sufficient to only consider components U^M_{11} and U^M_{12} as input variables of the RNN (as $\det(\mathbf{U}^M) = 1$, U^M_{11} and U^M_{12} dictate the value of U^M_{22}). The work flow of the ANN-accelerated MOR is depicted in Fig. 3.5.

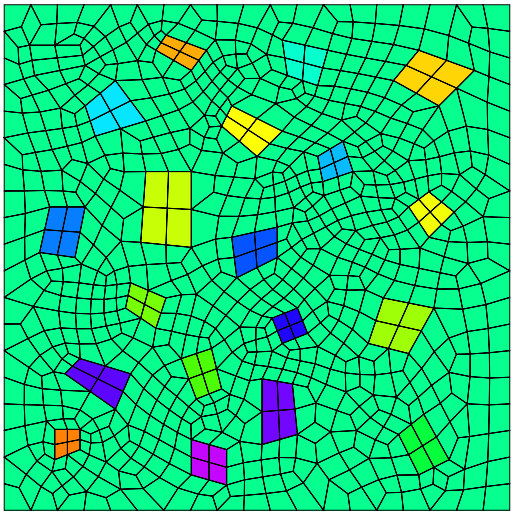


Figure 3.4: The discretized RVE with particles.

A single RNN is simultaneously trained to emulate the basis coefficients for both cyclic and random loading. Random loading is not per se simulated because it is expected in nested multiscale simulations. Instead, it is considered to enhance the training because in true multiscale simulations with cyclic loading, the cyclic loading path of each RVE will slightly differ for each cycle [83]. The loading paths of the cyclic training simulations are presented in the left diagram of Fig. 3.6. Each involves a loading stage and an unloading stage, each stage consisting of 500 increments. In the random loading simulations (one loading path is shown on the

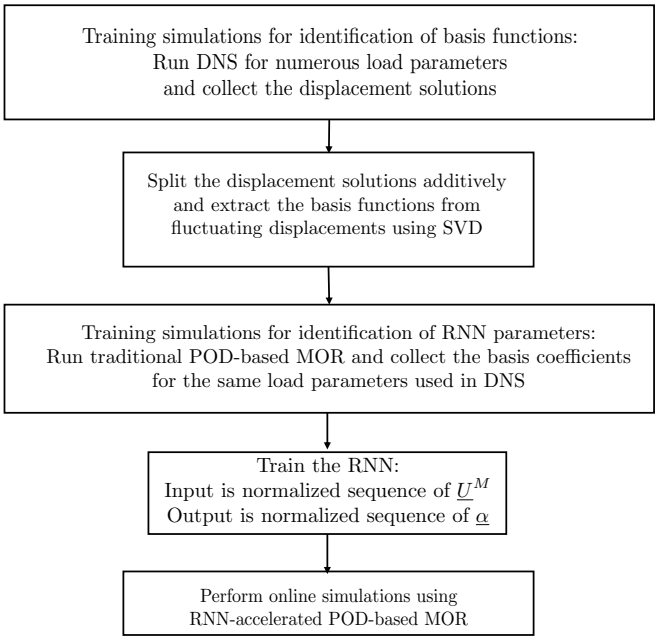


Figure 3.5: A flow chart of the main steps necessary to obtain the RNN-accelerated POD-based MOR.

right in Fig. 3.6), the loading direction is randomly selected for each of the 1000 increments and the loading step is fixed.

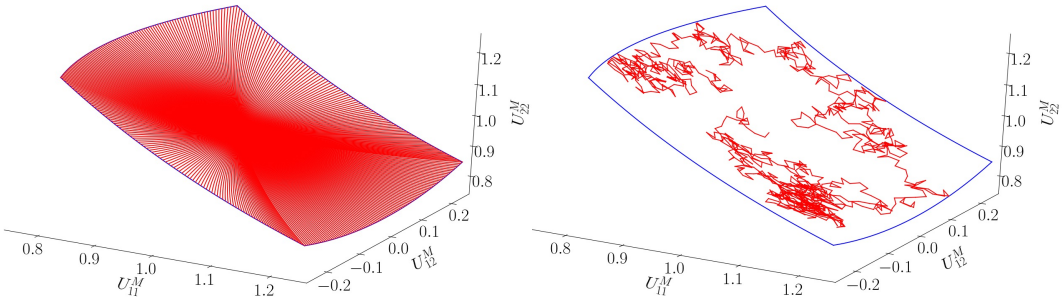


Figure 3.6: Left: Each red line presents the load path of a cyclic training simulation. Right: Load path of a single training simulation for random loading. Bounds $0.75 < U_{11}^M < 1.25$, $0.75 < U_{22}^M < 1.25$ and $0.75 < U_{12}^M < 1.25$ of surface $\det(\mathbf{U}^M) = 1$ are presented by blue lines.

In order to obtain the data for training the neural network, the DNS described in section 2.2 is solved for 350 cyclic loading training simulations (+10 verification simulations) and 10,000 random loading simulations (+100 verification simulations). The displacement values are extracted at every load increment of the training simulations. The extracted displacement solutions are additively decomposed into

a homogeneous and a fluctuating deformations following eq. (2.14) as depicted in fig. 2.2. The basis functions representing the training solutions are obtained by performing SVD to the fluctuating deformations. The the traditional POD-based MOR described in section 2.3 is solved for the same set of load parameters. During the online stage of POD, the coefficients $\underline{\alpha}$ of the basis functions are extracted at every load increments of all the POD simulations. Therefore, the data set for training the RNN has an input of all the macroscopic deformation sequences prescribed as load parameters of the RVE. The output of the data set is the coefficient sequences of basis functions $\underline{\alpha}$, which are expected to be predicted by the RNN later during online simulation.

In this chapter 100 POD basis functions are used, whose coefficients are extracted at every load increment of each training simulation. We use a large number of basis functions, because elastoplasticity yields non-ellipticity, entailing that a substantial number of basis functions are required to obtain an acceptable accuracy.

RNN predictions

The learning stage of a neural network (i.e. the algorithm to minimise the loss function in order to identify the network's weights and biases) requires the selection of several hyperparameters. Different combinations of the numbers of layers, neurons and hidden variables are investigated with respect to the convergence of the loss function. Fig. 3.8 shows that the number of hidden variables and the number of hidden layers of FFN_O influence the RNN's accuracy the most.

It can also be inferred from Fig. 3.8 that increasing the number of layers in FFN_O, number of hidden variables '$\underline{h}$' and neurons 'N' in FFN_I increases the loss function. Therefore, in this chapter the number of hidden layers in FFN_I, FFN_O and the number of GRU layers is set to 1. The number of neurons in the hidden layer of the FFN_I is set to 100. Both FFNs use the 'Leaky ReLu' activation function. The number of hidden state variables in the GRU and the number of neurons in the hidden layer of FFN_O are set to 1600.

In this chapter, the ADAM optimizer is used to perform the stochastic gradient descent. The initial learning rate is set to 0.001 (i.e. symbol 'γ' in ADAM terminology) and its exponential decay values are given by 0.9 (for the first moment estimates) and 0.999 (for the second moment estimates) (symbols 'β^1' and 'β^2' in ADAM terminology, respectively), whilst a cut-off values of 10^{-8} is used to prevent dividing by zero (symbol 'ϵ' in ADAM terminology).

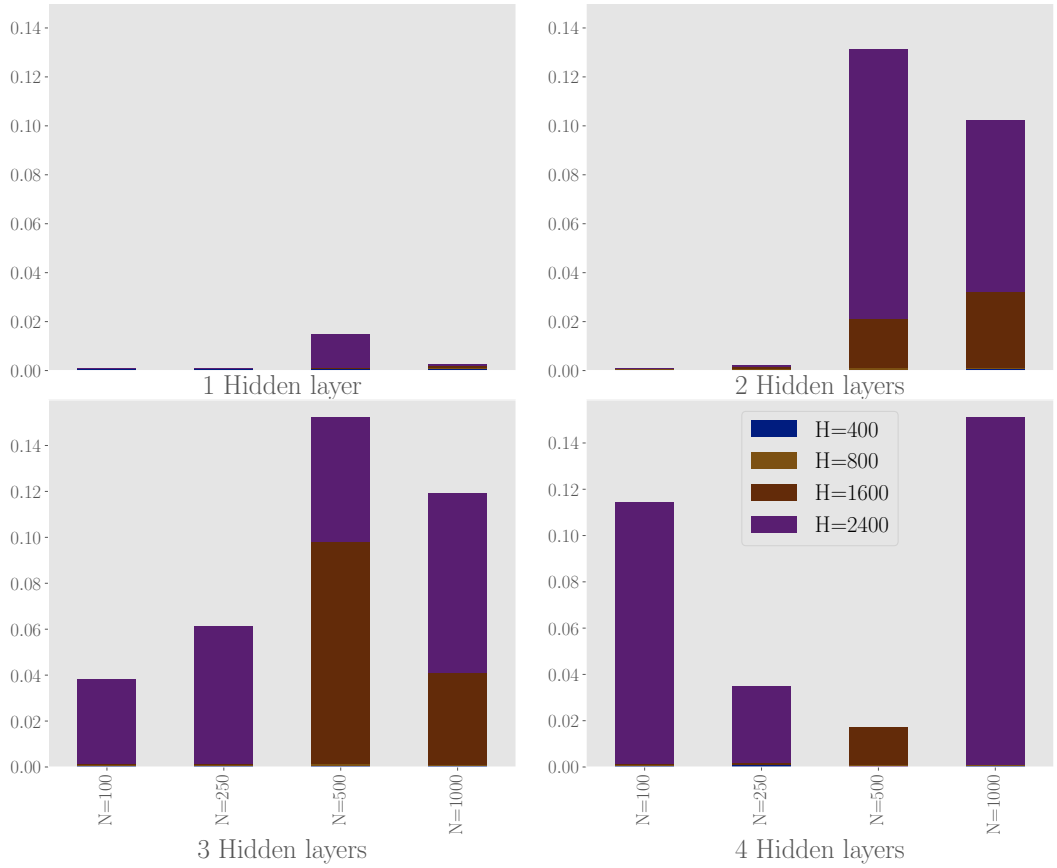


Figure 3.7: The loss function value of training data after training for 60,000 epochs for three neural network parameters: (i) Number of hidden layers in the FFN_O, (ii) Number of neurons 'N' in the hidden layers of FFN_I and (iii) Number of hidden variables 'H' of the GRU. The size of the bar corresponds to the value of the loss function (i.e. a large bar corresponds to a large value of the loss function).

The mini-batch size equals the total number of load increments in the traditional POD problem, which is set to 1000. The RNN is trained for a total of 450,000 epochs, after which the loss function did not decrease further for both training and verification data, as shown in Fig. 3.9 (a) and Fig. 3.9 (b). Each epoch consists of approximately 1% of the whole training data in a mini-batch. Therefore, mini-batch is switched every 50 epochs to include all the training simulations. After training, the loss function (Eq. (3.1)) is reduced to $4.7 \cdot 10^{-5}$.

Coefficients for some POD basis functions predicted by the RNN are presented in Figs. 3.10 (cyclic loading) and 3.11 (random loading), together with the exact coefficients. It is clearly visible that the RNN's accuracy for cyclic loading is higher than for random loading, although the accuracy for random loading is still acceptable

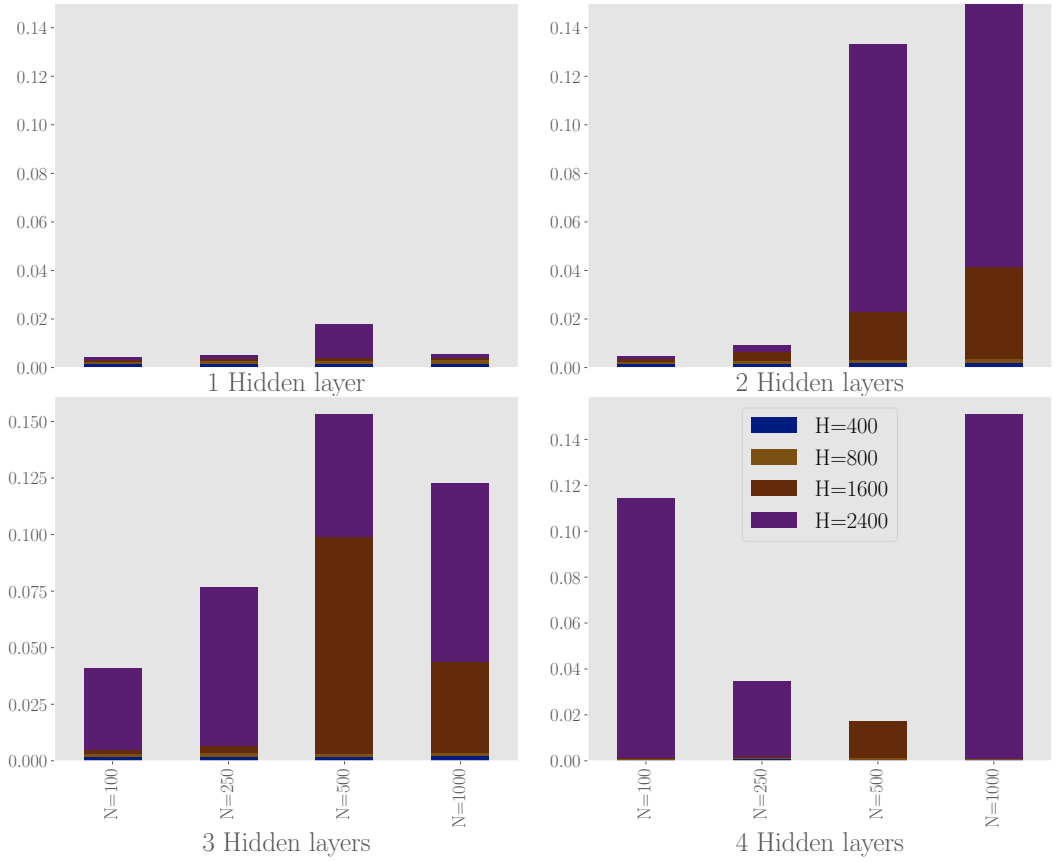


Figure 3.8: The loss function value of verification data after training for 60,000 epochs for three neural network parameters: (i) Number of hidden layers in the FFN_O, (ii) Number of neurons 'N' in the hidden layers of FFN_I and (iii) Number of hidden variables 'H' of the GRU. The size of the bar corresponds to the value of the loss function (i.e. a large bar corresponds to a large value of the loss function).

in our opinion. The average error calculated using Eq. (3.1) for the 10 cyclic loading verification simulations is around $3 \cdot 10^{-5}$, whereas the average error for the 100 random loading verification simulations is around $4 \cdot 10^{-4}$.

Mechanical predictions

In this subsection, we compare the results of RNN-accelerated MOR with those of the conventional MOR and the DNS. We start with Fig. 3.12 in which the components of the homogenized 1^{st} Piola-Kirchhoff stress are presented for one of the cyclic loading verification simulations. We can see that the POD results match those of the DNS fairly accurately (albeit not perfectly), indicating that the number of 100 basis functions is sufficiently large. The results of the RNN-accelerated MOR also match those of the DNS and those of the MOR fairly accurately. Clearly, some differences

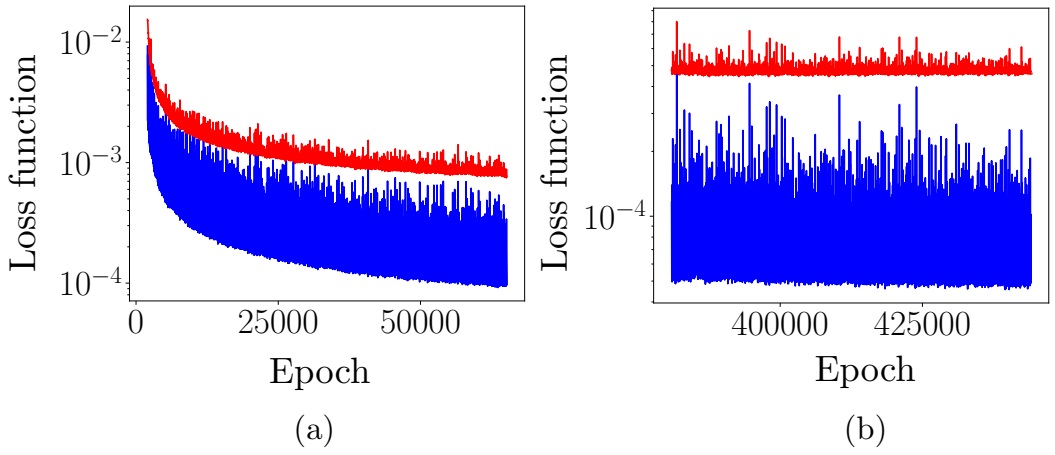


Figure 3.9: (a) Right: The evolution of loss function for both training data (blue) and verification data (red) for first 50,000 epochs. (b) Left: The loss function evolution from 350,000 to 450,000 epochs.

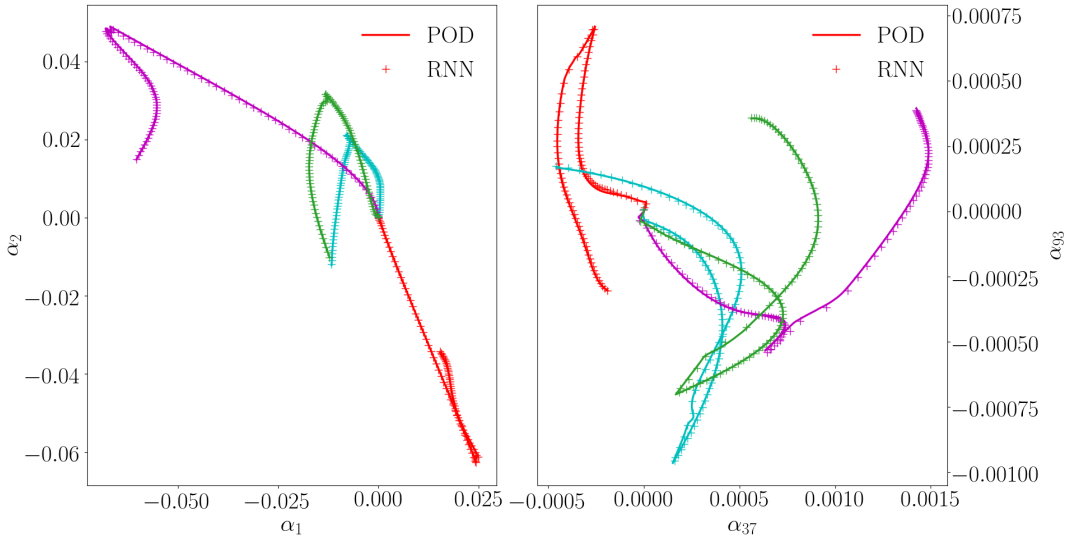


Figure 3.10: Cyclic loading verification simulations: Some RNN predictions (crosses) and the actual values (lines). The colors distinguish the four verification simulations.

are present, but the results indicate that the errors introduced by the RNN hardly influence the predicted macroscale stress.

As the appeal of RNN-accelerated MOR is the preservation of detailed information (microstructural information in case of RVEs), we also compare the plastic variables (λ in Eq. (2.3)) predicted by the different approaches. Fig. 3.13 shows that the difference in the plastic variables predicted by the DNS and predicted by the MOR

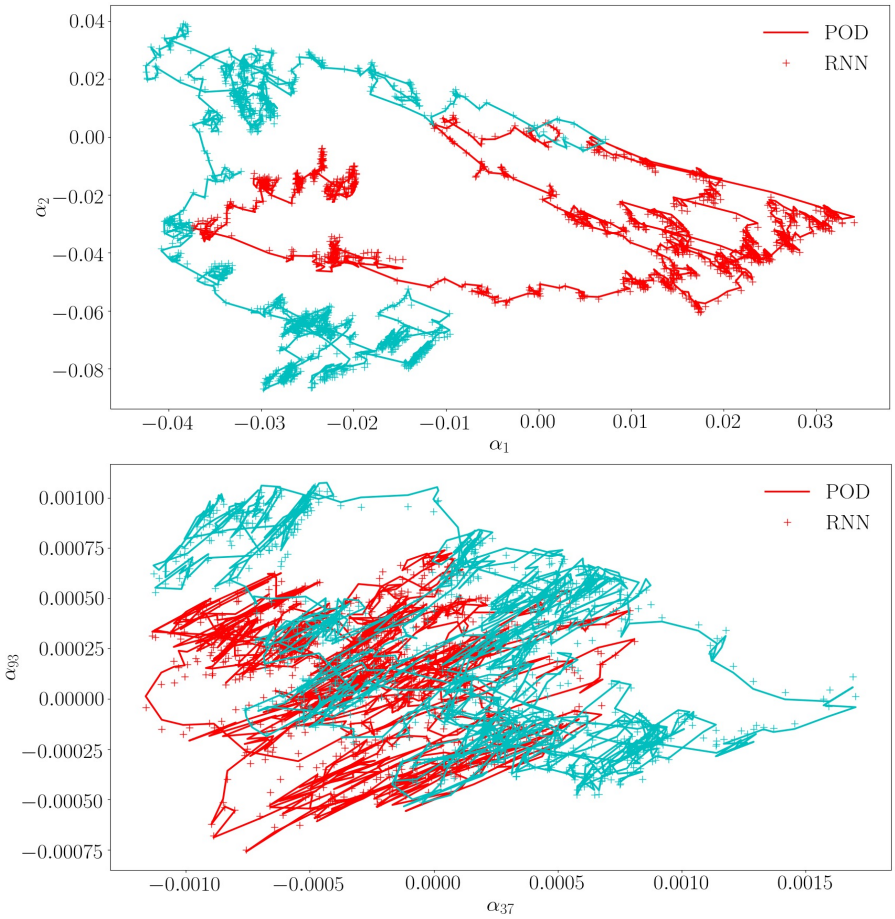


Figure 3.11: Random loading verification simulations: Some RNN predictions (crosses) and the actual values (lines). The colors distinguish two verification simulations.

is not negligible, although also not completely unacceptable. The difference can be minimized by incorporating even a large number of POD basis functions, as opposed to 100 basis functions used in the current chapter. In turn, the difference in the plastic variables predicted by the conventional MOR and predicted by the RNN-accelerated MOR is substantially smaller.

We continue with results for random loading scenarios. In Fig. 3.14, the components of the 1st Piola-Kirchhoff stress are again presented, but now for one of the random loading verification simulations and as a function of the increment number (instead of the deformation, for readability purpose). On the other hand, Fig. 3.15 shows the plastic variables predicted by the different approaches. Comparing Fig. 3.12 with Fig. 3.14 and Fig. 3.13 with Fig. 3.15, it can be concluded that the RNN-accelerated MOR is more accurate for cyclic loading than for random loading. The results

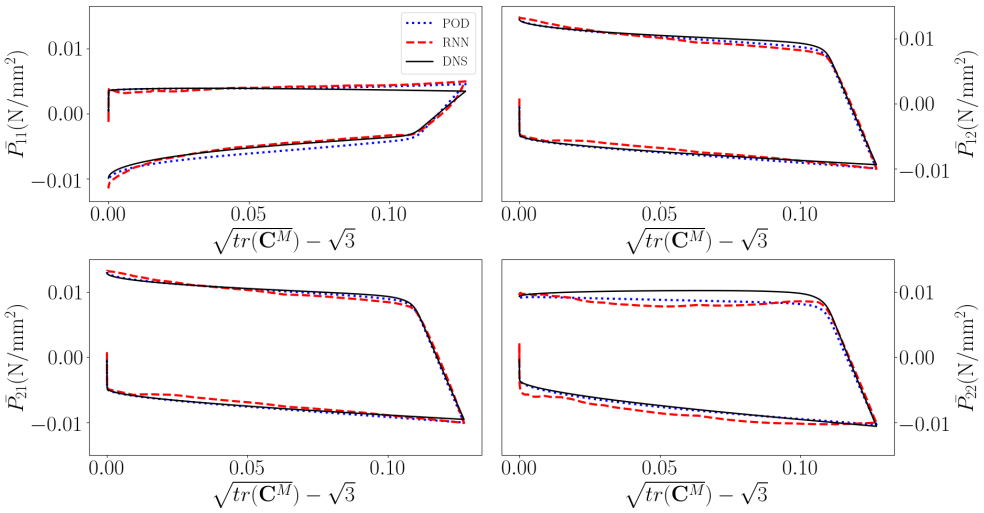


Figure 3.12: Components of the macroscale 1^{st} Piola-Kirchhoff stress as functions of the deformation for a cyclic loading verification simulation predicted by the DNS (black solid), by the conventional MOR (blue dashed), and by the RNN-accelerated MOR (red dotted).

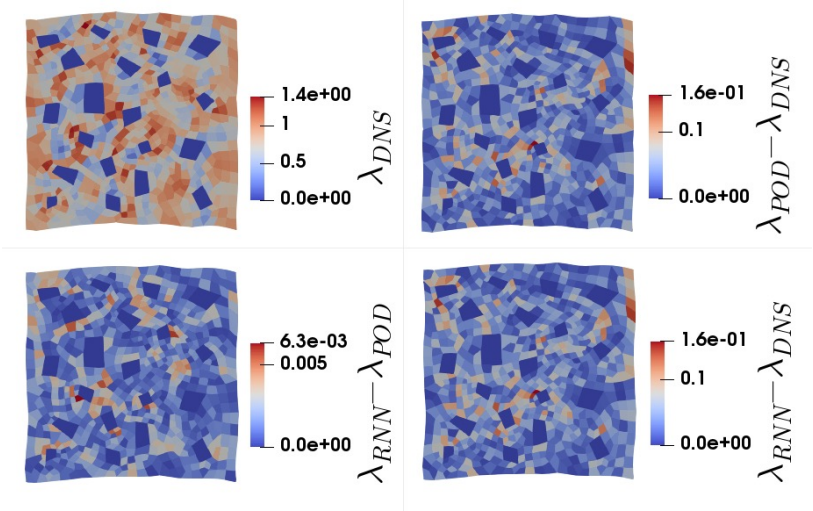


Figure 3.13: The plastic variable (λ) computed by the three methods for one of the cyclic loading verification simulations. Top-left: the DNS results, top-right: the difference between the POD results and the DNS results, bottom-left: the difference between the RNN-POD results and the POD results, bottom-right; the difference between the RNN-POD results and the DNS results.

can be argued to be sufficiently accurate, because, the random loading simulations were considered only to effectively train the RNN. But, in practice, the purpose of RNN accelerated MOR is to be utilized for loading cases arising in multi-scale simulations, which are closer to cyclic loading simulations.

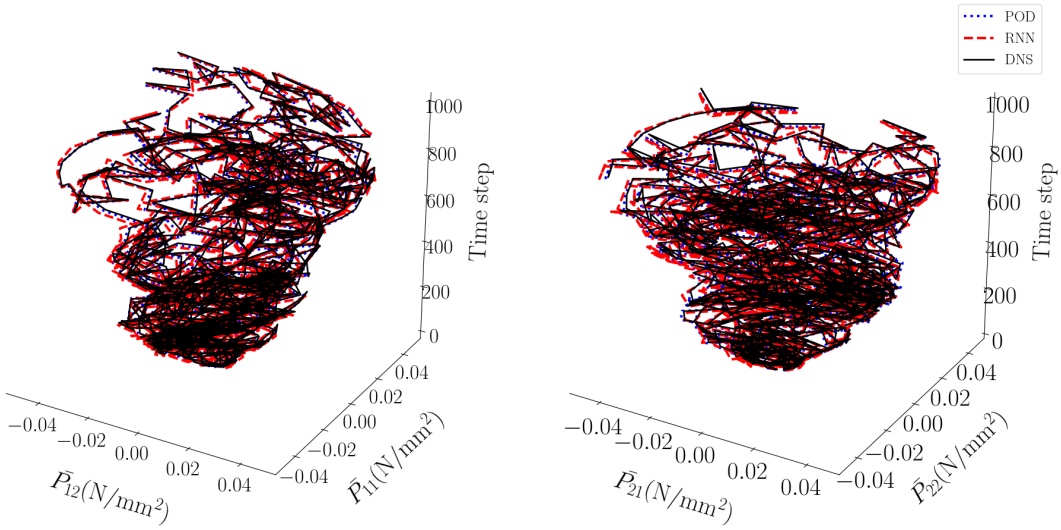


Figure 3.14: Components of the macroscale 1^{st} Piola-Kirchhoff stress values as functions of the number of increments for a random loading verification simulation predicted by the DNS (black solid), by the conventional MOR (blue dashed), and by the RNN-accelerated MOR (red dotted).

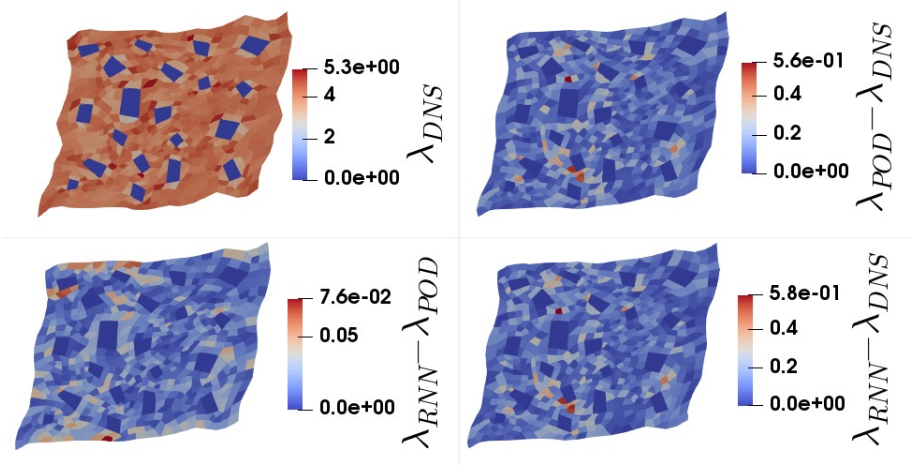


Figure 3.15: The plastic variable (λ) computed by the three methods for one of the random loading verification simulations. Top-left: the DNS results, top-right: the difference between the POD results and the DNS results, bottom-left: the difference between the RNN-POD results and the POD results, bottom-right; the difference between the RNN-POD results and the DNS results.

The computational time required to prepare the training data and to test the verifications are summarized in Table 3.1 and 3.2. The training of the RNN was performed on HPC using 32GB of GPU computational resource for 7 days. Though the data

preparation and training the RNN required a total of two weeks of computational time, the RNN-accelerated MOR is approximately 100 times as fast as the DNS and 22 times as fast as the conventional MOR in case of random loading. For cyclic loading on the other hand, the RNN-accelerated MOR is only 13 times as fast as the DNS, whilst the conventional MOR is hardly faster than the DNS.

The difference in time savings of the RNN-accelerated MOR for cyclic and random loading is because the DNS and conventional MOR require more iterations for random loading than for cyclic loading. The reasons are that (1) the load step at each increment for cyclic loading are substantially shorter than those for random loading whilst the same total number of increments is employed, and (2) the previous plastic state is known at the start of each increment of a cyclic loading simulation (i.e. DNS and conventional MOR) in order to increase the speed of the cyclic loading simulations.

Data preparation	POD	RNN-POD
Cyclic loading	350×1hr	350×1hr + 350×0.75hr
Random loading	10000×7hr	10000×7hr + 10000×1.5hr

Table 3.1: Computational time for data preparation

Online stage	DNS	POD	RNN-POD
Cyclic loading	55 min	50 min	4 min
Random loading	7 hr	1.5 hr	4 min

Table 3.2: Computational time for verification simulations

3.4 Conclusion

In this chapter, a recurrent neural network (RNN) is used to emulate the basis function coefficients of projection-based model-order-reduction (MOR) for a representative volume element described by finite plasticity, subjected to cyclic loading and random loading. The RNN is simultaneously trained for cyclic loading and random loading. We have used an RNN, because elastoplasticity is history-dependent and in analogy to the plastic variables in elastoplasticity, an RNN uses hidden variables to quantify its history.

Our results have shown that the RNN-accelerated MOR yields speed ups between factors 13 and 100 relative to the direct numerical simulations (and between factors 13 and 22 relative to conventional MOR). The accuracy is similar to conventional

MOR, which is not entirely negligible relative to the direct numerical simulations. Nevertheless, with speeds up of up to factors of 100, the RNN-acceleration of MOR seems to make MOR for finite plasticity an interesting possibility. A point to be noted is that the accuracy of RNN-accelerated MOR can be increased by employing even more number of basis functions. However, increasing the number of basis functions could reduce the RNN efficiency. Therefore, the next chapter focuses on improving the accuracy of POD-based MOR for elastoplastic solids with a reduced number of basis functions.

Chapter 4

MACHINE LEARNING FOR ADAPTIVE BASIS SELECTION IN PROJECTION-BASED MODEL-ORDER-REDUCTION FOR ELASTOPLASTICITY

ABSTRACT

Projection-based model-order-reduction is accurate and fast for simulations of (hyper)elastic solids, amongst others because only a few basis functions suffice. Model-order-reduction for simulations of (hyper)elastoplastic solids however require many more basis functions to achieve an acceptable accuracy. This compromizes the acceleration of the model-order-reduction. The aim of this chapter is to reduce the number of basis functions of elastoplastic reduced-order-models, by adaptively selecting the basis functions during the course of a simulation. To this purpose, several enhancements of a previously proposed unsupervised clustering approach are investigated, but the results lack the desired accuracy. A new approach for the adaptive selection of the basis functions is therefore formulated, which is based on a k-nearest neighbour search. The framework is applied to monotonic and cyclic loading of a representative volume element with stiff elastic particles in an elastoplastic matrix. The accuracy of the method using the k-nearest neighbour search is orders of magnitude better than those of the previous approaches, whilst only a few basis functions are used.

4.1 Introduction

Model-order-reduction (MOR) encompasses numerical methods that reduce the complexity of time-consuming computations in order to accelerate them. Whereas surrogate models such as response surfaces and Kriging replace the input-output relation of the computation of interest by a fast alternative, MOR only modifies the computation in order to accelerate it. Consequently, more results often remain available for post-processing using MOR, whilst surrogate models only provide the output for which they are trained [83].

In the field of numerical predictions of physical systems such as finite element (FE) simulations, two MOR categories may be distinguished: *a posteriori* methods and *a priori* methods. *A posteriori* MOR involves precomputing numerous responses of the physical system of interest in advance and utilizing the precomputed solutions to accelerate the subsequent simulations that remain. *A posteriori* MOR is thus only useful if the same physical system must be simulated numerous times, each time with different 'load' parameters (e.g. material parameters or boundary conditions). Consequently, *a posteriori* MOR finds its use in (probabilistic and deterministic) inverse modelling, optimization, uncertainty quantification and computational homogenization, to name a few.

On the other hand, *a priori* MOR does not require any precomputations to be performed and hence, it can be used the very first time the physical system of interest is simulated. Consequently, *a priori* MOR is more widely applicable than *a posteriori* MOR, since *a posteriori* MOR is only useful if the same type of computation must be performed numerous times. On the other hand, *a posteriori* MOR often yields larger accelerations.

One type of *a priori* MOR is proper generalized decomposition [57, 15], which enriches the approximation of the computation's solution per iteration, thereby approaching the exact result as more iterations are computed. Another type of *a priori* MOR is the quasicontinuum method, which superimposes an FE interpolation with associated quadrature points (i.e. 'summation' or 'sampling' in quasicontinuum terminology, or 'hyperreduction' in projection-based MOR terminology) over an atomistic [5], spring [6] or beam [14] lattice.

In this chapter, we focus on projection-based *a posteriori* MOR, in particular on the proper-orthogonal-decomposition (POD) method [42, 43, 52, 11] and the reduced-basis (RB) method [59, 60]. Both approaches involve an offline training stage, during which numerous computations are performed, from which solutions are harvested

that are used to construct the solutions of the future computations (i.e. in the online stage). The difference between the two methods concerns the manner in which the training solutions are treated to construct the solutions of future simulations.

In the POD method, the precomputed solutions are decomposed using singular value decomposition (SVD) in order to extract the most dominant features of the training solutions. In the method of RB on the other hand, a selection of the precomputed training solutions are directly employed (after orthonormalisation) to construct the solutions of future computations. The ensemble of selected training solutions should optimally enclose the characteristic features of the possible solutions of future simulations. This is often performed using a greedy algorithm. In both the POD and RB method, the precomputed solutions yield orthonormalized vectors that are used to construct the solutions of future computations. These vectors are referred to as basis functions or modes.

In projection-based MOR, each basis function comes with its own degree of freedom (i.e. basis coefficient), which together need to be computed in the online simulations. Consequently, it is essential for the speed of the online simulations to minimize the number of employed basis functions. In case of simulations of hyperelastic solids, only a few basis functions yield an excellent accuracy of the online simulations.

Simulations of elastoplastic solids on the other hand require a large number of basis functions to obtain an accuracy that is merely acceptable. This was for instance demonstrated in [70], in which 100 basis functions were employed for an elastoplastic model yielding an acceptable but not excellent accuracy. (The acceleration on the other hand was excellent, because of the emulation of the basis coefficients using AI-algorithms.)

The aim of this chapter is to devise a technique to reduce the number of employed basis functions in elastoplastic reduced-order-models. Because the load parameters that vary for each simulation of interest to this chapter are the load paths (which are irrelevant for the mechanical response of (hyper)elastic solids, but have a substantial effect in case of elastoplastic, i.e. history-dependent solids), the parametrization of the load paths is employed to decide whether or not to adapt the basis.

Adaptive basis selection during the course of a simulation is not new. Generally, several bases are identified during the offline stage, by clustering the training solutions according to the load parameters of (e.g. physical) relevance. In the online stage, the current load parameters are then used to classify to which group the current

configuration belongs and the basis functions of the associated group are employed. Because the load parameters may change during the course of a simulation, basis functions from different groups are typically used during the course of an online simulation.

These types of approaches were for instance proposed and/or investigated in [19, 3, 34], where the time was considered as the load parameter, based on which the training simulations were clustered into different groups. Often however, manual clustering yields rather poor results [58]. Consequently, machine learning, i.e. unsupervised learning, was proposed to obtain more suitable groups and hence, more accurate results [2, 58, 79].

To the best of the authors' knowledge, unsupervised learning was exploited in the context of projection-based MOR for dissipative solids in [13] and for viscoplasticity in [29]. [13] and [29] used clustering to improve the discrete emperical interpolation method as suggested in [58]. Similarly as in the current chapter, [13] applied clustering to the load paths. Clustering is also used for elastoplastic reduced order models which are not based on projections. In [51] for instance, the clustering of spatial domains is the essential ansatz for another type of MOR, limited to multiscale simulations.

The novelty of the current chapter starts with the application of unsupervised learning to cluster training simulations as proposed in [2, 58, 79], to elastoplastic simulations. When k-means clustering, as investigated in [2, 79], is applied to the elastoplastic simulations of the interest to the current chapter, substantial inaccuracies occur during the course of an online simulation every time the basis changes. We investigate several techniques to reduce these inaccuracies that have not been investigated yet (i.e. besides k-means clustering, we investigate DBSCAN [23], as well as approaches to smooth the transition between clusters). However, the resulting accuracies are not robustly better than that of conventional MOR based on POD.

The second novelty of this chapter is therefore to propose another technique to adaptively change the basis. Because this chapter reports that the inaccuracies for clustering appear when the (entire) basis is changed, a method that continuously changes only part of the basis is proposed. To this purpose, a k-NN search is exploited, which determines for each load increment which k training solutions are nearest to the current load path. These k training solutions are then orthonormalized with respect to each other and used as basis functions for the increment. The resulting scheme is thus a type of RB approach.

The numerical methodology is investigated for a representative volume element (RVE) with stiff elastic particles in an elastoplastic matrix that undergoes large monotonic and asymmetrical cyclic loading, in order to work towards multiscale simulations based on computational homogenization [46, 27]. The application to an RVE is however somewhat arbitrary; the methodology can just as well be applied to any other elastoplastic FE simulation.

The remainder of the chapter is organized as follows: two conventional projection-based MOR strategies are discussed in section 4.3. Clustering to enable adaptive basis selection is explained in section 4.4, including the different enhancements we have incorporated to improve the results. The k-NN search to continuously change the basis functions during the online simulations is described in section 4.5. The results of the different approaches are presented and compared to each other in section 4.6. Finally, a short conclusion is presented in section 4.7.

4.2 Direct Numerical Simulations

The plane strain simulations employ bilinear quadrilateral (four node) FEs with four Gauss quadrature points. An F-bar method is utilized to alleviate locking due to the incompressibility of the plastic deformation. In the employed F-bar method, the volume change of the deformation gradient tensor at a quadrature point is replaced with the volume change at the center of the element. The resulting deformation gradient tensor, $\bar{\mathbf{F}}$, is multiplicatively decomposed into an elastic (subscript e) and a plastic (subscript p) deformation gradient tensor: $\bar{\mathbf{F}} = \mathbf{F}_e \cdot \mathbf{F}_p$.

The following strain energy density is employed:

$$W = \frac{E(I_e - 3 - 2\ln(J_e))}{4(1+\nu)} + \frac{E\nu(\ln(J_e))^2}{2(1+\nu)(1-2\nu)}, \tag{4.1}$$

where E and ν denote Young's modulus and Poisson's ratio, respectively. Furthermore: $I_e = \mathrm{tr}(\mathbf{F}_e^T \cdot \mathbf{F}_e)$ and $J_e = \det(\mathbf{F}_e)$, where superscript T denotes the transpose. Differentiating the strain energy with respect to $\mathbf{F}_e$ gives a 1$^{\text{st}}$ Piola-Kirchhoff stress tensor, $\mathbf{P}_e$: $\mathbf{P}_e = \frac{\partial W}{\partial \mathbf{F}_e}$, which is related to the Mandel stress, $\mathbf{M}$, as $\mathbf{M} = \mathbf{F}_e^T \cdot \mathbf{P}_e$.

The employed Von Mises yield function reads:

$$y = \sqrt{\frac{3}{2}\mathbf{M}^{dev} : \mathbf{M}^{dev}} - M_0 - h\,\lambda^n, \tag{4.2}$$

where material parameters M_0, h and n denote the initial yield stress, the hardening modulus and an exponential hardening parameter, respectively. Furthermore, $\mathbf{M}^{dev}$ denotes the deviatoric Mandel stress and λ the plastic multiplier. The following associated flow rule is employed:

$$\dot{\mathbf{F}}_p = \dot{\lambda} \frac{\partial y}{\partial \mathbf{M}} \cdot \mathbf{F}_p. \tag{4.3}$$

The Karush-Kuhn-Tucker conditions close the constitutive model:

$$y \leq 0, \qquad \dot{\lambda} \geq 0, \qquad y\dot{\lambda} = 0. \tag{4.4}$$

A periodic mesh is employed in the simulations. Dirichlet boundary conditions are used for the four corner nodes, where the displacement values are dictated by the right stretch tensor of the macroscale deformation ($\mathbf{U}^M$, assuming that the RVE is used in a nested multiscale computation), given by:

$$\mathbf{u}_j - \mathbf{u}_i = \left(\mathbf{U}^M - \mathbf{I}\right) \cdot \left(\mathbf{X}_j - \mathbf{X}_i\right), \tag{4.5}$$

where $\mathbf{u}$ and $\mathbf{X}$ denote the displacement vector and reference location of a finite element node, respectively. The subscripts denote the numbers of two corner nodes. As the displacement of one of the four corner nodes is set to zero (and all reference locations are known), the displacement vectors of the other three corner nodes are completely known, since $\mathbf{U}^M$ is known for each increment.

In case of nodes on the RVE's opposing edges, the above vector equation yields two scalar constraints in a 2D setting (as is the case here). In this thesis, these constraints are enforced using Lagrange multipliers.

The incorporation of periodic boundary conditions using Lagrange multipliers results in the following system of linear equations, which must be constructed and solved for each iteration, at each increment:

$$\begin{bmatrix} \underline{\underline{K}}_{\text{int}}(\underline{u}, \underline{z}) & \left(\frac{\partial \underline{\underline{c}}}{\partial \underline{u}}\right)^T \\ \frac{\partial \underline{\underline{c}}}{\partial \underline{u}} & \underline{\underline{0}} \end{bmatrix} \begin{bmatrix} d\underline{u} \\ d\underline{g} \end{bmatrix} = \begin{bmatrix} \underline{f}_{\text{ext}} - \underline{f}_{\text{int}}(\underline{u}, \underline{z}) - \underline{g}^T \frac{\partial \underline{c}}{\partial \underline{u}} \\ \underline{c}(\underline{u}) \end{bmatrix}, \tag{4.6}$$

where column $\underline{u}$ collects the displacement components of all FE nodes at an intermediate solution, column $\underline{z}$ the plastic variables in all Gauss quadrature points at an

intermediate solution (λ and $\mathbf{F}_p$), column $\underline{g}$ the Lagrange multipliers, column $\underline{c}$ the scalar constraints due to the periodic boundary conditions of Eq. (4.5) ($\underline{c}$ is linear in $\underline{u}$), column $\underline{f}_{\text{ext}}$ the components of the reaction forces, column $\underline{f}_{\text{int}}$ the components of the internal forces ($\underline{f}_{\text{int}}$ depends non-linearly on $\underline{u}$ and $\underline{z}$) and matrix $\underline{\underline{K}}_{\text{int}}$ the derivatives of the internal forces components with respect to the displacement components ($\underline{\underline{K}}_{\text{int}}$ depends non-linearly on $\underline{u}$ and $\underline{z}$). $d\underline{u}$ and $d\underline{g}$ together denote the correction to the intermediate solution given by $\underline{u}$ and $\underline{g}$, that is to be computed each iteration. The new plastic variables are computed together with the new internal forces for each quadrature point, after new solution $\underline{u} + d\underline{u}$ is computed.

4.3 Conventional projection-based model-order-reduction

Traditionally, projection-based MOR aims to reduce the computational times of the DNS by ensuring that less governing equations need to be solved (i.e. to reduce degrees of freedom, DoFs) and, often in case of non-linear governing equations, by ensuring that the governing equations require less time to be constructed for each iteration. The latter is typically achieved using hyperreduction (or by ensuring that the governing equations only need to be constructed once per increment using an artificial neural network, as demonstrated in the previous chapter). Because this chapter solely focuses on the adaptive basis selection, the conventional projection-based MOR of the current section are only discussed with respect to the reduction of the DoFs with the help of basis functions.

The reduction of the DoFs in projection-based MOR is achieved by expressing all n_u kinematic variables of the DNS, $\underline{u}$, in terms of only a limited number of n_b kinematic variables, $\underline{\alpha}$, according to:

$$\underline{u} \approx \sum_{i=1}^{n_b} \underline{\phi}_i \alpha_i = \underline{\underline{\Phi}}\, \underline{\alpha}, \tag{4.7}$$

where $\underline{\phi}_i$ of length n_u denotes the i^{th} basis function and $\underline{\underline{\Phi}}$ denotes the $n_u \times n_b$ matrix that stores all basis functions.

For RVEs however, which are the application of this thesis, the kinematic variables (i.e. the displacement components of the FE nodes) are first additively decomposed in a homogeneous, $\bar{\underline{u}}$, and a microstructurally fluctuating, $\tilde{\underline{u}}$, contribution (see Fig. 4.1).

$$\underline{u} = \bar{\underline{u}} + \tilde{\underline{u}}. \tag{4.8}$$

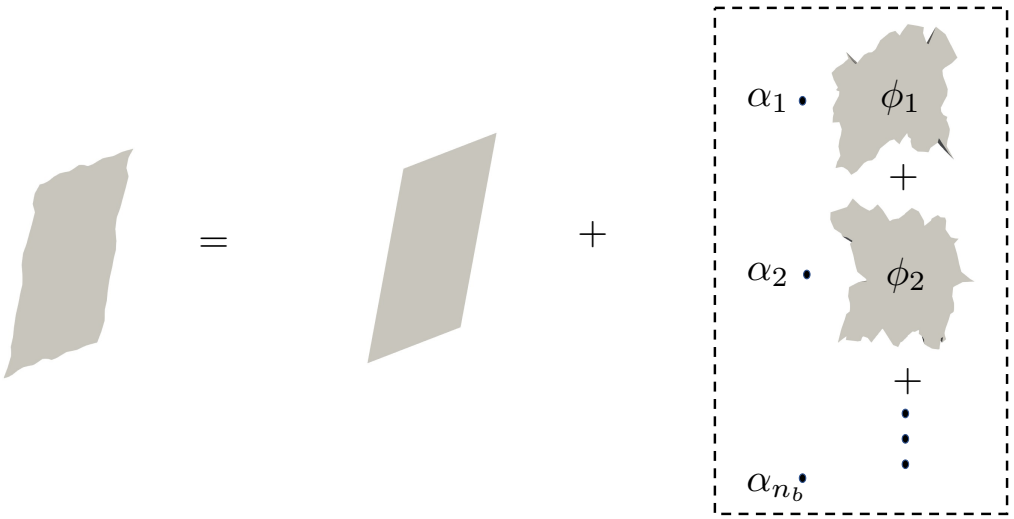


Figure 4.1: Illustration of the additive split of an RVE's displacement field in a homogeneous displacement field and a microstructurally fluctuating displacement field is employed in a projection-based MOR. The basis functions ($\underline{\underline{\Phi}}$) are only used for the fluctuating displacement field.

The macroscale deformation ($\mathbf{U}^M$) fully governs the homogeneous part of the displacement and as this is imposed by the macroscale in computational homogenization, $\underline{\bar{u}}$ is completely known. Consequently, only $\underline{\tilde{u}}$ needs to be computed. For this reason, the interpolation of projection-based MOR is not applied to the complete displacements as indicated in Eq. (4.3), but only to the fluctuating part:

$$\underline{\tilde{u}} \approx \underline{\underline{\Phi}}\,\underline{\alpha}. \tag{4.9}$$

The approximation of the complete displacements may then be written as:

$$\underline{u} \approx \underline{\underline{\Psi}}\,\underline{\beta} + \underline{\underline{\Phi}}\,\underline{\alpha}, \tag{4.10}$$

where $\underline{\underline{\Psi}}\,\underline{\beta}$ together denotes the known, homogeneous part of the displacements with:

$$\underline{\beta} = \begin{bmatrix} U^M_{xx} - 1 & U^M_{xy} & U^M_{yy} - 1 \end{bmatrix}^T, \tag{4.11}$$

where U^M_{xx}, U^M_{xy} and U^M_{yy} denote the three independent components of $\mathbf{U}^M$. Row i of $\underline{\underline{\Psi}}$ (of size $n_u \times 3$) is furthermore expressed as follows:

$$\left(\underline{\underline{\Psi}}\right)_{i,:} = \begin{cases} \begin{bmatrix} X_{\lceil \frac{i}{2} \rceil} & Y_{\lceil \frac{i}{2} \rceil} & 0 \end{bmatrix} & \text{if } i \text{ is odd,} \\ \begin{bmatrix} 0 & X_{\lceil \frac{i}{2} \rceil} & Y_{\lceil \frac{i}{2} \rceil} \end{bmatrix} & \text{if } i \text{ is even.} \end{cases} \tag{4.12}$$

This expression for $\underline{\underline{\Psi}}$ is only valid if the reference location of the bottom-left FE node of a 2D RVE is set to the origin, if the bottom-left FE node cannot displace, and if the displacement components in $\underline{u}$ are ordered as follows:

$$\left(\underline{u}\right)_i = \begin{cases} u_{\lceil \frac{i}{2} \rceil} & \text{if } i \text{ is odd,} \\ v_{\lceil \frac{i}{2} \rceil} & \text{if } i \text{ is even,} \end{cases} \tag{4.13}$$

where $\left(\underline{u}\right)_i$ denotes the i^{th} component of $\underline{u}$, and $u_{\lceil \frac{i}{2} \rceil}$ and $v_{\lceil \frac{i}{2} \rceil}$ denote the horizontal and vertical displacement of FE node $\lceil \frac{i}{2} \rceil$, respectively.

The expression for $\underline{\underline{\Phi}}$ depends on whether the POD or RB method is applied. However, both approaches have several aspects in common. First, the goal is that all columns (each column is a basis function) in $\underline{\underline{\Phi}}$ together encapsulate all characteristic features of the online solutions. Second, each column must be orthonormal, or at least orthogonal, with respect to the other columns (i.e. $\underline{\underline{\Phi}}^T \underline{\underline{\Phi}} = \underline{\underline{I}}$, where identity matrix $\underline{\underline{I}}$ is of size $n_b \times n_b$). Third, both approaches require numerous DNS to be performed for different load parameters. In this contribution, each set of load parameters is a sequence of $\mathbf{U}^M(t)$ (where t denotes a pseudo-time). From each training simulation, the solutions at several increments (after deducting the homogeneous displacements) are stored.

Proper-orthogonal-decomposition

In the method of proper-orthogonal-decomposition, the training solutions are stored as columns in a so-called snapshot matrix, $\underline{\underline{S}}$. Matrix $\underline{\underline{S}}$ is of size $n_u \times n_t n_s$, where n_t denotes the number of training simulations and n_s the number of solutions extracted from each training simulation. Singular value decomposition (SVD) is then applied to snapshot matrix $\underline{\underline{S}}$ and the left singular vectors with the n_b largest singular values are used as basis functions. Because the fluctuating displacement parts of the solutions ($\underline{\tilde{u}}$), which are stored in the snapshot matrix are periodic, the left singular vectors are periodic as well. (Furthermore, as the definition of the left singular vectors implies, the basis functions are orthonormal with respect to each other.)

More formally, the SVD of the snapshot matrix may be written as the following optimization problem:

$$\begin{bmatrix} \underline{\underline{P}}^* & \underline{\underline{D}}^* & \underline{\underline{Q}}^* \end{bmatrix} = \underset{\underline{\underline{P}},\underline{\underline{D}},\underline{\underline{Q}}}{\operatorname{argmin}} \| \underline{\underline{S}} - \underline{\underline{P}}\,\underline{\underline{D}}\,\underline{\underline{Q}}^T \|_F, \tag{4.14}$$

such that $\underline{\underline{D}}$ denotes a diagonal matrix of size $n_t n_s \times n_t n_s$ with entries $\left(\underline{\underline{D}}\right)_{11} \geq \left(\underline{\underline{D}}\right)_{22} \geq \left(\underline{\underline{D}}\right)_{33} \geq ... \geq \left(\underline{\underline{D}}\right)_{n_t n_s, n_t n_s}$ and

$$\underline{\underline{P}}^T \underline{\underline{P}} = \underline{\underline{Q}}^T \underline{\underline{Q}} = \underline{\underline{I}}, \tag{4.15}$$

where $\| \bullet \|_F$ denotes the Frobenius norm and matrices $\underline{\underline{P}}$, $\underline{\underline{Q}}$ and $\underline{\underline{I}}$ are of size $n_u \times n_t n_s$, $n_t n_s \times n_t n_s$ and $n_t n_s \times n_t n_s$, respectively. This optimization problem is so frequently used that each numerical software has its own dedicated function to compute it. The matrix with basis functions, $\underline{\underline{\Phi}}$, is obtained by only using the first n_b columns of $\underline{\underline{P}}^*$.

Another way to obtain the n_b left singular values (and faster if $n_u \gg n_t n_s$) is to apply eigenvalue decomposition to $\underline{\underline{\hat{S}}} = \underline{\underline{S}}^T \underline{\underline{S}}$ ($\underline{\underline{\hat{S}}}$ is symmetric and of size $n_t n_s \times n_t n_s$), which may be expressed by finding eigenvalues b^i and eigenvectors $\underline{b}^{*i}$ according to:

$$\left(\underline{\underline{\hat{S}}} - b^i \underline{\underline{I}}\right) \underline{b}^{*i} = 0, \tag{4.16}$$

$$\underline{b}^{*iT} \underline{b}^{*j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} = 0, \end{cases} \tag{4.17}$$

where $b^i > b^{i+1}$ and $\underline{\underline{I}}$ is of size $n_t n_s \times n_t n_s$. The matrix with the left singular values can then be computed as:

$$\underline{\underline{\Phi}} = \underline{\underline{S}} \begin{bmatrix} \underline{b}^{*1} & \underline{b}^{*2} & \underline{b}^{*3} & ... & \underline{b}^{*n_b} \end{bmatrix}, \tag{4.18}$$

where the eigenvalues of the associated eigenvectors are ordered according to $b^1 > b^2 > ... > b^{n_b}$. It may be clear that eigenvalue decomposition is so important that, similarly as for singular value decomposition, practically all numerical software packages have their own dedicated functions to efficiently compute it.

Reduced basis

The method of reduced-basis (in its empirical sense) also uses the $n_t n_s$ training solutions, but does not require to store them in a snapshot matrix. The RB method aims to find those n_b training solutions (whose numbers are stored in index set B^* of length n_b) of all $n_t n_s$ training solutions (whose numbers are stored in index set $A = \{1, 2, 3, ..., n_t n_s\}$, $B^* \subseteq A$) which, after being orthonormalized with respect to each other, together minimize the worst match with the training solutions.

It may thus be expressed as the following combinatorial optimization problem:

$$B^* = \underset{B \subseteq A}{\operatorname{argmin}} \max_{i \in A} \min_{\underline{a}} \|\underline{\tilde{u}}^i - \sum_{j \in B} \underline{\hat{\tilde{u}}}^j a^j\|_2, \tag{4.19}$$

where $\| \bullet \|_2$ denotes the L^2-norm, the length of $\underline{a}$ is n_b and the hat on $\underline{\hat{\tilde{u}}}_j$ denotes that the j^{th} training solution (whose index is stored in B, $j \in B$) is orthonormalized with respect to all other training solutions whose indices are in B, i.e.:

$$\underline{\hat{\tilde{u}}}^{jT} \underline{\hat{\tilde{u}}}^k = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{else} \end{cases} \quad \text{where} \quad j, k \in B. \tag{4.20}$$

The columns of $\underline{\underline{\Phi}}$ are thus those training solutions $\underline{\hat{\tilde{u}}}^j$ (after the homogeneous part is deducted from them and after being orthogonalized with respect to each other) whose indices are present in B^*.

It may be clear that the combinatorial optimization problem of Eq. (4.19) is non-deterministic polynomial-time (NP) hard, yielding enormous computational times. For this reason, the problem is often solved using some kind of greedy algorithm, in which one training solution at a time is added to the set of training solutions employed as basis functions. One greedy algorithm that avoids NP hardness can be written as follows:

Algorithm 1 RB- Greedy algorithm 1

$B^* = \{\}$, $\check{A} = A$.
for n_b times
  $k^* = \underset{k \in \check{A}}{\operatorname{argmin}} \max_{i \in A} \min_{\underline{a}} \|\underline{\tilde{u}}^i - \sum_{j \in B^*} \underline{\hat{\tilde{u}}}^{j+k} a^{j+k}\|_2$
  $\check{A} = \check{A} \setminus \{k^*\}$, $B^* = B^* \cup \{k^*\}$

where it may be clear that the length of $\underline{a}$ increases with one every time the for-loop is repeated.

Although the latter combinatorial optimization problem is not NP hard, it is still time consuming, because minimizer $\underline{a}$ must be computed numerous times ($n_t n_s (n_t n_s n_b - n_b(n_b-1)/2)$ times in fact). For this reason, another greedy algorithm may be exploited that avoids the computation of minimizer $\underline{a}$. This greedy algorithm can be explained by starting that a number of training solutions are already selected for use as basis functions. The question is which training solution of the remaining, unselected training solutions, is best to add. The assumption is that the best one to be added is the one that minimizes the projection with respect to the already selected training solutions. (This assumes that the training solution that minimizes the projection encompasses the most characteristic features that are not yet encapsulated in the already selected training solutions.)

This substantially faster greedy algorithm starts with normalizing each training solution (denoted by $\breve{\underline{u}}$). The remainder may then be concisely written as:

Algorithm 2 RB- Greedy algorithm 2

Draw one sample from $\mathsf{U}(1, n_b)$ as first number in B^*, $\breve{A} = A$.
for n_b times
$k^* = \underset{k \in \breve{A}}{\operatorname{argmin}} \sum_{j \in B^*} \hat{\underline{\breve{u}}}^{jT} \breve{\underline{u}}^k$
Orthonormalize $\breve{\underline{u}}^{k^*}$ with respect to $\hat{\underline{\breve{u}}}^j$ $(j \in B^*)$
$\breve{A} = \breve{A} \setminus \{k^*\}$, $B^* = B^* \cup \{k^*\}$

As the two aforementioned greedy algorithms obviously yield suboptimal basis functions (compared to the initial combinatorial optimization problem of Eq. (4.19)), one would typically need a larger number of basis functions (n_b) than if the POD method is applied. RB methods are nevertheless useful if n_u and $n_t n_s$ are substantially large, because then the singular value or eigenvalue decomposition in the POD method is time consuming.

System of linear equations to be solved in the online simulations

Now the components of the interpolation employed in projection-based MOR are defined, the resulting system of linear equations that must be computed for each iteration, for each increment, can be rewritten as:

$$\left(\underline{\underline{\Phi}}^T \underline{\underline{K}}_{int}(\underline{\underline{\Psi}}\underline{\beta} + \underline{\underline{\Phi}}\underline{\alpha}, \underline{z})\underline{\underline{\Phi}}\right) d\underline{\alpha} = \underline{\underline{\Phi}}^T \left(\underline{f}_{ext} - \underline{f}_{int}(\underline{\underline{\Psi}}\underline{\beta} + \underline{\underline{\Phi}}\underline{\alpha}, \underline{z}) - \underline{\underline{K}}_{int}(\underline{\underline{\Psi}}\underline{\beta} + \underline{\underline{\Phi}}\underline{\alpha}, \underline{z})\underline{\underline{\Psi}} d\underline{\beta}\right), \tag{4.21}$$

where we repeat once more that update $d\underline{\alpha}$ denotes the DoFs that must be computed. Furthermore, $d\underline{\beta} = \underline{0}$ during the application of Newton's method.

Although the stiffness matrix of the projection-based MOR ($\underline{\underline{\Phi}}^T \underline{\underline{K}}_{int} \underline{\underline{\Phi}}$) is full, instead of sparse as the stiffness matrix of the DNS, its size is $n_b \times n_b$, which is a significant reduction. It may also be noted that the periodicity constraints are not present anymore (Eq. (4.6)), because the basis functions are formed using the microstructurally fluctuating part of the training solutions, which are periodic. Consequently, the basis functions, $\underline{\underline{\Phi}}$, are also periodic, regardless if the POD method or the RB method is employed.

4.4 Clustering for adaptive basis selection

As mentioned before, conventional projection-based MOR is accurate for (hyper)elastic, but not for simulations using elastoplastic constitutive models. A solution in these days often sought in the clustering of the training simulations and to construct a group of basis functions for each cluster. During the course of an online simulation, a classification is then performed to decide from which cluster the basis functions must be used for a given time increment. Clustering training solutions has been demonstrated to substantially improve the accuracy of projection-based MOR for a range of different models [71, 49, 4], but not yet for simulations involving elastoplastic solids.

Clustering is typically performed in terms of the 'load' parameters that are considered in the projection-based MOR. To aid the concept of clustering of training solutions, we now consider the current macroscale deformation imposed to the RVE as the 'load' parameters, i.e. $\mathbf{U}^M(t^{cur})$ where t^{cur} denotes the current time (although due to the path-dependency of elastoplasticity, it makes more sense to consider both the past and currently imposed macroscale deformations, i.e. $\mathbf{U}^M(0 < t \leq t^{cur})$, as load parameters).

If we assume that each load path is discretized with 100 increments, we may consider storing all 100 combinations of the load parameters for each path, (where one, at t^{cur}, is given by $\mathbf{U}^M(t^{cur})$). We also consider that for all 100 time increments, a training solution is stored (i.e. $n_s = 100$). Each combination of load parameters is thus associated with one training solution (i.e. the microstructurally fluctuating part of the training solution in fact).

The assumption is now that if an online simulation is performed with the load path in a certain direction, the training solutions of load paths in completely different directions do not provide any useful characteristics to construct the online solutions for the current load path. Therefore, the idea is to only use training solutions of the load paths that are close to the load path of the online simulation for the construction of the current basis functions.

POD method per cluster

For now, we assume that clustering has taken place (based on the load parameters) and this has resulted in n_c clusters of training solutions and load parameters. The index set of the training solutions and load parameters used for the i^{th} cluster is denoted by B^{*i} such that $A = B^{*1} \cup B^{*2} \cup ... \cup B^{*n_c}$ and each index can only be present in one of the subsets (B^{*1} until B^{*n_c}). The $n_{spc/i}$ (spc stands for *training solutions per cluster*, $n_{spc/i} = \#B^{*i}$) training solutions and load parameters that belong to the i^{th} cluster are stored in snapshot matrix $\underline{\underline{S}}^i$ (of size $n_u \times n_{spc/i}$) and load parameter column $\underline{\mathbf{U}}^{Mi}$ (of length $n_{spc/i}$). In order to determine the matrix with the basis functions for the i^{th} cluster using the POD method (as performed here), denoted by $\underline{\underline{\Phi}}^i$, one must solve the following minimization problem:

$$\begin{bmatrix} \underline{\underline{P}}^{*i} & \underline{\underline{D}}^{*i} & \underline{\underline{Q}}^{*i} \end{bmatrix} = \underset{\underline{\underline{P}}^i, \underline{\underline{D}}^i, \underline{\underline{Q}}^i}{\operatorname{argmin}} \| \underline{\underline{S}}^i - \underline{\underline{P}}^i \, \underline{\underline{D}}^i \, \underline{\underline{Q}}^{iT} \|_F, \tag{4.22}$$

such that $\underline{\underline{D}}^i$ is a diagonal matrix of size $n_t n_s \times n_t n_s$ with entries $\left(\underline{\underline{D}}^i\right)_{11} \geq \left(\underline{\underline{D}}^i\right)_{22} \geq \left(\underline{\underline{D}}^i\right)_{33} \geq ... \geq \left(\underline{\underline{D}}^i\right)_{n_{spc/i} n_{spc/i}}$ and

$$\underline{\underline{P}}^{iT} \underline{\underline{P}}^i = \underline{\underline{Q}}^{iT} \underline{\underline{Q}}^i = \underline{\underline{I}}, \tag{4.23}$$

where $\underline{\underline{\Phi}}^i$ is constructed by extracting the n_b left singular vectors, i.e. the first n_b columns of $\underline{\underline{P}}^{*i}$. It can be noted that, although each cluster may in principle come with its own number of $n_{spc/i}$ basis functions, the same number of basis functions is used for each cluster in the current chapter.

Classification in the online phase

Also in the current subsection, we still assume that clustering has taken place, which has resulted in n_c clusters of training solutions and load parameters.

In an online simulation, one must decide to which cluster the load parameters of the current time increment ($\mathbf{U}^M(t^{cur})$) belongs, so that from that cluster the basis functions are used (which may change during the course of an online simulation). In order to answer this question, one must parametrize each cluster in the space of the load parameters. These parametrizations may for instance use one or more locations in the space of the load parameters. For the i^{th} cluster, these locations are stored in column $\breve{\underline{\mathbf{U}}}^{Mi}$. One then choses some classification function, r, that quantifies some measure between the current online load parameters ($\mathbf{U}^M(t^{cur})$) and the parametrization of the i^{th} cluster ($\breve{\underline{\mathbf{U}}}^{Mi}$).

The decision to which cluster the load parameters of the current time step in the online simulation belongs is then formulated as a combinatorial optimization problem, in the form of:

$$i^* = \underset{i\in\{1,2,\ldots,n_c\}}{\operatorname{argmin}} \; r(\mathbf{U}^M(t^{cur})|\breve{\underline{\mathbf{U}}}^{Mi}). \tag{4.24}$$

It may be clear that this combinatorial optimization problem must in principle be solved for each time step of the online simulation and therefore has an influence on the speed of the online simulation. It may then also be clear that this combinatorial optimization problem affects the speed of the online simulation more substantially if the number of employed clusters (n_c) is large and if the number of locations in the space of the load parameters needed to evaluate r (i.e. if the length of column $\breve{\underline{\mathbf{U}}}^{Mi}$) is large.

In the case of clustering using DBSCAN for instance, the load parameters used for parametrization in this chapter are the same as the clustered load parameters, i.e. $\breve{\underline{\mathbf{U}}}^{Mi} = \underline{\mathbf{U}}^{Mi}$ (for the i^{th} cluster). Classification function r in Eq. (4.24) is then defined as follows:

$$r(\mathbf{U}^M(t^{cur})|\breve{\underline{\mathbf{U}}}^{Mi}) = \min_{j\in B^{*i}} \|\mathbf{U}^M(t^{cur}) - \breve{\mathbf{U}}^{Mj}\|_2. \tag{4.25}$$

k-means clustering

To the best of the candidate's knowledge, unsupervised clustering of training solutions in projection-based MOR has mainly been performed using k-means clustering [58, 79, 29, 2]. k-means clustering has several advantages compared to other clustering approaches. First, it requires only one hyperparameter to be selected by the user: the number of clusters, n_c. Second, the clustering algorithm is fast. In other words, the speed of the computations in the offline stage is hardly affected by it. Third, the classification is fast, because classification function r in Eq. (4.24) only uses one location in the space of the load parameters to parametrize each cluster (i.e. the center of each cluster). Consequently, the online simulations are hardly affected by the classification.

k-means clustering aims to minimize the sum of the distances between each point and the center of the cluster to which the point belongs, where the center of a cluster is the average location of all points that belong to that cluster. This may mathematically be written as the following combinatorial optimization problem:

$$\left[B^{*1}, B^{*2}, B^{*3}, ..., B^{*n_c}\right] = \underset{B^1,B^2,B^3,...,B^{n_c}}{\operatorname{argmin}} \sum_{i=1}^{n_c} \sum_{j \in B^i} \|\mathbf{U}^{Mj} - \frac{1}{n_{spc/i}} \sum_{k \in B^i} \mathbf{U}^{Mk}\|_2. \quad (4.26)$$

It may be clear that the center of the i^{th} cluster, needed in classification function r, is then given by:

$$\breve{\mathbf{U}}^{Mi} = \frac{1}{n_{spc/i}} \sum_{k \in B^{*i}} \mathbf{U}^{Mk}. \quad (4.27)$$

The problem with the above optimization problem is its NP hardness. For this reason, most numerical software include dedicated (yet sub-optimal) functions for k-means clustering, whose steps can roughly be described as follows:

One of the main advantages of k-means clustering is that the classification in the online simulations is fast. The function r, that quantifies the measure between the current online load parameters ($\mathbf{U}^M(t^{cur})$) and the parametrization of the i^{th} cluster ($\underline{\breve{\mathbf{U}}}^{Mi}$) is the Euclidian distance between the current load parameter of the online simulation, $\mathbf{U}^M(t^{cur})$, and the center of the i^{th} cluster (meaning that $\underline{\breve{\mathbf{U}}}^{Mi}$ only includes one location, i.e. the length of column $\underline{\breve{\mathbf{U}}}^{Mi}$ is one).

$$r = \| \mathbf{U}^M(t^{cur}) - \breve{\mathbf{U}}^{Mi} \|_2, \quad (4.28)$$

Algorithm 3 K-Means clustering

Randomly select one point ($\mathbf{U}^M$) as the center of each cluster
($\check{\mathbf{U}}^{Mi}$ for the i^{th} cluster).
Repeat until cluster centers do not change anymore
Initialise all subsets to be empty: $B^{*1} = \{\}$, $B^{*2} = \{\}$, ..., $B^{*n_c} = \{\}$
For each point j: $i^* = \underset{i \in \{1,2,3,\ldots,n_c\}}{\text{argmin}} \|\mathbf{U}^{Mj} - \check{\mathbf{U}}^{Mi}\|_2 \qquad B^{*i} = B^{*i} \cup \{j\}$
For each cluster i: $\check{\mathbf{U}}^{Mi} = \frac{1}{\#B^{*i}} \sum_{k \in B^{*i}} \mathbf{U}^{Mk}$

where the equation must be read such that the independent components of $\mathbf{U}^M$ and $\check{\mathbf{U}}^{Mi}$ are stored in column format.

However, k-means clustering also has a number of disadvantages. An important one is that k-means clustering has difficulties with clustering distinct patterns. This is clearly visible in Fig. 4.2, in which clustering of two different data sets is presented: a set with random data and a set with clear patterns. Another disadvantage is that k-means clustering does not automatically determine how many clusters are optimal, as the number of clusters (n_c) is a hyperparameter that must be set by the user. This is especially a problem in the high-dimensional case, in which the clustering results are difficult to visually confirm.

DBSCAN: Density-based spatial clustering of applications with noise

Because of the disadvantages of k-means clustering, DBSCAN is also considered. Compared to k-means clustering, DBSCAN has several advantages. First, it is better capable to distinguish distinct patterns (see the images on the right in Fig. 4.2, where DBSCAN has captured the individual curves in the data). Second, it is able to distinguish outliers (see bottom left image in Fig. 4.2). Third, DBSCAN is hardly stochastic, meaning that every time the algorithm is applied (with the same values for the hyperparameters), the difference between the clusters is substantially smaller compared to k-means clustering.

DBSCAN requires the user to set two hyperparameters. The first one, which is typically denoted by ϵ, is an Euclidian distance that defines a neighbourhood around each point (a circle in 2D, a sphere in 3D). The second one is an integer, $n_{minp} > 2$, and is used to define *core* points. DBSCAN uses these two hyperparameters to subdivide all points in three types: $A = B^c \cup B^{nc} \cup B^o$, where each point can only be present in one subset and superscripts c, nc and o refer to *core* points, *non-core* points and *outliers*, respectively.

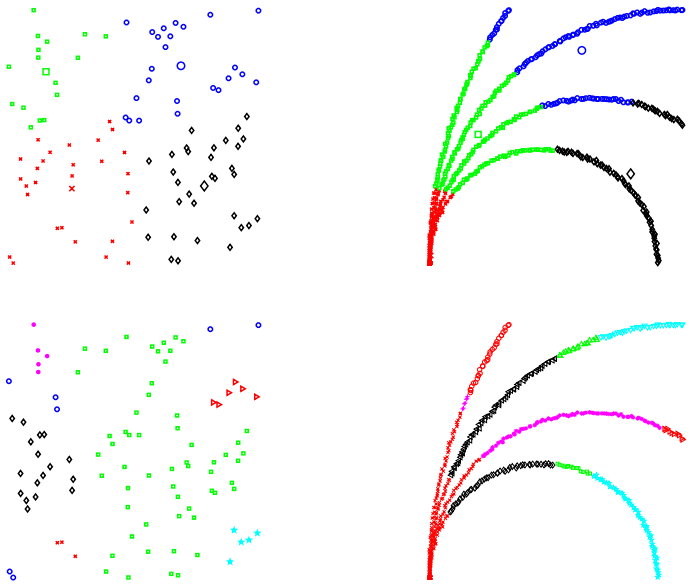


Figure 4.2: Illustrative comparison of k-means clustering and DBSCAN. Left column: random data set, right column: data set with patterns. Top row: k-means clustering for $n_c = 4$ (cluster centers are presented as the large shapes). Bottom row: DBSCAN clustering, which is better capable of distinguishing patterns (bottom right). Note that in the bottom left image, DBSCAN classifies the blue circles as outliers.

A *core* point is a point that has at least n_{minp} points in its neighbourhood (including the point itself), with ϵ as the 'radius' of the neighbourhood. In more detail, if the number of neighbours of point p, given by $\mathbf{U}^{Mp}$, is defined as follows:

$$n_{p\epsilon} = \#\Big\{i \in A, \text{such that } \|\mathbf{U}^{Mp} - \mathbf{U}^{Mi}\|_2 \leq \epsilon\Big\}, \tag{4.29}$$

then point p is a *core* point if $n_{p\epsilon} \geq n_{minp}$. Alternatively, point p is a *non-core* point $1 < n_{p\epsilon} < n_{minp}$. Finally, point p is an *outlier* if $n_{p\epsilon} = 1$.

With this classification as a start, DBSCAN performs the clustering. First, all outliers are dismissed (points that do not have a single other point in their neighbourhood). This means that *outliers* will not be part of any cluster.

Subsequently, one *core* point is randomly taken as the start of the first cluster. All other points in the neighbourhood of the first point are selected and will be part of the same cluster, regardless whether they are *core* or *non-core* points. However, for all neighbouring *core* points, the process is repeated: also for these points their

neighbouring points are selected to be part of the same cluster. The process of adding new points if they are in the neighbourhoods of *core* points is repeated until the cluster does not grow anymore.

When the first cluster does not grow anymore, a random *core* point that is not yet selected to be part of the first cluster will form the start of the next cluster and the same procedure as outlined in the previous paragraph is repeated until the second cluster does not grow anymore. This process is repeated until all *core* points belong to a cluster.

The clustering part of algorithm (excluding the classification into *core* points, *non-core* points and *outliers*) can also be written as follows (where ${}^{c}B^{n_c}$ and ${}^{nc}B^{n_c}$ denote the sets of *core* points and *non-core* points of the set that is currently being formed):

Algorithm 4 DBSCAN clustering

$n_c = 0$
while $\#B^c \neq 0$
  $n_c = n_c + 1$
  Randomly select a point, i, from B^c
  $B^c = B^c \setminus \{i\}$, ${}^{c}B^{n_c} = \{i\}$, ${}^{nc}B^{n_c} = \{\,\}$
  for $j \in {}^{c}B^{n_c}$
    $C^c = \left\{k \in B^c, \text{such that} \|\mathbf{U}^{Mj} - \mathbf{U}^{M}\|_2 \leq \epsilon\right\}$
    $C^{nc} = \left\{k \in B^{nc}, \text{such that} \|\mathbf{U}^{Mj} - \mathbf{U}^{M}\|_2 \leq \epsilon\right\}$
    $B^c = B^c \setminus C^c$, $B^{nc} = B^{nc} \setminus C^{nc}$, ${}^{c}B^{n_c} = {}^{c}B^{n_c} \cup C^c$, ${}^{nc}B^{n_c} = {}^{nc}B^{n_c} \cup C^{nc}$
  end
  $B^{n_c} = {}^{c}B^{n_c} \cup {}^{nc}B^{n_c}$
end
$B^o = B^o \cup B^{nc}$

One may thus summarize that:

- All *core* points belong to a cluster - even if a *core* point is the only *core* point of a cluster.
- The points in the neighbourhood of a *core* point are part of the same cluster as the *core* point itself.
- Conversely, a *non-core* point is not used to enlarge the cluster with its neighbouring points.
- Consequently, *non-core* points are reclassified as *outliers* if the only points in their neighbourhood are *non-core* points.

Similar to k-means clustering, DBSCAN is so often used that many numerical software include predefined functions to apply it. However, the implementation is of course more wisely performed than conceptually outlined above, in order to minimize the computational times.

Of course, DBSCAN also has a number of disadvantages. An important one is that, similar to k-means clustering, the clustering result is difficult to visually assess in case each point is governed by more than three dimensions. This makes it hard to optimize the values of the hyperparameters.

Complications for elastoplasticity

There are two issues that make the two aforementioned unsupervised clustering approaches difficult to employ for the clustering of training solutions of elastoplastic FE simulations. The first one is that when one time increment in the online simulation, let us denote this by t^1, belongs to cluster i and the next time increment, t^2, belongs to cluster j, the employed basis functions abruptly change from $\underline{\underline{\Phi}}^i$ to $\underline{\underline{\Phi}}^j$. Consequently, the online solution found at the end of t^1, $\underline{\tilde{u}}^{1*} = \underline{\underline{\Phi}}^i \underline{\alpha}^{1*}$ (where superscript $*$ refers to a converged solution at the end of a time increment), cannot serve as the initial guess for time increment t^2, because force equilibrium can then often not be achieved for time increment t^2.

One thus needs to first compute a suitable initial guess for time increment t^2. This is accomplished by two steps. First, one aims to find $\underline{\alpha}^2$ such that the L^2-norm between $\underline{\tilde{u}}^2 = \underline{\underline{\Phi}}^j \underline{\alpha}^2$ and $\underline{\tilde{u}}^{1*}$ is minimal. In other words, one first solves:

$$\underline{\alpha}^2 = \underset{\underline{\alpha}}{\operatorname{argmin}} \|\underline{\underline{\Phi}}^i \underline{\alpha}^{1*} - \underline{\underline{\Phi}}^j \underline{\alpha}\|_2. \tag{4.30}$$

Solving this minimization problem ensures that displacement fields $\underline{\tilde{u}}^{1*}$ and $\underline{\tilde{u}}^2$ match each other optimally. However, it also means that force equilibrium does not hold for initial guess $\underline{\tilde{u}}^2$. For this reason, an additional increment must be applied (for which no increment of $\mathbf{U}^M$ is applied), in which $\underline{\tilde{u}}^2$ is adjusted (obviously via adjusting $\underline{\alpha}^2$), as well as the history variables, such that force equilibrium is achieved.

This unavoidable procedure comes with two problems. The first one is that it is not at all guaranteed that force equilibrium can be found. Second, if force equilibrium is found, it is very well possible that $\underline{\tilde{u}}^2$ and the corresponding history variables have been adjusted so much that they do not match those of the DNS accurately anymore. (as we will see in the results section, the mismatch between the results of

the projection-based MOR (with clustering) and those of the DNS reduces during the next time increments to some extent, but it nevertheless causes substantial differences between the results of projection-based MOR with clustering and those of the DNS). For this reason, the next subsection describes several approaches that are investigated that either aim to decrease the differences between the basis functions of each cluster and its adjacent cluster, or aim to smooth the transition of the basis functions of one cluster to those of another cluster.

The second issue that makes unsupervised clustering difficult to employ for the grouping of training solutions of elastoplastic FE simulations involves the fact that not only the currently imposed macroscale deformation should be considered as the load parameters, but also the history of the imposed macroscale deformation. In other words, one should consider $\mathbf{U}^M(0 < t \leq t^{cur})$ instead of $\mathbf{U}^M(t^{cur})$ in order to define the clusters.

The complication that this change entails is that clustering must not be applied in two dimensions (in terms of $U^M_{xx}(t^{cur})$ and $U^M_{xy}(t^{cur})$), as we focus on isochoric macroscale deformations in 2D settings, i.e. $\det(\mathbf{U}^M)$, $\mathbf{U}^M$ only has two independent parameters), but in many more dimensions. This not only makes it hard to determine suitable values for the hyperparameters of the clustering algorithms (since visual inspection is impossible), it also makes the clustering results less useful. The reason for this is that the metric for clustering in both k-means clustering and DBSCAN is the Euclidian distance, which loses its efficiency if the number of dimensions is increased.

This chapter leaves the problem associated with clustering in high dimensions untouched. The reason for this is that the smoothing of the transition between clusters could not be appropriately overcome for monotonic loading (as will be demonstrated in the results section), and for monotonic loading it is sufficient to only consider the currently imposed macroscale deformation, $\mathbf{U}^M(t^{cur})$.

Smoothing the switch between clusters

The current subsection discusses three approaches that are investigated to ease the transition between clusters. The first two approaches discussed below aim to decrease the difference between the basis functions of each cluster. These approaches may be perceived as suboptimal, because by making sure that the differences between the different sets of basis functions reduce, one also ensures that each basis of a particular cluster is not optimal for that cluster. The third approach discussed below

aims to not compromize on each basis at all. The issue that occurs then is that force equilibrium can often not be found in a single additional increment. The third approach overcomes this by introducing several additional increments and adjusting cluster-wise weights for each increment.

Sharing training solutions between clusters

An approach that has been used in the literature [2] to ensure that the basis functions of the clusters do not vary too significantly with respect to each other is to construct the basis functions of a given cluster by not only using the training solutions associated with that cluster, but to also use training solutions of other clusters that are near the current cluster.

If we consider this approach for cluster i, it can be summarized as follows. First, n_d points from cluster i (B^{*i}) are selected that are furthest away from its cluster center, $\breve{\mathbf{U}}^{Mi}$. This entails that the points in B^{*i} must be ranked in ascending order according to the distance with the center.

As sets do not permit ordering, the notation here resorts to sequences. The indices in B^{*i} are now thus elements in sequence $(b_j^{*i})_{j\in\{1,2,\ldots,\#B^{*i}\}}$, and the ordering of the sequence is such that one can write for each pair of sequential elements, with $k_1, k_2 \in B^{*i}$:

$$\begin{matrix} b_j^{*i} = k_1 \\ b_{j+1}^{*i} = k_2 \end{matrix} \quad \text{such that} \quad \|\mathbf{U}^{Mk_1} - \breve{\mathbf{U}}^{Mi}\|_2 \geq \|\mathbf{U}^{Mk_2} - \breve{\mathbf{U}}^{Mi}\|_2. \tag{4.31}$$

We now compute the average distance of the n_d points that are furthest away from the i^{th} cluster center. In other words, we take the first n_d elements of $(b_j^{*i})_{j\in\{1,2,\ldots,\#B^{*i}\}}$ and compute their average distance from the cluster center, d^{mean}:

$$d^{mean} = \frac{1}{n_d}\sum_{j=1}^{n_d} \|\mathbf{U}^{Mb_j} - \breve{\mathbf{U}}^{Mi}\|_2. \tag{4.32}$$

Finally, we multiply the average distance with scalar $\rho > 1$ and search for all training solutions that do not belong to cluster i for which the load parameters are within distance $d^{mean}\rho$ from the i^{th}cluster center:

$$B^{*i_{add}} = \left\{ j \in A \setminus B^{*i} \quad \text{such that } \|\mathbf{U}^{Mj} - \breve{\mathbf{U}}^{Mi}\|_2 \leq d^{mean}\rho \right\}. \tag{4.33}$$

The training solutions of other clusters for which their load parameters fall within $d^{mean}\rho$ from the i^{th} cluster center are now used together with the training simulations of the i^{th} cluster (i.e. $B^{*i} \cup B^{*i_{add}}$) to construct its basis functions, $\underline{\underline{\Phi}}^i$, according to Eq. (4.22). In other words, snapshot matrix $\underline{\underline{S}}^i$ in Eq. (4.22) includes the training solutions whose indices are stored in both B^{*i} and $B^{*i_{add}}$.

Weighing the training solutions

Another approach to decrease the differences between the basis functions of the different clusters is also investigated. In this second approach, all training solutions are used for each cluster, but they are given a weight in the snapshot matrix. The weight is set to one for each training solution of the considered cluster, but decreases linearly with the distance away from the cluster.

In more detail, each snapshot matrix $\underline{\underline{S}}^i$ is now of size $n_u \times n_t n_s$ and the j^{th} column of the snapshot matrix can now be written as:

$$\left(\underline{\underline{S}}^i\right)_{:,j} = w^j \underline{\tilde{u}}^j. \tag{4.34}$$

Here, w^j denotes the weight associated with the j^{th} training solution. The value for this weight is now calculated as follows:

$$w^j = \begin{cases} 1 & \text{if } j \in B^{*i} \\ \eta^j & \text{else (i.e. } j \in A \setminus (B^{*i} \cup B^o) \end{cases}, \tag{4.35}$$

where B^o again denotes the index set with outliers in case of DBSCAN (which is empty for k-means clustering), and $0 \leq \eta^j \leq 1$ reads:

$$\eta^j = \frac{d^{*j} - d^{max}}{d^{max}}. \tag{4.36}$$

On the one hand, d^{*j} denotes the minimum distance between the j^{th} load parameters and all the load parameters that belong to the i^{th} cluster. In other words, one can write:

$$d^{*j} = \min_{k \in B^{*i}} \|\mathbf{U}^{Mj} - \mathbf{U}^{Mk}\|_2, \tag{4.37}$$

On the other hand, d^{max} denotes the maximum of all these minimum distances:

$$d^{max} = \max_{j \in A \setminus (B^{*i} \cup B^o)} d^{*j}. \tag{4.38}$$

Adaptively mixing the basis functions

As mentioned above, the previous two approaches have the disadvantage that they reduce the accuracy of the basis functions of a given cluster, for the benefit that the basis functions do not differ too much with the basis functions of other clusters. The final approach investigated here does not affect the basis functions of the clusters with the benefit that the basis functions are not compromized. Instead, basis functions $\underline{\underline{\Phi}}^i$ are entirely based on the training solutions associated with the i^{th} cluster. The problem that then often occurs is that no force equilibrium is found when the basis is changed, which would terminate the online simulation.

To alleviate this problem, the two-step procedure detailed in subsection 4.4 that aims to find a suitable initial guess when the basis functions are changed is formulated in an adaptive manner. First, the switch from $\underline{\underline{\Phi}}^i$ to $\underline{\underline{\Phi}}^j$ is attempted directly. If after a certain number of iterations force equilibrium has not been obtained, the two basis sets are mixed and an additional intermediate increment is used. If again, no force equilibrium is found, the two basis sets are mixed further and one or more additional increments are used. This procedure continues until force equilibrium is found, and until basis $\underline{\underline{\Phi}}^j$ is used on its own.

The so-called 'mixing' of the two basis sets $\underline{\underline{\Phi}}^i$ and $\underline{\underline{\Phi}}^j$ is written as follows:

$$\underline{\underline{\Phi}}^{ij} = \underbrace{w^i \underline{\underline{\Phi}}^i + w^j \underline{\underline{\Phi}}^j}, \tag{4.39}$$

with $w^i = 1 - w^j$ and where the underbrace is used to denote that the columns in matrix $w^i \underline{\underline{\Phi}}^i + w^j \underline{\underline{\Phi}}^j$ are orthonormalized with respect to each other (using the Gram-Schmidt algorithm).

Using this notation, this adaptive procedure can be summarized in more detail as follows:

Algorithm 5 Adaptively mixing basis functions

$w^j = 0$, $\Delta w^j = 1$, $k = 0$
while $w^j \neq 1$
  $w^j = w^j + \Delta w^j$
  $\underline{\underline{\Phi}}^{ij} = \underbrace{w^i \underline{\underline{\Phi}}^i + w^j \underline{\underline{\Phi}}^j}$
  Attempt 2-step procedure with $\underline{\underline{\Phi}}^{ij}$
  if no force equilibrium is found
    $w^j = w^j - \Delta w^j$
    $k = k + 1$
    $\Delta w^j = \frac{1}{2^k}$
  end
end

4.5 k-NN for adaptive basis selection

Clustering as discussed in the previous section clearly has its disadvantages. The disadvantage revealed in the results section below is that a substantial inaccuracy occurs when the basis functions of one cluster transition to those of another cluster. For this reason, the current section proposes an approach that avoids clustering altogether.

The ansatz of the approach is to use the n_b training solutions of which the load parameters are closest to the current load parameters in the online simulation directly as basis functions (after orthonormalization). In this way, the characteristic features of the employed training solutions can be expected to match those of the current solution highly accurately. This approach is thus a type of adaptive RB method, in which the basis functions are potentially changed at each time increment, based on a k nearest neighbour (k-NN) search of the load parameters.

In the previous section on clustering, the load parameters were considered to be the current macroscale deformation, $\mathbf{U}^M(t^{cur})$. This is sufficient for the particular combination of (1) clustering, and (2) monotonic loading. For the k-NN search, and in particular for the application to cyclic loading however, the load parameters require a more elaborate definition.

General quantification of the load parameters: curve matching of the load paths

For more general cases than monotonic loading, we propose to first quantify which load paths of the training simulations (i.e. 'training load paths') are most similar to the current load path of the online simulation (i.e. 'online load path'). Once the n_b most similar training load paths are determined, we chose one training solution per selected training simulation. The reason for this is that two subsequent training solutions of a single training simulation are highly similar to each other and hence, hardly any new information is provided by selecting a second training solution per training simulation. We will therefore first focus on quantifying the similarity between the online load path and the i^{th} training load path.

To measure the similarity between the online load path and the i^{th} training load path, we propose to measure the distance between the two load paths. To properly include the history, the distance should be measured over the entire online load path, so some integral form seems useful. One may therefore propose the following measure of similarity:

$$s^i = \int_0^{t^cur} \|\mathbf{U}^M(t) - \mathbf{U}^{Mi}(t)\|_2 dt, \tag{4.40}$$

where $\mathbf{U}^M(t)$ and $\mathbf{U}^{Mi}(t)$ denote the online load path and the i^{th} training load path, which are known in terms of pseudo-time t. The sketch on the left in Fig. 4.3 illustrates this measure of similarity. (Again, even though $\mathbf{U}^M$ is written in tensor format, one should consider the components of $\mathbf{U}^M$ in column format in the above expression.)

The convenience of the expression above is that $\mathbf{U}^M$ and $\mathbf{U}^{Mi}$ are indeed known in terms of pseudo-time t. The disadvantage is that the length of the online load path may not be the same as the length of training load path at pseudo-time t^{cur}. In other words, even if both load paths would follow the same path but at different speeds (given by t), measure of similarity s^i will not be zero. For viscoelastoplasticity this may be desired, but for rate-independent elastoplasticity as considered here, Eq. (4.40) will likely introduce some amount of error in the measure of similarity.

Generally, it may therefore be more appropriate to use the following measure of similarity:

$$s^i = \int_0^{l^c ur} \|\mathbf{U}^M(l) - \mathbf{U}^{Mi}(l)\|_2 dl, \tag{4.41}$$

where $l^c ur$ denotes the length of the online load path at current pseudo-time t^{cur} and l denotes the length parameter of both load paths. This measure of similarity is illustrated on the right in Fig. 4.3.

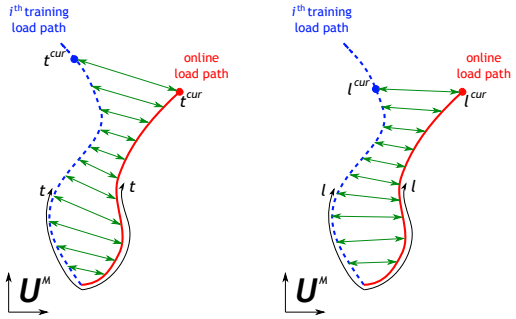


Figure 4.3: Illustrations for the measure of similarity of Eq. (4.40) on the left and of Eq. (4.41) on the right. The blue dashed curves denote a part of the the i^{th} training load path and the red curves the online load path. The measure of similarity is the integral of the distance between the two load paths, which we have attempted to illustrate by the green arrows.

Besides the fact that it is not trivial to integrate Eq. (4.41) (nor Eq. (4.40) for that matter), one must also determine at which pseudo-time t^k, the length of the i^{th} training load path is the same as the current length of the online load path. To this end, one first needs to integrate the online load path, from the start to the current time:

$$l^{cur} = \int_0^{t^{cur}} \|\frac{\partial \mathbf{U}^M(t)}{\partial t}\|_2 dt, \tag{4.42}$$

and then determine at which pseudo-time t^k the length of the i^{th} training load path is the same:

$$l^{cur} = \int_0^{t^k} \|\frac{\partial \mathbf{U}^{Mi}(t)}{\partial t}\|_2 dt. \tag{4.43}$$

If one assumes, that both pseudo-time t^k and Eq. (4.41) is numerically accurately integrated, one may then rank all measures of similarity, s^i ($i \in \{1, 2, ..., n_s\}$) from low to high and select the best n_b training simulations. For each selected training simulation, the question is then which solution to use as basis function (after orthonormalizing them altogether). Since generally no training solution is available at pseudo-time t^k, logical choices would be to either select the training solution of which its time signature is nearest to t^k, or one constructs some weighted average using the two training solutions that are available before and after pseudo-time t^k.

In the result section below, the investigated test cases are monotonic loading and particular case of cyclic loading. For both cases, the general quantification as discussed above is not used in order to accelerate the k-NN search, since we consider the aim of the current contribution to compare the capabilities of the k-NN-aided MOR with clustering-aided MOR.

Approximated quantification for monotonic loading

In case of monotonic loading, both the training load paths and the online load paths are assumed to be straight lines. Therefore, the employed approximation for monotonic loading only uses the current macroscale deformation of the online simulation, $\mathbf{U}^M(t^{cur})$, instead of the full history of the macroscale deformation (i.e. instead of load path $\mathbf{U}^M(0 < t \leq t^{cur})$). For each training simulation, we then search for the nearest macroscale deformation for which a training solution is available:

$$s^i = \min_{j \in \{1,2,...,n_t\}} \|\mathbf{U}^M(t^{cur}) - \mathbf{U}^{Mi}(t^j)\|_2. \tag{4.44}$$

Subsequently, we rank the distances (i.e. measures of similarity) and use the training solution of each training simulation that minimizes the distance.

Approximated quantification for cyclic loading

The character of the cyclic loading investigated in the result section is rather restricted. The limitations are that (1) one cycle of loading and unloading is considered, (2) loading always takes place until an edge of the domain (defined in terms of the components of $\mathbf{U}^M$), and (3) the same number of increments is considered for the loading part and the unloading part (in both the training and online simulations).

Because of these limitations and the preliminary nature of this chapter, Eq. (4.40) is used to quantify the similarity between the online load path and that of a training simulation (instead of Eq. (4.41), which was argued to be more suitable). In order to accelerate the computational efficiency of the k-NN search furthermore, the numerical integration employed to evaluate the measure of similarity of Eq. (4.40) lacks any weights:

$$s^i = \sum_{j=1}^{n^{cur}} \|\mathbf{U}^M(t^j) - \mathbf{U}^{Mi}(t^j)\|_2 dt, \tag{4.45}$$

where n^{cur} denotes the number of increments used to reach pseudo-time t^{cur}. In case s^i is within the first n_b measures of similarity, the training solution associated with $\mathbf{U}^{Mi}(t^{cur})$ is used.

Basis construction

The procedure to construct the basis functions for the current time increment of the online simulation, $\underline{\underline{\Phi}}$ (of size $n_u \times n_b$), based on the measures of similarity and the k-NN search described above is rather straightforward.

$$\underline{\underline{\Phi}} = \underbrace{\left[\underline{\tilde{u}}^1 \quad \underline{\tilde{u}}^2 \quad \dots \quad \underline{\tilde{u}}^{n_b}\right]}, \tag{4.46}$$

where $\underline{\tilde{u}}^i$ denotes the solution selected from the training simulation that is the i$^{\text{th}}$ most similar to the online simulation. The underbrace furthermore denotes that orthonormalisation is performed and the tilde again refers to the fact that the microstructurally fluctuating part of the training solution. It can furthermore be noted that orthonormalisation takes place for each time increment, even if the sequence of the measures of similarities for the training simulations remains identical. The reason for this is that the solutions that are employed from the training simulations generally differ (slightly).

Challenges, hypotheses and conditions

The approach as outlined above comes with several hypotheses, which must simultaneously hold if the approach is to be both accurate and fast:

- Most of the n_b training solutions that are used as basis functions in current time increment t^{cur} will also be used in next time increment t^{cur+1}. In other words, most of the columns in $\underline{\underline{\Phi}}^{cur}$ are the same as in $\underline{\underline{\Phi}}^{cur+1}$. On top of that, the basis functions that are replaced, will be replaced by those that are rather

similar. The requirement for this is that enough training simulations must be performed.

- Because only a few training solutions are used (i.e. n_b is a small number) and because only a few training solutions are replaced, the orthonormalization of the newly added training solutions with respect to the remaining training solutions is sufficiently fast. This is a requirement because basis functions may be replaced often, meaning that the orthonormalization must also be often performed.

- Instead of requiring a two-step approach to recalibrate the initial guess for each time increment in an online simulation, only the displacement field will be adjusted using the minimization problem of Eq. (4.30). It is assumed that the change of the displacement field is so minor that it induces a negligible discrepancy in the associated force equilibrium. This is a requirement because basis functions may be replaced very often, and the minimization problem of Eq. (4.30) is fast to compute, whereas recalibrating force equilibrium is time consuming.

- The k-NN search does not take much time. The requirement for this is that not too many training simulations are performed.

The main challenge of this k-NN adaptive RB method is that the hypotheses have competing requirements. The shared requirement of the first three hypotheses is that many training simulations are performed, whereas the requirement of the last hypothesis is that not too many training simulations are performed.

4.6 Results and discussion

In this section, the results of the above discussed frameworks are presented and compared with those of the DNS and a conventional projection-based MOR based on the POD method. The MOR approaches (in table 4.1) employing clustering are only compared for the monotonic loading of an RVE, because even for this relatively straightforward case, the results are deemed inaccurate. Cyclic loading is therefore only investigated for the MOR using the $k-$NN search.

In the next subsection, one may note that manual clustering would actually provide better clustering results for monotonic loading than the aforementioned machine learning approaches. This is however not the case for cyclic loading and because

Clustering algorithm	Sharing solutions	Weighing solutions	Mixing solutions
K-Means	Eq. (4.33)	Eq. (4.35)	Eq. (4.39)
DBSCAN		Eq. (4.35)	Eq. (4.39)

Table 4.1: The MOR approaches employing clustering.

monotonic loading is more straightforward to analyze, clustering for monotonic loading presents a valuable investigation. One of the reasons that monotonic loading is more straightforward from a clustering perspective is that the path-dependency of the deformation can be ignored for monotonic loading, which is impossible for cyclic loading, i.e. $\mathbf{U}^M(t^{cur})$ is considered for monotonic loading, whereas $\mathbf{U}^M(0 < t \leq t^{cur})$ is considered for cyclic loading.

Comparison for monotonic loading

The discretized RVE considered for monotonic loading is presented in Fig. 4.4. It consists of an elastoplastic matrix material with voids. The mechanical parameters are set to $E = 1$, $\nu = 0.3$, $M_0 = 0.01$, $h = 0.01$ and $n = 1.05$ (see Eq. (4.1) and Eq. (4.2)). 40 training simulations are performed using the DNS. The load paths of these training simulations are presented as red dashed lines in Fig. 4.5. Four verification simulations are considered, with their load paths presented by the black lines in Fig. 4.5. The domain of the load paths is given by $0.95 \leq \det(\mathbf{U}^M) \leq 1.25$, $0.5 < U^M_{xx} \leq 1.5$, $0.5 < U^M_{yy} \leq 1.5$ and $-0.5 < U^M_{xy} \leq 0.5$, which is indicated by the blue curves in Fig. 4.5. All types of simulations are performed using 1000 increments. Instead of using all available training solutions for the identification of the basis functions, only the training solutions of the first increment and every multitude of 50 increments are used (i.e. 840 training solutions are included). Furthermore, all types of MOR only employ 10 basis functions.

Fig. 4.6 presents some results for one of the verification simulations with k-means clustering. The clustering results for different numbers of clusters are presented in the left column of Fig. 4.6, whilst one of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress predicted for the associated clustering result is presented on the right (with that of k-NN-aided MOR for comparison). The stress-deformation responses demonstrate that the so-called mixing of the training solutions provides the best results, because it does not use training solutions of other clusters in the construction of the basis functions of each cluster. The results also reveal that the so-called sharing of training solutions between clusters (as formulated in [2]) performs better than the so-called weighing of the training solutions. In fact, sharing the

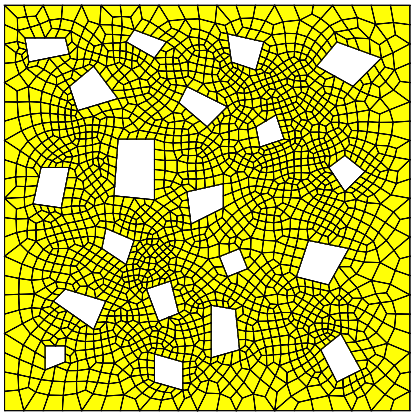


Figure 4.4: The discretized RVE with voids.

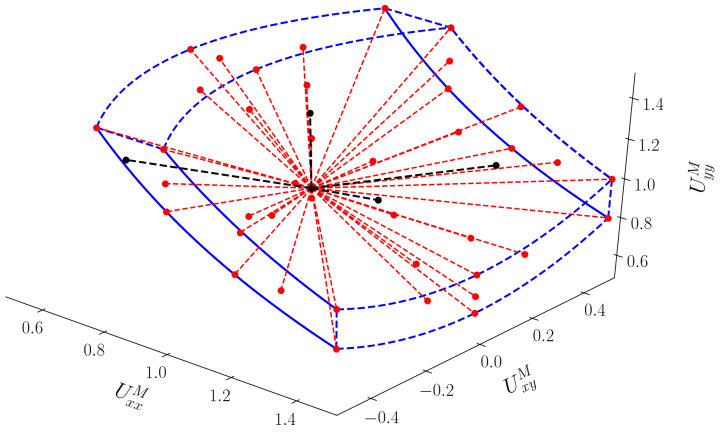


Figure 4.5: Training load paths for all monotonic loading simulations shown in red. The black curves present the load path of the verification simulations.

training solutions yields similar or better results than the conventional POD-based MOR, whereas weighing the training solutions yields similar or worse results than the conventional POD-based MOR.

Although almost all k-means-assisted POD-based MOR simulations associated with Fig. 4.6 involve the switching of the basis functions during the course of the verification simulation (with the exception of $n_c = 3$), the switching itself does not cause a substantial inaccuracy at the moment that the basis functions are changed. For the other verification simulation however, the switching of the basis functions compro-

mizes the quality of the results substantially, as can be seen in Fig. 4.7. Figs. 4.6 and 4.7 demonstrate that all enhancements considered to facilitate the transition between clusters do not make k-means-assisted MOR robustly outperform the conventional POD-based MOR.

On the other hand, the k-NN-assisted projection-based MOR outperforms the conventional POD-based MOR for both verification simulations. The k-NN-assisted MOR is especially accurate for the second verification simulation (compared to the POD-based MOR), because the load paths of the training simulations (from which the k-NN-assisted MOR picks its solutions as basis functions) are substantially closer to the load path of the second verification simulation than the load path of the first verification simulation.

The results obtained using DBSCAN are presented in Figs. 4.8 and 4.9 for verification simulations 1 and 2, respectively (again with those of k-NN-aided MOR for comparison). Compared to k-means clustering, DBSCAN is better capable to capture the distinct patterns of the data, because all training solutions of a particular training simulation at least belong to the same cluster. Thanks to this, no switching between the clusters occurs for the purely monotonically increasing load paths of the current subsection, which drastically compromized the responses predicted with the help of k-means clustering. Nevertheless, when inspecting Figs. 4.8 and 4.9, one cannot but conclude that DBSCAN does not consistently improve the results. Indeed for some cases, and in particular for those of the second verification simulation in Fig. 4.9, the predicted stresses are better than those predicted by the conventional POD-based MOR, but for a substantial number of others, the results are actually worse.

To understand the reason for this, we consider the case of three clusters for the first verification simulation, i.e. the two top diagrams in Fig. 4.8. The load path of this first verification is closest to one of the training simulations of the blue cluster and therefore uses the basis functions of the blue cluster (note that the verification simulation is not on top of the training simulation, cf. Fig 4.5). The issue here is that all other load paths of the blue cluster are in the top right area of the domain and therefore, the basis functions of the blue cluster are more suited for load paths in the top right area of the domain than the load path of the first verification simulation.

This phenomenon does not always occur. For $n_c = 10$ and $n_c = 15$ for the first verification load path for instance (i.e. the third and fourth rows in Fig. 4.8), the verification load path is very close to the load paths of a few training simulations

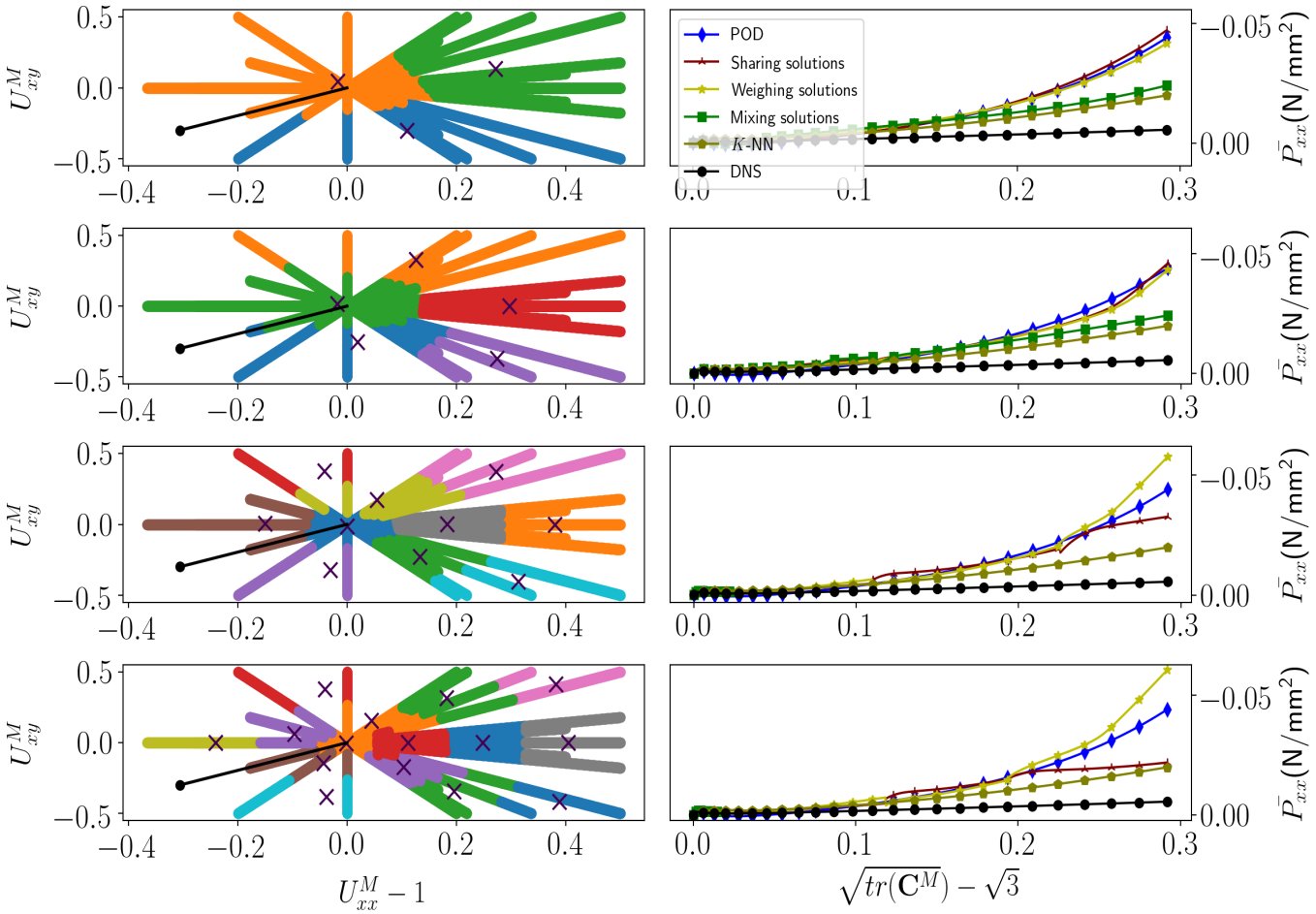


Figure 4.6: Monotonic loading: The results for the DNS, conventional POD-based MOR, k-means clustering (with sharing the training solutions, weighing the training solutions and mixing the training solutions) for verification simulation 1 using 10 basis functions and different numbers of clusters. Left column: k-means clustering results together with the load path of verification simulation 1. Right column: One of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress tensor as predicted by the different frameworks. Row 1: $n_c = 3$, row 2: $n_c = 5$, row 3: $n_c = 10$ and row 4: $n_c = 15$.

that are all part of the same cluster. For those cases, the DBSCAN-aided POD-based MOR with the sharing of the training solutions outperforms the conventional POD-based MOR. However, a robust improvement of the results cannot be guaranteed with DBSCAN (nor with k-means clustering).

Although a sufficient amount of results are already considered to conclude none of the clustering approaches systematically improves the results relative to those of the conventional POD-based MOR, whereas the k-NN-aided MOR outperforms the

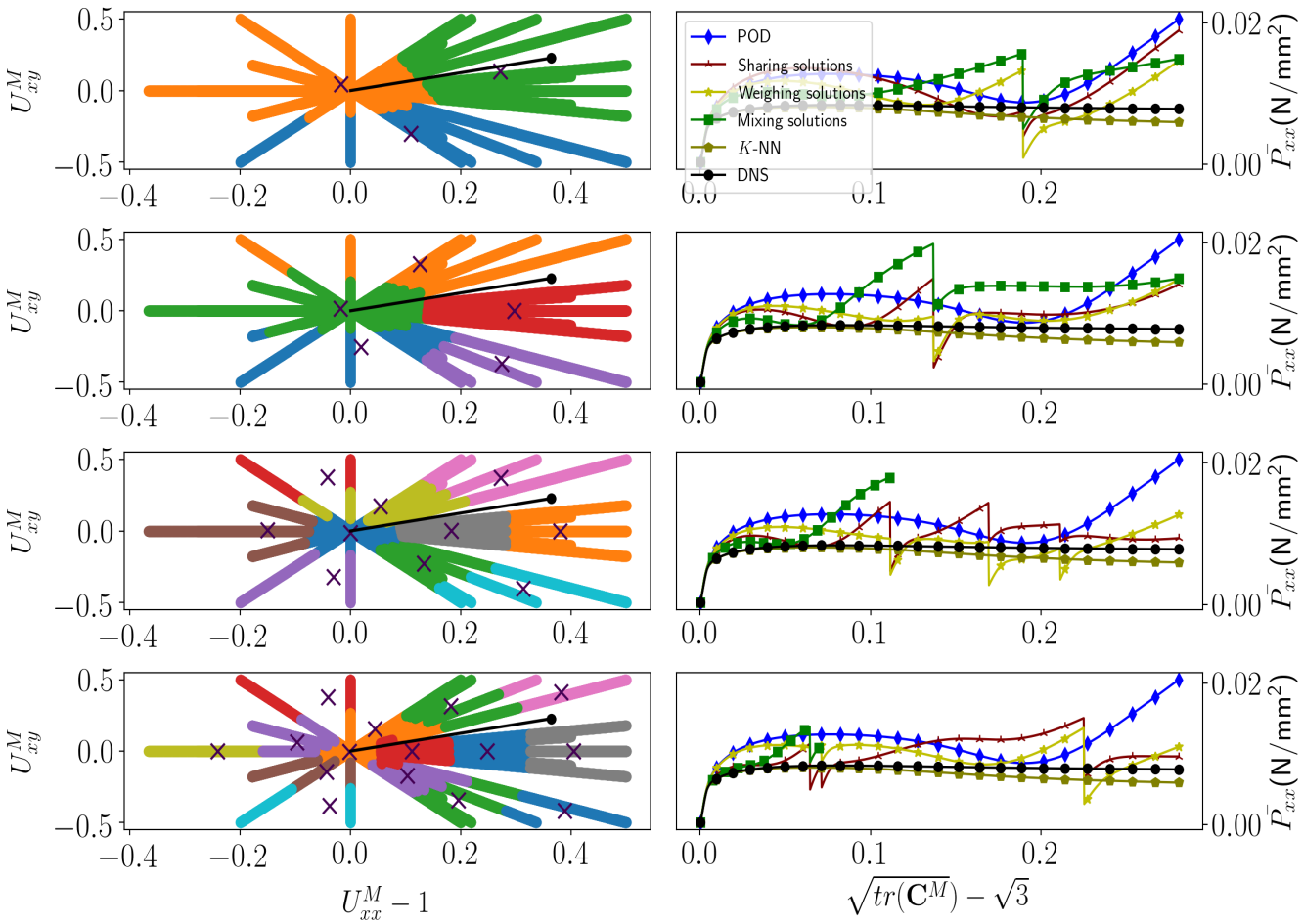


Figure 4.7: Monotonic loading: The results for the DNS, conventional POD-based MOR, k-means clustering (with sharing the training solutions, weighing the training solutions and mixing the training solutions) for verification simulation 2 using 10 basis functions and different numbers of clusters. Left column: k-means clustering results together with the load path of verification simulation 2. Right column: One of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress tensor as predicted by the different frameworks. Row 1: $n_c = 3$, row 2: $n_c = 5$, row 3: $n_c = 10$ and row 4: $n_c = 15$.

conventional MOR, one last verification simulation is considered. The load path of this last verification simulation is constructed such that one switch of the basis functions occurs for DBSCAN. The reason for this is that in the higher-dimensional case of cyclic loading, switching of the basis is highly likely to occur for the DBSCAN-aided POD-based MOR and we want to investigate whether a temporal inaccuracy occurs at the moment the switching of the basis takes place (as observed for k-means-aided MOR).

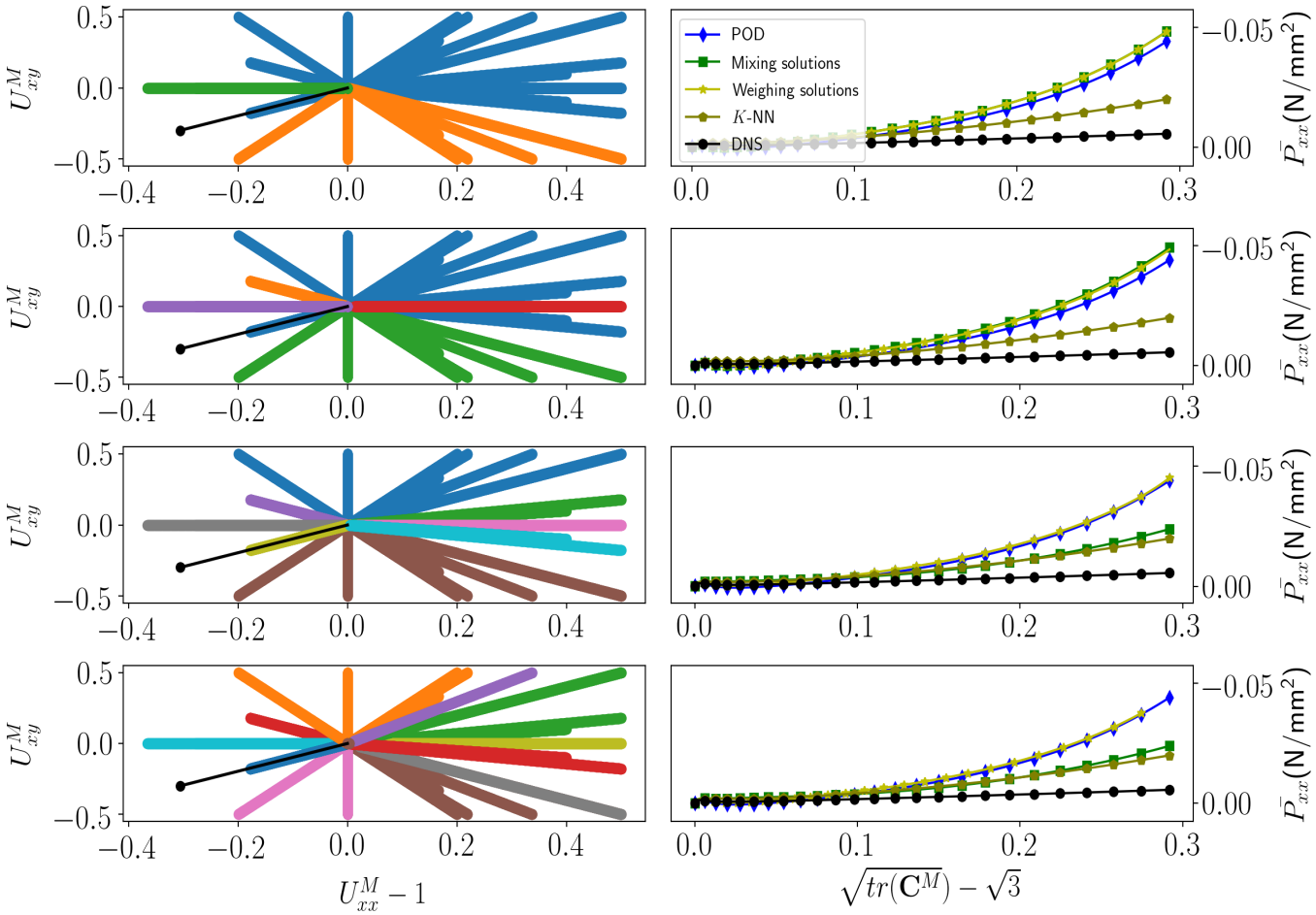


Figure 4.8: Monotonic loading: The results for the DNS, conventional POD-based MOR, DBSCAN clustering (with sharing the training solutions and weighing the training solutions) for verification simulation 1 using 10 basis functions and different numbers of clusters. Left column: DBSCAN clustering results together with the load path of verification simulation 1. Right column: One of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress tensor as predicted by the different frameworks. Row 1: $n_c = 3$, row 2: $n_c = 5$, row 3: $n_c = 10$ and row 4: $n_c = 15$.

The load path that forces a switch of the basis for DBSCAN is presented in Fig. 4.10. The associated stress response shows that, similarly to the case of k-means clustering, if a switch occurs, the predicted stress is temporarily highly inaccurate.

k−NN search for cyclic loading

Because all the POD-based MOR approaches aided by clustering are not systematically better than the conventional MOR for monotonic loading, they are not considered for cyclic loading. Instead, only the RB-based MOR aided by k-NN searching is investigated for cyclic loading.

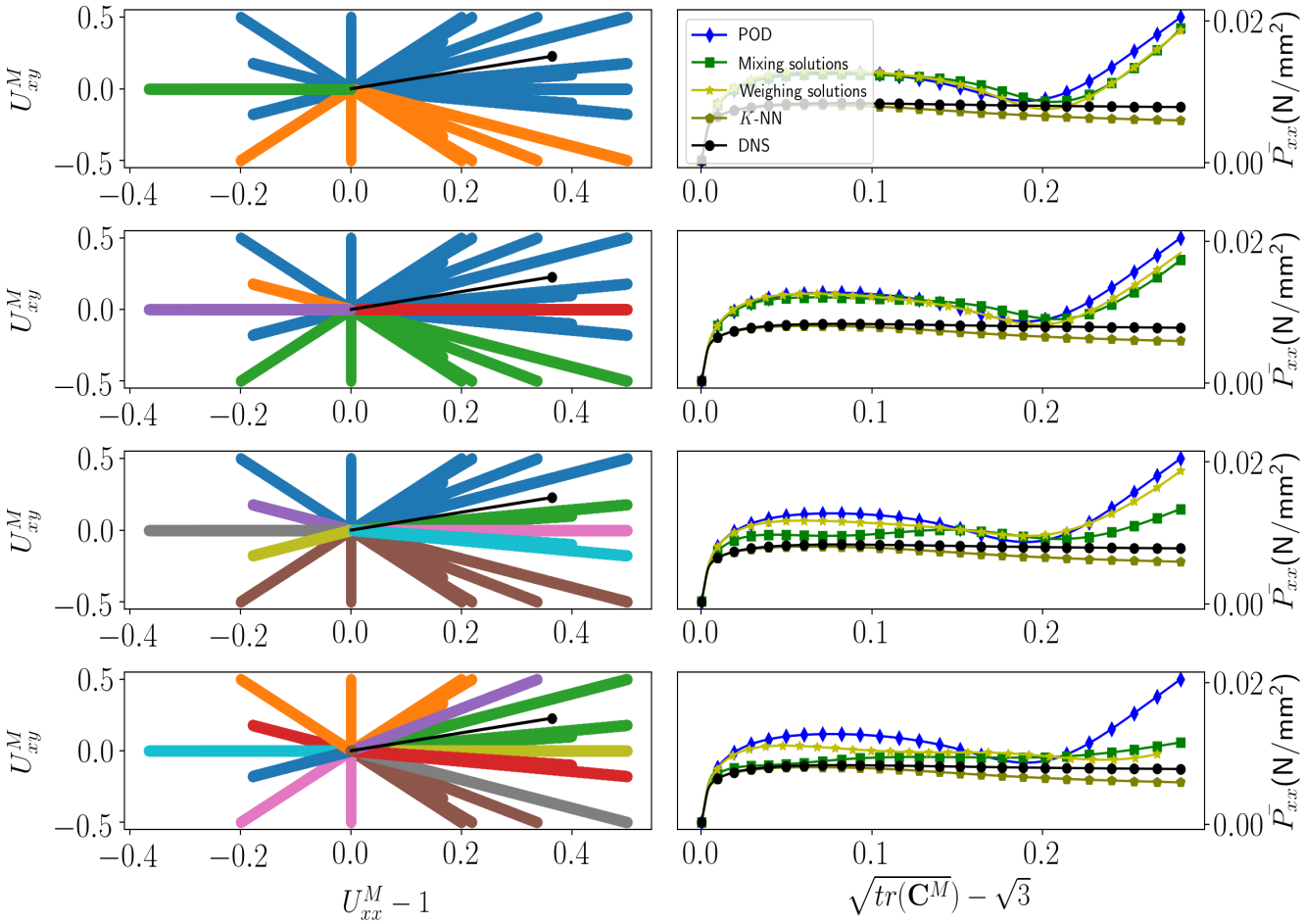


Figure 4.9: Monotonic loading: The results for the DNS, conventional POD-based MOR, DBSCAN clustering (with sharing the training solutions and weighing the training solutions) for verification simulation 2 using 10 basis functions and different numbers of clusters. Left column: DBSCAN clustering results together with the load path of verification simulation 2. Right column: One of the components of the homogenized 1st Piola-Kirchhoff stress tensor as predicted by the different frameworks. Row 1: $n_c = 3$, row 2: $n_c = 5$, row 3: $n_c = 10$ and row 4: $n_c = 15$.

The discretized RVE considered for cyclic loading is presented in Fig. 4.11. It is essentially the same as the one considered for monotonic loading, except that the elastoplastic matrix includes stiff elastic particles instead of voids. The reason for this change is that our implementation lacks an arc-length solution algorithm, which would be necessary for cyclic loading with voids. The mechanical parameters of the matrix are set to $E = 1$, $\nu = 0.3$, $M_0 = 0.01$, $h = 0.02$ and $n = 1.05$. The elastic properties of the stiff elastic particles are set to $E = 20$ and $\nu = 0.3$ and an initial yield stress of $M_0 = \infty$ prevents the particles from deforming plastically.

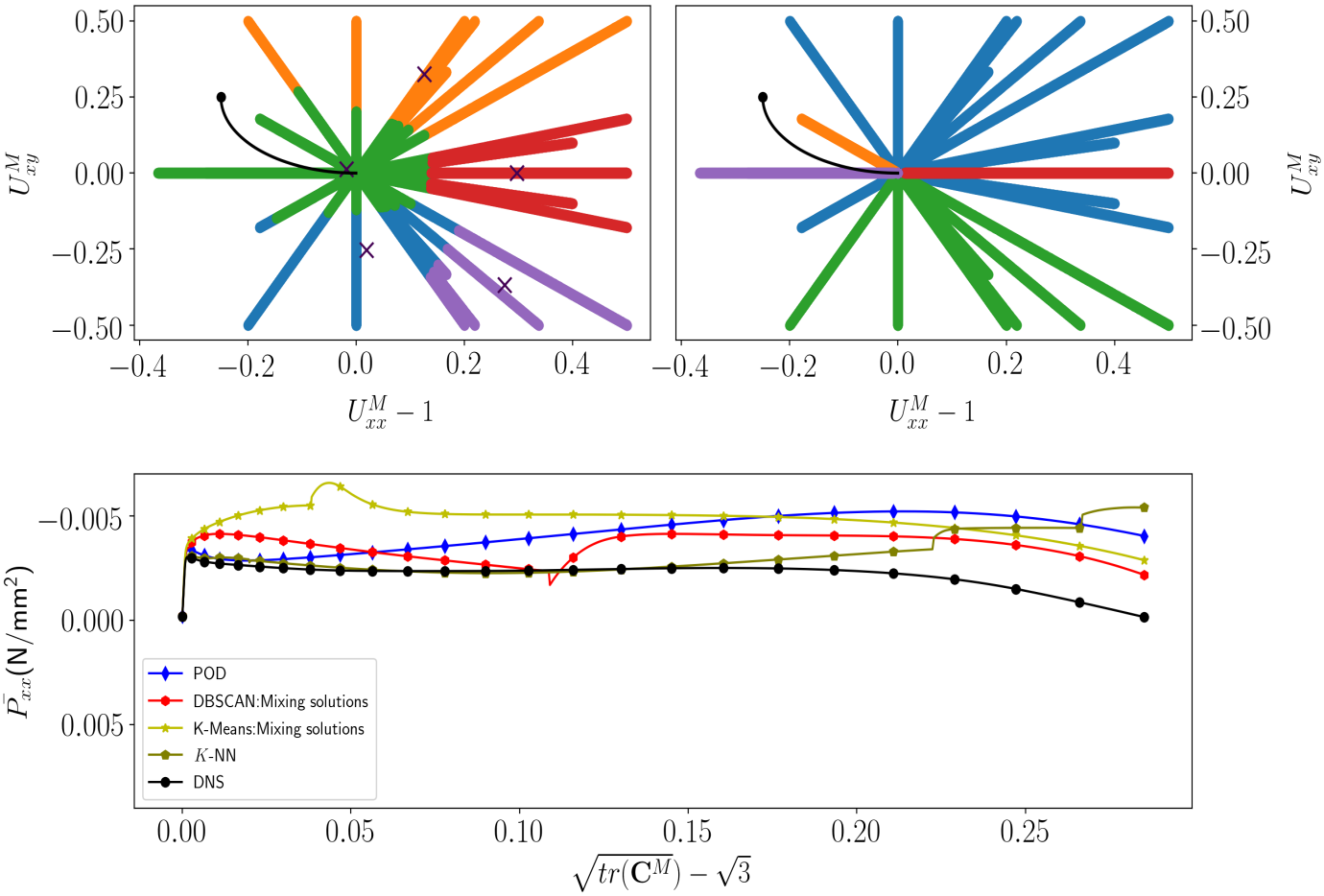


Figure 4.10: Monotonic loading: The results for the DNS, conventional POD-based MOR, DBSCAN (with mixing the training solutions), k-means clustering (with mixing the training solutions) and k-NN search for verification simulation 3 using 10 basis functions with five clusters of training solutions for DBSCAN and k-means. Top-left: k-means clustering results for $n_c = 5$ together with the load path of verification simulation 3. Top-right: DBSCAN clustering results for $n_c = 5$ with the load path of verification simulation 3. Bottom: One of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress tensor as predicted by the different frameworks.

Because the matrix deforms mostly plastically, and plastic deformation is isochoric, and because the deformation of the particles is minimal due to their high Young's modulus relative to that of the matrix, only isochoric macroscale deformations are considered in the training and verification simulations: $\det(\mathbf{U}^M) = 1$. The domain of the load paths of the training and verification simulations is furthermore given by $0.5 < U^M_{xx} \leq 1.5$, $0.5 < U^M_{yy} \leq 1.5$ and $-0.5 < U^M_{xy} \leq 0.5$. However, because

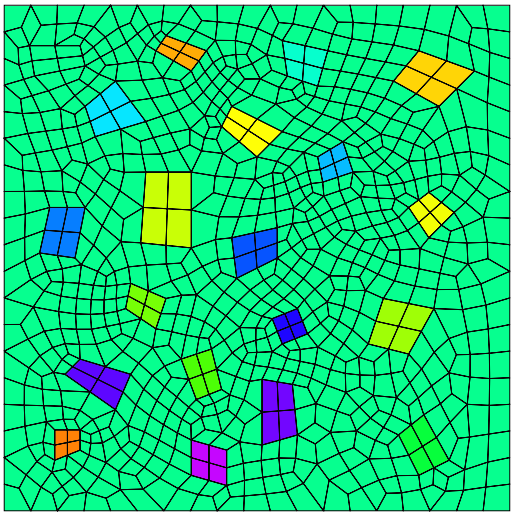


Figure 4.11: The discretized RVE with stiff elastic particles.

only isochoric deformation is considered, macroscale deformation gradient tensor $\mathbf{U}^M$ only consists of two independent scalars (e.g. if U^M_{xx} and U^M_{xy} are known, U^M_{yy} can directly be calculated).

3000 training simulations are performed using the DNS. 1000 increments are used to subdivide each training simulation (and verification simulation); 500 for the loading phase and 500 for the unloading phase. Loading always take place until a boundary of the domain is reached (given by $0.5 < U^M_{xx} \leq 1.5$, $0.5 < U^M_{yy} \leq 1.5$ and $-0.5 < U^M_{xy} \leq 0.5$) and unloading always takes place until no macroscale deformation remains (i.e. until $\mathbf{U}^M = \mathbf{I}$). The direction of each load path in the U^M_{xx}-U^M_{xy}-plane is parametrized by a single scalar (θ in Fig. 4.12), which is sampled from a uniform distribution with bounds 0 and 2π. The curvature of the loading part is given by a radius, which is sampled from a bimodal uniform distribution between bounds -2 and -0.5, and 0.5 and 2. This entails that if a negative radius is generated, the loading part of the path is placed on the opposite side of the direction given by θ in Fig. 4.12. The same sampling is performed for the unloading part, but this is not presented in Fig. 4.12. Three of such load paths are presented on the left in Fig. 4.13 (although these load paths are for the verification simulations and their presentation does not distinguish the loading and the unloading phase).

A single component of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress for the three load paths on the left in Fig. 4.13 is presented on the right in Fig. 4.13. Both the conventional POD-based MOR and the k-NN search-aided MOR use 10 basis

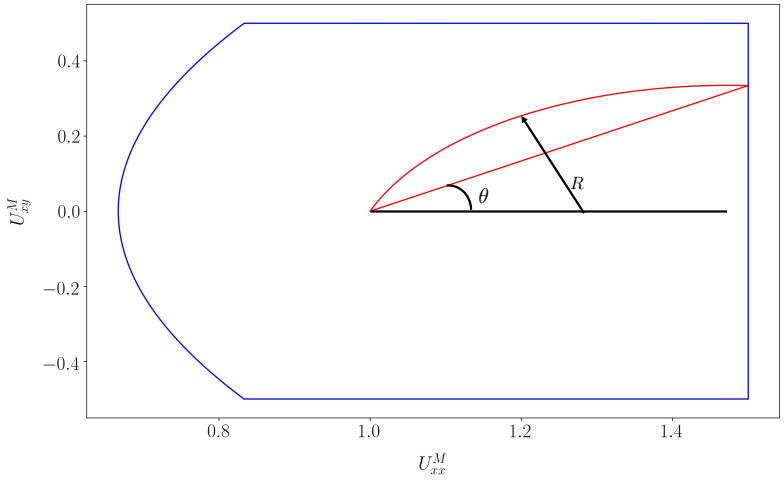


Figure 4.12: Parametrization used for the cyclic load paths.

functions. The results predicted by the k-NN search-aided MOR are virtually indistinguishable from those of the DNS and clearly outperform the conventional POD-based MOR. (In fact, they match those of the DNS so accurately that we do not consider it useful to quantify the difference.)

The offline phase of the k-NN search-aided MOR is faster than that of the conventional POD-based MOR, because the singular value decomposition necessary to identify the basis functions of the POD-based MOR is avoided. The online phase of the k-NN search-aided MOR is slightly slower, because three additional calculations are needed. First, the k-NN search must be applied at each increment. However, because this constitutes a search over 3000 distances each increment, it only requires 0.035 second (averaged over the three verification simulations). Second, the basis functions must be orthonormalized using the Gramm-Schmidt process. However, because only 10 basis functions are considered, this only requires 0.0005 second (averaged over the three verification simulations). Third, a new initial guess must be established if the basis functions are changed. However, because this only requires the matching of the displacement field (using Eq. (4.21)) and avoids recomputing force equilibrium, it only requires 0.0025 second (averaged over the three verification simulations).

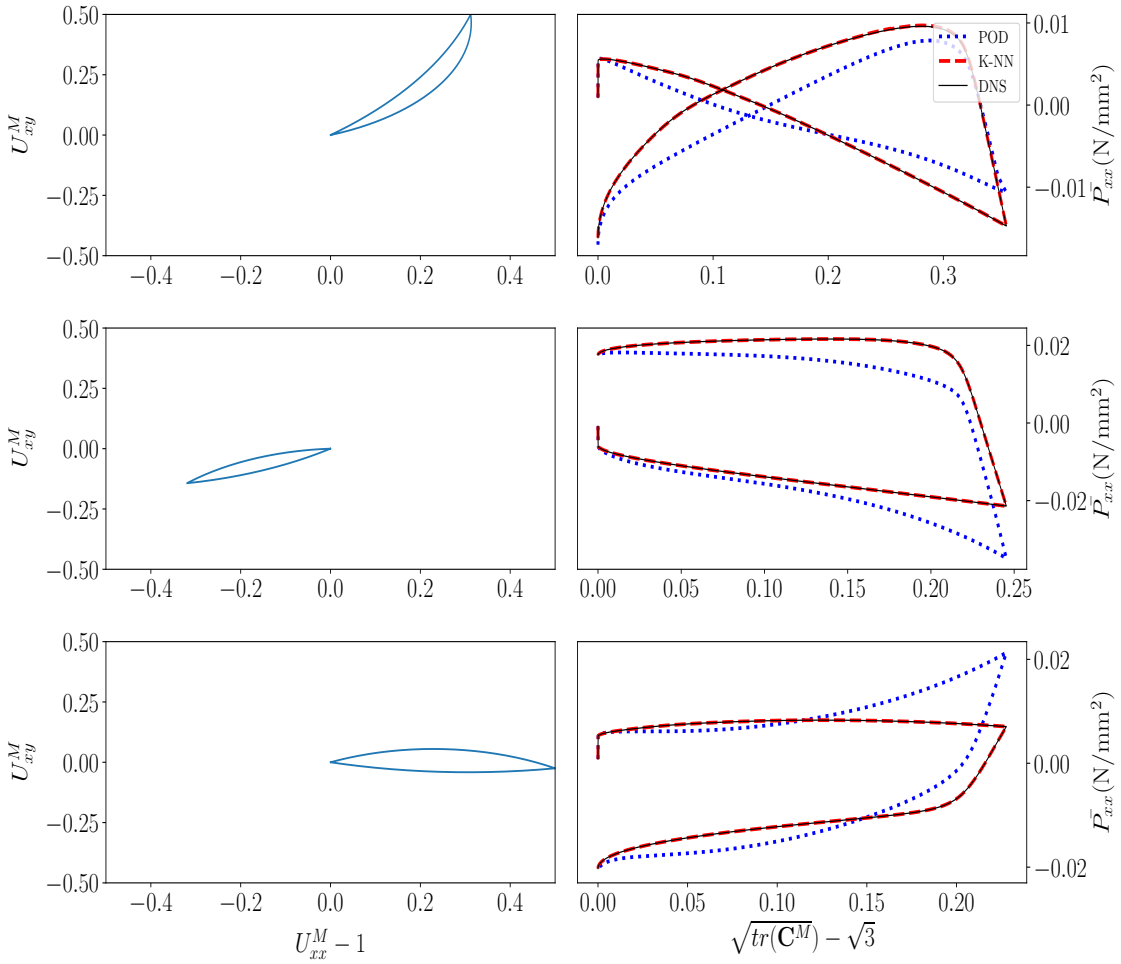


Figure 4.13: Cyclic loading: The results for the DNS, conventional POD-based MOR, and K-NN for three verification simulations using 10 basis functions. Left column: The load path of each verification simulation. Right column: One of the components of the homogenized 1$^{\text{st}}$ Piola-Kirchhoff stress tensor as predicted by K-NN framework.

4.7 Conclusion

Projection-based model-order-reduction (MOR) for elastoplastic models require a large number of basis functions to obtain an acceptable accuracy at the online prediction stage. Since, each basis functions has its own degrees of freedom that needs to be computed, a reduced number of basis functions are necessary to speedup the online computations of projection-based MOR. Therefore, the goal of this chapter was to devise a MOR that substantially reduce the number of basis functions required at the online stage.

To this purpose, unsupervised machine learning algorithms, k-means and DBSCAN, were investigated to group the training solutions according to its load parameters. During the course of online simulation, the group that is closest to the current load path is chosen, and the basis functions associated to the chosen group are employed. Therefore, basis functions from different groups are used during the course of an online simulation. The challenge in incorporating unsupervised learning to MOR for elastoplastic models is the occurrence of inaccuracies every time the basis changes. Three techniques were investigated to reduce these inaccuracies by smoothing the switch between clusters.

The three smoothing techniques investigated were: (i) sharing the the training solutions between clusters, (ii) weighing the training solutions, where weight is set to one for each training solution of the considered cluster and decreases linearly with the distance away from the cluster, and (iii) adaptively mixing the basis functions between clusters.

The results of unsupervised learning approaches indicate that, the switching of basis functions affects the quality of online predictions. Also, the techniques considered to facilitate the transition of basis functions between clusters is not robust and does not systematically improves the results of online predictions compared to those of the conventional MOR based on proper-orthogonal-decomposition.

Therefore, a new MOR approach aided by k nearest neighbour searching was proposed. In contrast to the inaccuracy of the clustering-based MOR at the moment that the basis is switched, the k-NN approach provides a continuous change only to part of the basis. In this approach, for each load increment, k load paths of the training simulations that are nearest to the current load path in the online simulation are identified, and the most suitable solution of each of the k training simulations are used together as the basis functions for that particular load increment.

For the test cases involving monotonic loading, the k-NN-aided MOR clearly outperformed all clustering-assisted MOR approaches and the conventional POD-based MOR. The results of the k-NN-aided MOR were not perfect for the monotonic loading cases, because only 40 training simulations were considered. For the cyclic loading cases however, for which we included 3000 training simulations, the results of the k-NN-aided MOR were indistinguishable from those of the direct numerical simulations. With such a great accuracy, the slight deceleration of the online simulations compared to conventional MOR is a small price to pay.

We believe that an interesting extension can be obtained by combining the k-NN-aided MOR with a recurrent neural network to emulate the basis coefficients. However, compared to the recurrent neural network of [70], this new recurrent neural network should not only be able to treat the path-dependency of elastoplasticity, it should also be able to account for the fact that the basis functions change continuously - something that our initial investigations have shown to be non-trivial.

Another issue required to make the approach based on k-NN searching more general than presented here is associated with our test cases. The reason is that we only considered one cycle loading, where loading always occurred until the border of the domain, which would never occur in a nested multiscale simulation based on computational homogenization. To solve this issue, more training simulations are required and a more involved k-NN search is needed that properly integrates over the load paths. Both issues are highly likely to decelerate the k-NN search and in turn, it may be necessary to use a second neural network to emulate the k-NN search.

Chapter 5

CONCLUSIONS AND OUTLOOK

In this thesis, *a posteriori* projection-based model-order-reduction methods were developed specifically for elastoplastic finite element models. *A posteriori* projection-based MOR uses precomputed solutions (either orthonormalized directly as in the method of reduced basis, or in a decomposed form as in the method of proper orthogonal decomposition) as global basis functions to speed up the simulations of future computations.

In the case of hyperelastic finite elements, a few global basis functions are sufficient to obtain excellent results for future computations. On the other hand, hyperelastoplastic simulations requires a large number of basis functions to acquire a reasonable accuracy.

In this regard, the thesis presents an innovative approach in chapter 2 that reduces the number of global basis functions by incorporating an additional local interpolation using a coarse finite element discretization. Two approaches were discussed: (i) The conventional global basis functions interpolate the majority of fluctuations in the geometry and the local basis functions accounts for deficiencies of global basis functions, and (ii) The local basis functions interpolate majority of fluctuations and the global basis functions account for inaccuracies of the local basis functions. Two local interpolation schemes were investigated.

The results of chapter 2 demonstrate that the local/global interpolation approach of scheme 1 improves the accuracy of online computations compared to the conventional POD-based MOR. However, the results of the local/global approaches were not consistent, and require more development to make it usable in a robust manner.

The robustness may be improved by investigating further on the identification of global basis functions. Currently, either the global basis functions or the local basis functions interpolates majority of displacement field. One may identify basis functions in the center of the spectrum. How to achieve this is currently unclear to the candidate.

Another avenue for improvement may be found in staggered solution schemes. In the present chapter, a monolithic approach was applied (see e.g. [63]), since we

simultaneously solve for the coefficients of the basis functions and for the DoFs of the local interpolation. Staggered approaches, in which one solves for one set of variables in one iteration and for the other set of variables in the next iteration, are completely accepted in fluid-structure simulations (see e.g. [18]) and have shown to provide stability to phase-field damage simulations [74]. They may therefore also help the local/global schemes of chapter 2.

Another approach that may improve the robustness of the presented schemes is increasing the hardening modulus in the stiffness matrix (h in Eq. (2.2)). Such an 'engineering trick' may be considered somewhat ad-hoc, because there is no theory to choose the most appropriate value for this increase. On the other hand, for perfect plasticity in infinitesimal strain settings, this modification is completely accepted to achieve convergence.

The results of chapter 2 lead to the development of a consistent neural network accelerated POD-based MOR approach in chapter 3. The advantage of incorporating neural network with POD-based MOR facilitates to preserve all the microstructural information in the quadrature points. The developed method avoids the stiffness matrix construction and the Newton-Raphson iterative procedure involved in the conventional POD-based MOR. Therefore, many global basis functions can be used for the online computations, which is particularly useful for elastoplastic finite element models.

A recurrent neural network was developed to predict the basis coefficients for an RVE described by finite plasticity, subjected to cyclic and random loading. The results have shown that the accuracy of predictions of RNN accelerated POD-based MOR is similar to the conventional POD-based MOR. The RNN acceleration yields speed ups of factors between 13 and 100 times relative to direct numerical simulations for cyclic and random loading conditions, respectively.

The RNN in chapter 3 emulates the coefficients of 100 global basis functions. Since the use of 100 global basis functions yields acceptable accuracies, but not superb accuracies. To achieve this, one would require even more basis functions, which would compromize the speed of the MOR simulations. To reduce the number of basis functions, machine learning is investigated in chapter 4 in order to adaptive select a limited number of basis functions.

The use of k-means clustering (as investigated in the literature for other types of simulations than elastoplastic finite element simulations) and DBSCAN (not yet

investigated in the literature to the best of the candidate's knowledge) to group training simulations and construct a basis for each group shows high inaccuracies while switching between the basis functions at the online computation stage. Therefore, different smoothing approaches were investigated to reduce such inaccuracies. However, the smoothing approaches did not provide a consistent improvement of the accuracy in the online stage.

Therefore, a method that continuously changes the basis functions was incorporated. This continuous change was achieved using the k-NN search algorithm. The algorithm determines the k most suitable for each load increment, which are then orthonormalized to be used as basis functions for the current increment. The method was investigated for an elastoplastic RVE subjected to large monotonic loading and asymmetrical cyclic loading conditions. The results in chapter 4 have indicated that the adaptive basis section using k-NN provides an accurate result even with the use of substantially reduced number of basis functions compared to the traditional POD-based MOR.

An interesting extension may be obtained by combining the k-NN-aided MOR with a recurrent neural network to emulate the basis coefficients. However, compared to the recurrent neural network of chapter 3, this new recurrent neural network should not only be able to treat the path-dependency of elastoplasticity, it should also be able to account for the fact that the basis functions change continuously - something that our initial investigations have shown to be non-trivial.

Another issue required to make the approach based on k-NN searching more general than presented here is associated with the considered test cases. The reason is that only one cycle loading was considered, and loading always occurred until the border of the domain, which would never occur in a nested multiscale simulation based on computational homogenization. To solve this issue, more training simulations are required and a more involved k-NN search is needed that properly integrates over the load paths. Both issues are highly likely to decelerate the k-NN search and in turn, it may be necessary to use a second neural network to emulate the k-NN search. Although the proposed k-NN-aided MOR is only investigated for relatively simple test cases with clear limitations in terms of their application, its superb accuracy can make it a promising approach for future development that should focus on generalizing the computational strategy.

Appendix A

ACTIVATION FUNCTIONS

An activation function (f in Fig. 3.1) is one of the hyper-parameters, which plays an important role in restricting the output of any neuron to a certain limit, such that the input for the next neuron is not magnified to a large value. Another feature of an activation function is the ability to provide the network with non-linearity. This is a significant feature because, as discussed below, the use of linear activation functions makes the response of the entire network linear, thereby limiting the applicability of neural networks.

In this appendix, commonly used activation functions are discussed. To ease the notation in this appendix, the weighted response of the input of a neuron is abbreviated by x, whose expressions reads:

$$x = b + \sum_{i=1}^{k} w_i O_i, \tag{A.1}$$

where O_i denotes the i^{th} input of the neuron of interest, w_i denotes its associated weight and b denotes the bias of the neuron.

Binary step function is a threshold based activation function that passes the output of a neuron if the value is higher than certain threshold, or restricts them if the output value is less than the threshold. It is thus nothing more than a Heaviside step function:

$$f_{BIN}(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}. \tag{A.2}$$

Though this function is simple to implement, it can only be used for a binary classification problem. The reason for this is that the gradient of the binary step function is zero, making it impossible to update the parameters during the learning stage of the network.

Linear activation function takes the form where the output is proportional to the input. The function (with parameter a) is given by:

$$f_{LIN}(x) = ax, \tag{A.3}$$

and its derivative is:

$$f'_{LIN}(x) = a. \tag{A.4}$$

Because the gradient of the linear activation function is constant, it effectively disappears in the weights and biases of the neuron (w_i and b in Fig. 3.1). For this reason, the constant of the linear activation function cannot be identified during the learning phase, so it is in practise simply ignored ($a = 1$).

The problem of the linear activation function is that the combined result of a layer of neurons is a linear function and hence, a neural network using linear activation functions cannot emulate non-linear relations between the input and the output. This motivates the need for non-linear activation functions for most practical applications.

Sigmoid activation function is one of the most widely used non-linear activation functions in regression models [47]. This activation function takes the form:

$$f_{SIG}(x) = \frac{1}{1 + e^{-x}}. \tag{A.5}$$

The output of the sigmoid function is in the range between 0 and 1, thereby ensuring a normalized output of a neuron. The function provides a smooth s-shaped curve, that is C^∞-continuous, and its derivative reads:

$$f'_{SIG}(x) = f_{SIG}(x)(1 - f_{SIG}(x)). \tag{A.6}$$

Practically, the gradient of the sigmoid function is flat for values of x greater than 2 and less than -2. This means that the update values are often substantially small, resulting in a problem that the gradient vanishes thereby compromising the identification of the weights and biases in the learning phase. Also, the output of the sigmoid function for all neurons are of the same sign, since the function is not symmetric around zero. This issue is rectified using the hyperbolic tangent function.

Hyperbolic tangent function (tanh) is similar to sigmoid function, except that output ranges between -1 to 1. Tanh activation function reads:

$$f_{TAN}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{A.7}$$

and its derivative is given as:

$$f'_{TAN}(x) = 1 - f_{TAN}(x)^2. \tag{A.8}$$

The above mentioned non-linear functions are computationally expensive. This has motivated the construction of different activation functions.

Rectified linear unit (ReLU) is a non-linear activation function that is computationally efficient [55]. ReLU activates a neuron only if its input values are positive. The function is nothing more than the binary step activation function and the linear activation function multiplied with each other:

$$f_{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}. \tag{A.9}$$

with the following derivatives:

$$f'_{ReLU}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}. \tag{A.10}$$

The problem with ReLU is that, when the input is lower than zero, the gradients of ReLU is zero. Therefore, the identification of the weights and biases during the learning phase is compromised. This phenomenon is known as the dying ReLU problem.

Leaky ReLU activation function is an improved version of ReLU, which overcomes the dying ReLU problem [50]. Instead of defining 0 for negative input values, Leaky ReLU multiplies the input with a small scalar. The function reads:

$$f_{LR}(x) = \begin{cases} x, & x \geq 0 \\ \epsilon x, & x < 0 \end{cases}. \tag{A.11}$$

Its derivative reads:

$$f'_{LR}(x) = \begin{cases} 1, & x \geq 0 \\ \epsilon, & x < 0 \end{cases}. \tag{A.12}$$

Appendix B

RNN UNITS

In this appendix, the two main types of RNN gates are briefly introduced: the long-short-term-memory unit (LSTM) and the gated-recurrent-unit (GRU).

B.1 Long Short Term Memory

LSTM has a cell state that allows to keep or forget the sequential information. The processing capability of the cell state to only consider relevant information, facilitates to carry data from past time steps to future time steps. The control of information in the cell state is accomplished using gates. Gates effectively decide which information has to be kept and discarded during training. The working of LSTM is presented in Fig. B.1 can be summarized in the following steps:

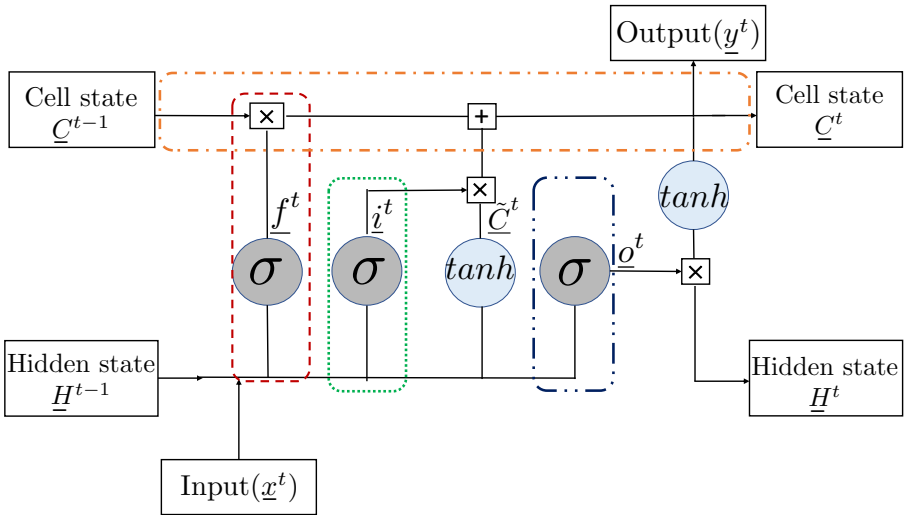


Figure B.1: A detailed LSTM architecture. Red dashed box represents the forget gate. Green dotted box is the input gate, blue dash dotted is the output gate and the orange dash dot box shows the cell state. $\boxed{+}$ and $\boxed{\times}$ is an element-wise summation and element-wise multiplication operator respectively.

1. At first, there is a forget gate. This forget gate decides what information has to be kept in cell state '$\underline{C}^{t-1}$'. The decision is made by passing the hidden variables of previous time step '$\underline{H}^{t-1}$' and the input of current time step '$\underline{x}^t$' through a sigmoid activation function (σ). The output of forget gate '$\underline{f}^t$' is

between 0 and 1, with a vector size being the number of variables in the cell state '$\underline{C}^{t-1}$'. The information is retained for values closer to 1 and discarded for values closer to 0. The output of a forget state can be expressed as follows:

$$\underline{f}^t = \sigma(\underline{\underline{W}}_{Hf}\underline{H}^{t-1} + \underline{\underline{W}}_{xf}\underline{x}^t + \underline{b}_f), \tag{B.1}$$

where, $\underline{\underline{W}}_{Hf}$ and $\underline{\underline{W}}_{xf}$ are the weights of the previous hidden state '$\underline{H}^{t-1}$' and current input '$\underline{x}^t$' respectively. '$\underline{b}_f$' is the bias term .'f' in the subscript refers that the weights and bias corresponds to forget gate.

2. Next is an input gate. This gate decides what information to be additionally stored in the cell state, by creating a new set of values '$\underline{\tilde{C}}^t$'. This operation is performed in two steps. First, the previous hidden variables '$\underline{H}^{t-1}$' and the current input '$\underline{x}^t$' are passed through a sigmoid activation (σ), that decides which values will be updated. This yields a new input vector '$\underline{i}^t$' whose values are between 0 and 1. Again, the same data, '$\underline{H}^{t-1}$' and '$\underline{x}^t$', is passed through 'tanh' to create '$\underline{\tilde{C}}^t$'.

$$\underline{i}^t = \sigma(\underline{\underline{W}}_{Hi_1}\underline{H}^{t-1} + \underline{\underline{W}}_{xi_1}\underline{x}^t + \underline{b}_{i_1}), \tag{B.2}$$

$$\underline{\tilde{C}}^t = \tanh(\underline{\underline{W}}_{Hi_2}\underline{H}^{t-1} + \underline{\underline{W}}_{xi_2}\underline{x}^t + \underline{b}_{i_2}), \tag{B.3}$$

3. In the third step, new cell state '$\underline{C}^t$' is updated based on $\underline{f}^t$, $\underline{i}^t$ and $\underline{\tilde{C}}^t$. To discard some values of '$\underline{C}^{t-1}$', an element-wise multiplication is performed between $\underline{f}^t$ and $\underline{C}^{t-1}$. In order to update the cell state, $\underline{i}^t$ and $\underline{\tilde{C}}^t$ are multiplied element-wise (denoted by $\odot$). The update of the cell state for the current time step is given as:

$$\underline{C}^t = \underline{f}^t \odot \underline{C}^{t-1} + \underline{i}^t \odot \underline{\tilde{C}}^t, \tag{B.4}$$

4. Finally, an output gate decides the next hidden state variables '$\underline{H}^t$' that will be passed on to the next sequence. In order to obtain that, $\underline{H}^{t-1}$ and $\underline{x}^t$ are passed through a sigmoid function and then the newly obtained cell state $\underline{C}^t$ is passed through the tanh activation function. The output hidden variables are computed as:

$$\underline{o}^t = \sigma(\underline{\underline{W}}_{Ho}\underline{H}^{t-1} + \underline{\underline{W}}_{xo}\underline{x}^t + \underline{b}_o), \tag{B.5}$$

and finally,

$$\underline{H}^t = \underline{o}^t \odot \tanh(\underline{C}^t). \tag{B.6}$$

The obtained hidden state variables and cell states are passed on to the next time step.

B.2 Gated Recurrent Unit

A GRU is a simplified form of LSTM, that enables control over the flow of information only through the hidden variables 'H'. There is no cell state in GRU. It also uses only two gates: an update gate ($\underline{z}^t$) and a reset gate ($\underline{r}^t$) to determine the amount of information to be passed on and to be retained by the hidden variable. GRU perform less computation, which makes them faster than LSTM. The working of a GRU is presented in B.2, and can be summarised as follows:

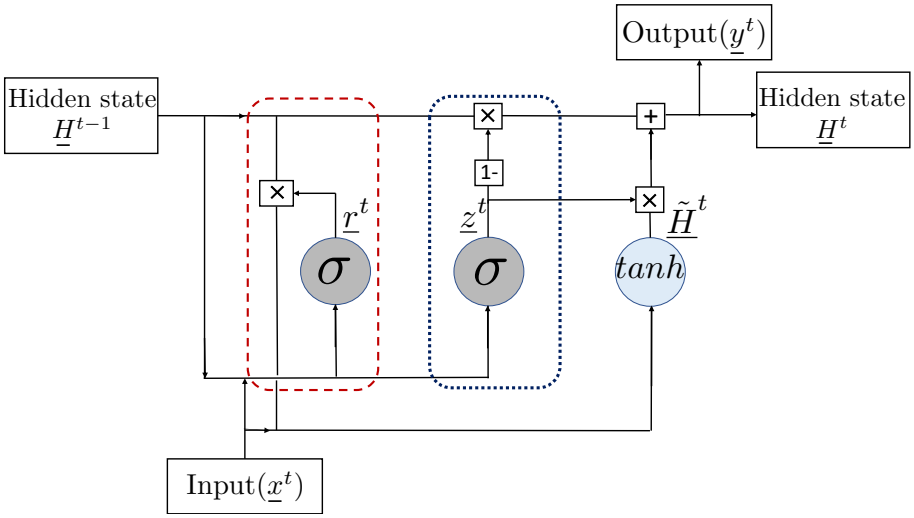


Figure B.2: A detailed GRU architecture. Red dashed box is the reset gate, blue dotted box represents the update gate. [1−] is an element-wise subtraction operator.

1. First, there is an update gate that takes the role of both the forget and input gates of LSTM. An update gate has a sigmoid function to which the hidden variables of previous time step '$\underline{H}^{t-1}$' and the current input '$\underline{x}^t$' is passed. The output update vector '$\underline{z}^t$' is given as:

$$\underline{z}^t = \sigma(\underline{\underline{W}}_{Hu}\underline{H}^{t-1} + \underline{\underline{W}}_{xu}\underline{x}^t + \underline{b}_u), \tag{B.7}$$

2. Another gate is the reset gate which decides the values that are to be replaced in the previous hidden layer with a vector of new reset values '$\underline{r}^t$'. The reset gate applies the sigmoid activation function to the hidden variables of previous time step 'H^{t-1}' and to the current input '$\underline{x}^t$'. The computations of reset gate can be expressed as follows:

$$\underline{r}^t = \sigma(\underline{\underline{W}}_{Hr1}\underline{H}^{t-1} + \underline{\underline{W}}_{xr1}\underline{x}^t + \underline{b}_{r1}), \tag{B.8}$$

3. In the third step, a candidate value '$\underline{\tilde{H}}^t$' is obtained by passing the current input $\underline{x}^t$ and previous hidden state '$\underline{H}^{t-1}$' weighted by the values from reset gate $\underline{r}^t$. This computation can be expressed as:

$$\underline{\tilde{H}}^t = \tanh(\underline{\underline{W}}_{Hr2}(\underline{r}^t \odot \underline{H}^{t-1}) + \underline{\underline{W}}_{xr2}\underline{x}^t + \underline{b}_{r2}), \tag{B.9}$$

4. Finally, the output of the GRU is obtained by weighing the candidate hidden variable '$\underline{\tilde{H}}$' with the output of the update gate and adding the result to the product of previous hidden state variable '$\underline{H}^{t-1}$' weighted by $1 - \underline{z}^t$, where $\underline{z}^t$ is the output of update gate. This operation can be written as:

$$\underline{H}^t = \underline{z}^t \odot \underline{H}^{t-1} + (1 - \underline{z}^t) \odot \underline{\tilde{H}}^t. \tag{B.10}$$

BIBLIOGRAPHY

[1] Quarteroni A., Manzoni A., and Negri F. *Reduced Basis Methods for Partial Differential Equations*. Vol. 92. 2015. DOI: https://doi.org/10.1007/978-3-319-15431-2.

[2] D. Amsallem, M. J. Zahr, and C. Farhat. "Nonlinear model order reduction based on local reduced-order bases". In: *International Journal for Numerical Methods in Engineering* 92.10 (2012), pp. 891–916. DOI: https://doi.org/10.1002/nme.4371.

[3] J. S. R. Anttonen. "Techniques for reduced order modeling of aeroelastic structures with deforming grids". PhD thesis. Air Force Institute of Technology, Jan. 2001.

[4] P. Astrid et al. "Missing Point Estimation in Models Described by Proper Orthogonal Decomposition". In: *IEEE Transactions on Automatic Control* 53.10 (2008), pp. 2237–2251. DOI: 10.1109/TAC.2008.2006102.

[5] L.A.A. Beex, R.H.J. Peerlings, and M.G.D. Geers. "Central summation in the quasicontinuum method". In: *Journal of the Mechanics and Physics of Solids* 70 (2014), pp. 242–261. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2014.05.019.

[6] L.A.A. Beex et al. "Higher-order quasicontinuum methods for elastic and dissipative lattice models: uniaxial deformation and pure bending". In: *GAMM-Mitteilungen* 38.2 (2015), pp. 344–368. DOI: https://doi.org/10.1002/gamm.201510018.

[7] N. C. Bender, H. C. Pedersen, and T. O. Andersen. "Feasibility of Deep Neural Network Surrogate Models in Fluid Dynamics". English. In: *Modeling, Identification and Control (Online)* 40.2 (Apr. 2019), pp. 71–87. ISSN: 0332-7353. DOI: 10.4173/mic.2019.2.1.

[8] P. Benner, S. Gugercin, and K. Willcox. "A survey of projection-based model reduction methods for parametric dynamical systems". In: *SIAM Review* 57.4 (2015), pp. 483–531. ISSN: 00361445.

[9] S. Bhattacharjee and K. Matouš. "A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials". In: *Journal of Computational Physics* 313 (2016), pp. 635–653. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.01.040.

[10] B. Bohn et al. "Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods". In: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, pp. 621–630. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.05.226.

[11] K. Carlberg et al. “Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations”. In: *International Journal for Numerical Methods in Engineering* 86.2 (2011), pp. 155–181. DOI: https://doi.org/10.1002/nme.3050.

[12] S. Chaturantabut and D. C. Sorensen. “Nonlinear Model Reduction via Discrete Empirical Interpolation”. In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764. DOI: https://doi.org/10.1137/090766498.

[13] S. Chaturantabut and D.C. Sorensen. “Discrete Empirical Interpolation for nonlinear model reduction”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. 2009, pp. 4316–4321. DOI: 10.1109/CDC.2009.5400045.

[14] L. Chen et al. “Generalized quasicontinuum modeling of metallic lattices with geometrical and material nonlinearity and variability”. In: *Computer Methods in Applied Mechanics and Engineering* 366 (2020), p. 112878. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.112878.

[15] F. Chinesta, P. Ladeveze, and E. Cueto. “A Short Review on Model Order Reduction Based on Proper Generalized Decomposition”. In: *Archives of Computational Methods in Engineering* 18.4 (2011), p. 395. URL: https://doi.org/10.1007/s11831-011-9064-7.

[16] K. Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[17] I. Chung, S. Im, and M. Cho. “A neural network constitutive model for hyperelasticity based on molecular dynamics simulations”. In: *International Journal for Numerical Methods in Engineering* 122.1 (2021), pp. 5–24. DOI: https://doi.org/10.1002/nme.6459.

[18] Joris Degroote, Klaus-Jürgen Bathe, and Jan Vierendeels. “Performance of a new partitioned procedure versus a monolithic procedure in fluidâstructure interaction”. In: *Computers & Structures* 87.11 (2009). Fifth MIT Conference on Computational Fluid and Solid Mechanics, pp. 793–801. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2008.11.013.

[19] M. Dihlmann, M. Drohmann, and B. Haasdonk. “Model reduction of parametrized evolution problems using the reduced basis method with adaptive time partitioning”. In: Jan. 2011.

[20] M. Doškář et al. “Microstructure-informed reduced modes synthesized with wang tiles and the generalized finite element method”. In: *arXiv* (2020). ISSN: 23318422. arXiv: 2010.02690.

[21] M. Drohmann, B. Haasdonk, and M. Ohlberger. “Adaptive Reduced Basis Methods for Nonlinear Convection–Diffusion Equations”. In: *Finite Volumes for Complex Applications VI Problems & Perspectives*. Ed. by Jaroslav Fořt

et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 369–377. ISBN: 978-3-642-20671-9.

[22] C. Eckart and G. Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/BF02288367.

[23] M. Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of 2nd International Conference on Knowledge Discovery and*. 1996, pp. 226–231.

[24] J. Fauque, I. Ramière, and D. Ryckelynck. "Hybrid hyper-reduced modeling for contact mechanics problems". In: *International Journal for Numerical Methods in Engineering* 115.1 (2018), pp. 117–139. DOI: https://doi.org/10.1002/nme.5798.

[25] F. Fritzen, M. Fernández, and F. Larsson. "On-the-Fly Adaptivity for Nonlinear Twoscale Simulations Using Artificial Neural Networks and Reduced Order Modeling". In: *Frontiers in Materials* 6 (2019), p. 75. ISSN: 2296-8016. DOI: 10.3389/fmats.2019.00075.

[26] T. Furukawa and M. Hoffman. "Accurate cyclic plastic analysis using a neural network material model". In: *Engineering Analysis with Boundary Elements* 28 (Mar. 2004), pp. 195–204. DOI: 10.1016/S0955-7997(03)00050-X.

[27] M.G.D. Geers, V.G. Kouznetsova, and W.A.M. Brekelmans. "Multi-scale computational homogenization: Trends and challenges". In: *Journal of Computational and Applied Mathematics* 234.7 (2010). Fourth International Conference on Advanced COmputational Methods in ENgineering (ACOMEN 2008), pp. 2175–2182. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2009.08.077.

[28] F. Ghavamian and A. Simone. "Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network". In: *Computer Methods in Applied Mechanics and Engineering* 357 (2019), p. 112594. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2019.112594.

[29] F. Ghavamian, P. Tiso, and A. Simone. "POD-DEIM model order reduction for strain-softening viscoplasticity". In: *Computer Methods in Applied Mechanics and Engineering* 317 (2017), pp. 458–479. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2016.11.025.

[30] C. Gogu et al. "Dimensionality Reduction Approach for Response Surface Approximations: Application to Thermal Design". In: *American Institute of Aeronautics and Astronautics Journal* 47.7 (2009), pp. 1700–1708. DOI: https://doi.org/10.2514/1.41414.

[31] C. Gogu and J.C. Passieux. “Efficient surrogate construction by combining response surface methodology and reduced order modeling”. In: *Structural and Multidisciplinary Optimization* 47.6 (2013), pp. 821–837. DOI: 10.1007/s00158-012-0859-4.

[32] M. B. Gorji et al. “On the potential of recurrent neural networks for modeling path dependent plasticity”. In: *Journal of the Mechanics and Physics of Solids* 143 (2020), p. 103972. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2020.103972.

[33] H.J.Logarzo, G.Capuano, and J.J. Rimoli. “Smart constitutive laws: Inelastic homogenization through machine learning”. In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113482. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113482.

[34] B. Haasdonk, M. Dihlmann, and M. Ohlberger. “A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space”. In: *Mathematical and Computer Modelling of Dynamical Systems* 17.4 (2011), pp. 423–442. DOI: 10.1080/13873954.2011.547674.

[35] J. S. Hale et al. “A hyper-reduction method using adaptivity to cut the assembly costs of reduced order models”. In: *Computer Methods in Applied Mechanics and Engineering* 380 (2021), p. 113723. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.113723.

[36] Y. M. A. Hashash, S. Jung, and J. Ghaboussi. “Numerical implementation of a neural network based material model in finite element analysis”. In: *International Journal for Numerical Methods in Engineering* 59.7 (2004), pp. 989–1005. DOI: https://doi.org/10.1002/nme.905.

[37] T. Hastie, J. Friedman, and R. Tibshirani. “Neural Networks”. In: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer New York, 2001, pp. 347–369. ISBN: 978-0-387-21606-5. DOI: 10.1007/978-0-387-21606-5_11.

[38] J.A. Hernández et al. “Dimensional hyper-reduction of nonlinear finite element models via empirical cubature”. In: *Computer Methods in Applied Mechanics and Engineering* 313 (2017), pp. 687–722. ISSN: 0045-7825.

[39] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[40] A. Hürkamp and M. Kaliske. “A model reduction approach for hyperelastic materials based on Proper Orthogonal Decomposition”. In: *Proceedings in Applied Mathematics and Mechanics* 15.1 (2015), pp. 203–204. DOI: https://doi.org/10.1002/pamm.201510092.

[41] P. Kerfriden, O. Allix, and P. Gosselet. “A three-scale domain decomposition method for the 3D analysis of debonding in laminates”. In: *Computational Mechanics* 44.3 (2009), pp. 343–362. DOI: 10.1007/s00466-009-0378-3.

[42] P. Kerfriden et al. “A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics”. In: *Computer Methods in Applied Mechanics and Engineering* 256 (2013), pp. 169–188. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2012.12.004.

[43] P. Kerfriden et al. “Bridging proper orthogonal decomposition methods and augmented Newton-Krylov algorithms: An adaptive model order reduction for highly nonlinear mechanical problems”. In: *Computer Methods in Applied Mechanics and Engineering* 200.5 (2011), pp. 850–866. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2010.10.009.

[44] J.F. Kolen and S.C. Kremer. “Gradient flow in recurrent nets: The difficulty of learning longterm dependencies. A Field Guide to Dynamical Recurrent Network”. In: *IEEE.* (2010).

[45] E.D. Koronaki et al. “Classification of states and model order reduction of large scale Chemical Vapor Deposition processes with solution multiplicity”. In: *Computers Chemical Engineering* 121 (2019), pp. 148–157. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2018.08.023.

[46] V. Kouznetsova, W. A. M. Brekelmans, and F. P. T. Baaijens. “An approach to micro-macro modeling of heterogeneous materials”. In: *Computational Mechanics* 27.1 (2001), pp. 37–48. DOI: 10.1007/s004660000212.

[47] B. A. Le, J. Yvonnet, and Q.C. He. “Computational homogenization of nonlinear elastic materials using neural networks”. In: *International Journal for Numerical Methods in Engineering* 104.12 (2015), pp. 1061–1084. DOI: https://doi.org/10.1002/nme.4953.

[48] P.A. LeGresley and J.J. Alonso. “Investigation of non-linear projection for POD based reduced order models for aerodynamics”. In: *39th Aerospace Sciences Meeting and Exhibit* c (2001).

[49] Y. Liu, N. Sukuntee, and S. Chaturantabut. “Parametric Nonlinear Model Reduction Using K-Means Clustering for Miscible Flow Simulation”. In: *Journal of Applied Mathematics* 2020 (2020), p. 3904606. DOI: 10.1155/2020/3904606. URL: https://doi.org/10.1155/2020/3904606.

[50] Yisi Liu et al. “A modified leaky ReLU scheme (MLRS) for topology optimization with multiple materials”. In: *Applied Mathematics and Computation* 352 (2019), pp. 188–204. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2019.01.038.

[51] Z. Liu, M.A. Bessa, and W.K.Liu. “Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials”. In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 319–341. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2016.04.004.

[52] M. Meyer and H. G. Matthies. “Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods”. In: *Computational Mechanics* 31.1-2 SPEC. (2003), pp. 179–191. ISSN: 01787675. DOI: https://doi.org/10.1007/s00466-002-0404-1.

[53] B. Miled, D. Ryckelynck, and S. Cantournet. “A priori hyper-reduction method for coupled viscoelasticâviscoplastic composites”. In: *Computers & Structures* 119 (2013), pp. 95–103. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2012.11.017.

[54] M. Mozaffar et al. “Deep learning predicts path-dependent plasticity”. In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 26414–26420. ISSN: 0027-8424. DOI: 10.1073/pnas.1911815116.

[55] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *ICML*. 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf.

[56] S. Niroomandi et al. “Model order reduction for hyperelastic materials”. In: *International Journal for Numerical Methods in Engineering* 81.9 (2010), pp. 1180–1206. DOI: https://doi.org/10.1002/nme.2733.

[57] A. Nouy. “A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations”. In: *Computer Methods in Applied Mechanics and Engineering* 199.23 (2010), pp. 1603–1626. ISSN: 0045-7825.

[58] B. Peherstorfer et al. “Localized Discrete Empirical Interpolation Method”. In: *SIAM Journal on Scientific Computing* 36.1 (2014), A168–A192. DOI: 10.1137/130924408.

[59] C. Prud’homme et al. “Reliable Real-Time Solution of Parametrized Partial Differential Equations: Reduced-Basis Output Bound Methods ”. In: *Journal of Fluids Engineering* 124.1 (Nov. 2001), pp. 70–80. ISSN: 0098-2202. DOI: 10.1115/1.1448332.

[60] A. Quarteroni, G. Rozza, and A. Manzoni. “Certified reduced basis approximation for parametrized partial differential equations and applications”. In: *Journal of Mathematics in Industry* 1.1 (2011), p. 3. DOI: 10.1186/2190-5983-1-3.

[61] A. Radermacher and S. Reese. “Model reduction in elastoplasticity: Proper orthogonal decomposition combined with adaptive sub-structuring”. In: *Comput. Mech* 54.3 (2014), pp. 677–687. ISSN: 01787675.

[62] A. Radermacher and S. Reese. “POD-based model reduction with empirical interpolation applied to nonlinear elasticity”. In: *International Journal for Numerical Methods in Engineering* 107.6 (2016), pp. 477–495.

[63] Srivathsan Ravi and Andreas Zilian. “Time and frequency domain analysis of piezoelectric energy harvesters by monolithic finite element modeling”. In: *International Journal for Numerical Methods in Engineering* 112.12 (2017), pp. 1828–1847. DOI: https://doi.org/10.1002/nme.5584.

[64] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[65] I.B.C.M. Rocha, P. Kerfriden, and F.P. van der Meer. “Micromechanics-based surrogate models for the response of composites: A critical comparison between a classical mesoscale constitutive model, hyper-reduction and neural networks”. In: *European Journal of Mechanics - A/Solids* 82 (2020), p. 103995. ISSN: 0997-7538. DOI: https://doi.org/10.1016/j.euromechsol.2020.103995.

[66] D. Ryckelynck. “A priori hyperreduction method: an adaptive approach”. In: *Journal of Computational Physics* 202.1 (2005), pp. 346–366. ISSN: 0021-9991.

[67] D. Ryckelynck. “Hyper-reduction of mechanical models involving internal variables”. In: *International Journal for Numerical Methods in Engineering* 77.1 (2009), pp. 75–89.

[68] *Multidimensional Hyper-Reduction of Large Mechanical Models Involving Internal Variables*. Vol. Volume 1: Advanced Computational Mechanics; Advanced Simulation-Based Engineering Sciences; Virtual and Augmented Reality; Applied Solid Mechanics and Material Processing; Dynamical Systems and Control. Engineering Systems Design and Analysis. July 2012, pp. 99–106. DOI: 10.1115/ESDA2012-82971.

[69] S.Jung and J.Ghaboussi. “Characterizing rate-dependent material behaviors in self-learning simulation”. In: *Computer Methods in Applied Mechanics and Engineering* 196.1 (2006), pp. 608–619. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2006.06.006.

[70] S.Vijayaraghavan et al. “Neural-network acceleration of projection-based model-order-reduction for finite plasticity: Application to RVEs”. In: *Submitted to Mechanics Research Communications* ().

[71] S. Sahyoun and S. M. Djouadi. “Nonlinear model reduction using Space Vectors Clustering POD with application to the Burgers’ equation”. In: *2014 American Control Conference*. 2014, pp. 1661–1666. DOI: 10.1109/ACC.2014.6859104.

[72] C. Settgast et al. “A hybrid approach to simulate the homogenized irreversible elastic–plastic deformations and damage of foams by neural networks”. In: *International Journal of Plasticity* 126 (2020), p. 102624. ISSN: 0749-6419. DOI: https://doi.org/10.1016/j.ijplas.2019.11.003.

[73] L. Sirovich. “Turbulence and the dynamics of coherent structures. I - Coherent structures. II - Symmetries and transformations. III - Dynamics and scaling”. In: *Quarterly of Applied Mathematics* 45 (Oct. 1987). DOI: 10.1090/qam/910463.

[74] Erlend Storvik et al. “An accelerated staggered scheme for variational phase-field models of brittle fracture”. In: *Computer Methods in Applied Mechanics and Engineering* 381 (2021), p. 113822. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.113822.

[75] R. Swischuk et al. “Projection-based model reduction: Formulations for physics-based machine learning”. In: *Computers and Fluids* 179 (2019), pp. 704–717. ISSN: 00457930. DOI: https://doi.org/10.1016/j.compfluid.2018.07.021.

[76] P. Tiernan, B. Draganescu, and M. Hillery. “Modelling of extrusion force using the surface response method”. In: *International Journal of Advanced Manufacturing Technology* 27.1-2 (2005), pp. 48–52. ISSN: 02683768.

[77] J. F. Unger and C. Könke. “Coupling of scales in a multiscale simulation using neural networks”. In: *Computers & Structures* 86.21 (2008), pp. 1994–2003. ISSN: 0045-7949. DOI: https://doi.org/10.1016/j.compstruc.2008.05.004.

[78] Y. Wardi. “A stochastic steepest-descent algorithm”. In: *Journal of Optimization Theory and Applications* 59.2 (1988), pp. 307–323. DOI: 10.1007/BF00938315. URL: https://doi.org/10.1007/BF00938315.

[79] K. Washabaugh et al. “Nonlinear Model Reduction for CFD Problems Using Local Reduced Order Bases”. In: June 2012. DOI: 10.2514/6.2012-2686.

[80] P. Weber, J. Geiger, and W. Wagner. “Constrained neural network training and its application to hyperelastic material modeling”. In: *Computational Mechanics* 68.5 (2021), pp. 1179–1204. DOI: 10.1007/s00466-021-02064-8. URL: https://doi.org/10.1007/s00466-021-02064-8.

[81] H. Wessels, C. Weißenfels, and P. Wriggers. “The neural particle method - An updated Lagrangian physics informed neural network for computational fluid dynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 368 (2020), p. 113127. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113127.

[82] L. Wu and L. Noels. In: *In Preparation* ().

[83] L. Wu et al. “A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths”. In: *Computer Methods in Applied Mechanics and Engineering* 369 (2020), p. 113234. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113234.