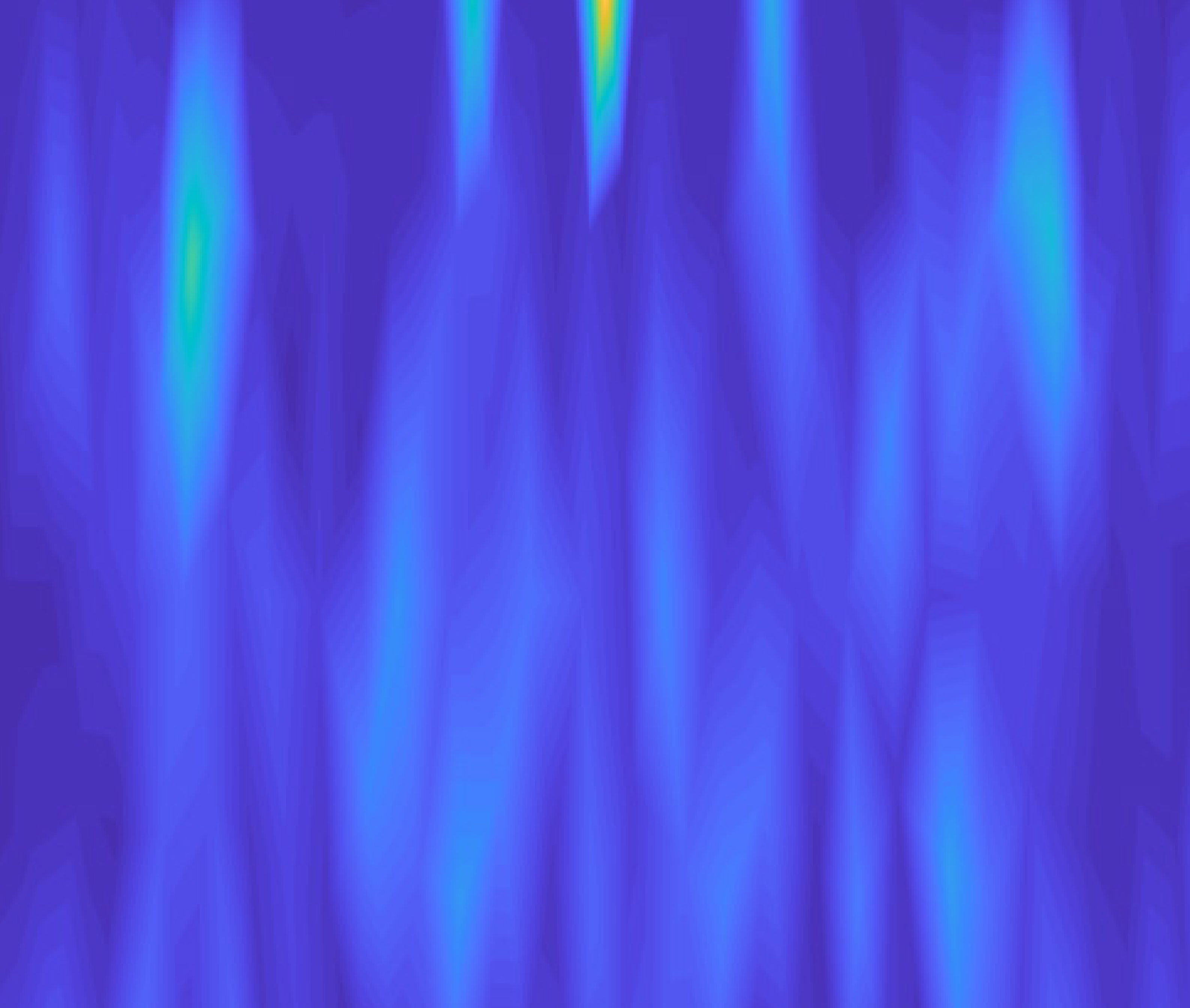


Optimizing the Reduced Basis Construction for Reduced-Order Mechanical Models

Automatic and efficient load case selection using Bayesian machine learning

K. Tjensvoll

TU Delft

Optimizing the Reduced Basis Construction for Reduced-Order Mechanical Models

Automatic and Efficient Load Case Selection using Bayesian Machine Learning

by

Knut Tjensvoll

in partial fulfillment of the requirements for the degree of

Master of Science
in Civil Engineering

at the Delft University of Technology
to be defended publicly on Thursday December 12th, 2019 at 15:00.

Student Number:	4745108	
Project Duration:	Dec. 10, 2018 - Dec. 12, 2019	
Thesis Committee:	Dr. ir. F. P van der Meer,	TU Delft, Chair
	Prof. dr. ir. L. J. Sluijs,	TU Delft, Supervisor
	Dr. ir. I. B. C. M. Rocha,	TU Delft, Daily supervisor
	Dr. ir. M. A. Bessa	TU Delft, Supervisor

TU Delft Delft University of Technology

ABSTRACT

Numerical simulations have become an essential part of design in every field of engineering, and the boundaries of technology are pushed further out every year. In structural engineering, the desire to design structures that have complex shapes or that are simply cheaper and more efficient, has necessitated the use of complex numerical simulations. This necessity is further substantiated when taking into consideration structures such as wind turbines that are subjected to extreme environmental conditions. In many cases, however, such simulations are prohibitively expensive due to complex material behaviour or the many-query nature of design optimization. The lack of knowledge and understanding of the behaviour and failure mechanisms is compensated by adopting less complex designs and/or high safety factors, which leads to less efficient and more expensive designs.

In recent years, methods to circumvent such high computational costs have been developed. Acceleration techniques such as model-order reduction (MOR) are now widely researched, and significant developments are being made to overcome the issues of prohibitively expensive high-fidelity models. This thesis uses Proper Orthogonal Decomposition (POD) to drastically reduce the degrees of freedom of a simply supported beam that is loaded in the downwards direction along the span. The result of the MOR process is a reduced-order model (ROM) that accurately approximates the behaviour of the full-order model (FOM). The ROM is constructed by determining a set of basis vectors that contain compressed information of representative full-order solutions collected in an *offline* training phase. The goal in this thesis is so collect the information and construct the reduced basis as efficiently as possible while guaranteeing a given accuracy of the ROM.

Two methods are presented to iteratively construct the reduced basis. The first one is the *Surrogate Parameter Space* (SPS) method, where greedy sampling is performed on incrementally finer grids of points along the beam. Each individual grid is referred to as an SPS, and they are exhausted by selecting the load location that in each iteration will add the most new information to the ROM. The other method is the *Gaussian Process Regression* (GPR). Greedy sampling using a Bayesian machine learning algorithm involving GPR is used to predict load locations along the beam that add the most new information to the ROM. A method to efficiently combine and compress information obtained in each iteration was also developed.

The results showed that the SPS method is efficient to construct an accurate ROM for the beam model in this thesis, but the GPR managed to recognize that some areas in the span did not have to be sampled as much as others. This makes GPR the more promising method for other high-fidelity models when high accuracy is desired. The results also showed that both methods depend greatly on input parameters that define how much information is kept in each iteration, and for GPR an additional parameter that determines how much accurate the regression itself should be. In order to execute an efficient offline phase, these parameters must be chosen carefully, and it is recommended for future work to develop methods to adaptively choose them. It was also shown that the number of ROMs that have to be run as part of the greedy sampling is the bottleneck of efficiency for both methods. For high-fidelity models with higher-dimensional parameter spaces, this bottleneck necessitates the use of hyper-reduction techniques such as the Empirical Cubature Method (ECM) to reduce the computation time of the ROM itself. It is recommended that the methods investigated and the corresponding results in this thesis are used as a stepping stone to implement automatic and efficient sampling methods to other high-fidelity models.

ACKNOWLEDGEMENTS

I would like to acknowledge and express my gratitude to the people who have helped me over this last year during my thesis:

First, I would like to thank my graduation committee, Frans van der Meer, Bert Sluijs, Iuri Rocha and Miguel Bessa. I am grateful for you introducing me into the field of computational mechanics and giving me opportunity to work on this topic, from which I have gained invaluable lessons that will certainly be useful in my career. I thank you for all your constructive feedback and critical questions during our many meetings. I would especially like to thank Iuri for his extensive and invaluable support throughout the thesis. Without his guidance in the topic of model-order reduction, this would be a near impossible task.

I would like to thank my friends here at TUD that have helped me and supported me through this arduous journey. You have kept my spirits high during times of infinite debugging, and you have always been around to join me for much-needed coffee breaks.

Thank you, Maria, for staying with me all those long evenings in the office and supporting me and giving me continuous encouragement and happiness.

Thanks to everyone who helped me to complete my MSc and thesis at TU Delft.

K. Tjensvoll
Delft, December 2019

CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF SYMBOLS

Latin Symbols

$\boldsymbol{a}$	Step size vector for gradient ascent
$\mathbf{C}$	Covariance matrix with noise
$\mathcal{C}$	Vector of candidate parameters for GPRP
$\mathcal{C}^m$	Vector of candidate parameters to add for GPRP
$\boldsymbol{E}$	Vector of exact error observations for GPRC
$E^p_{N_{it}}$	Exact error before enriching basis
$E^a_{N_{it}}$	Exact error after enriching basis
E^{max}	Exact error tolerance
$\mathcal{E}$	Error pool containing $\mathcal{I}^p$, $\mathcal{I}^a$, E^p and E^a
$\widetilde{E}$	Predictive mean of the exact error from GPRC
$\mathbf{f}^{\Omega}$	Internal force vector
$\mathbf{f}^{\Omega}_r$	Reduced internal force vector
$\mathbf{f}^{\Gamma}$	External force vector
$\mathbf{f}^{\Gamma}_r$	Reduced external force vector
$\boldsymbol{g}_t$	Gradient of objective function with respect to the hyperparameters for gradient ascent
$\boldsymbol{I}$	Identity matrix
$\mathcal{I}$	A posteriori error indicator in parameter space P
$\mathcal{I}^p$	Error indicator in parameter space P before enriching basis
$\mathcal{I}^a$	Error indicator in parameter space P after enriching basis
$\widehat{\mathcal{I}}$	Error indicator in surrogate parameter space $\widehat{P}_{N_p}$
$\mathcal{I}^c$	Candidate error indicators in parameter space P for GPRP
$\widetilde{\mathcal{I}}$	Predictive mean of the error indicator from GPRP
$\widetilde{\mathcal{I}}^{max}$	Maximum of the predictive mean of the error indicator from GPRP
$\widetilde{\mathcal{I}}^{min}$	Minimum of the predictive mean of the error indicator from GPRP
$\boldsymbol{\mathcal{I}}$	Vector of error indicator observations for GPRP
$\mathcal{I}^{\star}$	Test input error indicators to construct GPRC
k^e	Number of elastic modes in the truncated SVD
k^i	Number of inelastic modes in the truncated SVD
k^g	Number of global modes in the truncated SVD
k	Noiseless term of covariance function
$\boldsymbol{K}$	Covariance matrix without noise
$\tilde{\mathbf{k}}$	Vector of correlations between test inputs and observations
$\mathbf{K}_0$	Initial stiffness matrix
$\mathcal{L}$	Likelihood function
M	Number of FEM integration points
m	Additional candidate parameter iteration
$\boldsymbol{m}_t$	1st moment for gradient ascent
$\overline{m}$	Mean function hyperparameter
N	Total number of degrees of freedom in the domain Ω
N_{it}	Number of the training iteration
N^{max}_{it}	Maximum number of training iterations
N_t	Total number of time steps
N_T	Number of targets for regression improvement
N_p	Number of the surrogate parameter space
N_e	Number of elastic snapshots in $\mathbf{X}^e$

N_i	Number of inelastic snapshots in $\mathbf{X}^i$
N_c	Number of error indicator observations for GPRP
N_c^0	Initial number of error indicator observations for GPRP
N_c^{min}	Minimum number of additional error indicator observations for GPRP target improvement
N_c^{add}	Number of additional error indicator observations for GPRP target improvement
N_{GA}	Number of iterations in the gradient ascent algorithm
P	Parameter space
$\widehat{P}_{N_p}$	Surrogate parameter space
$\widehat{P}^*_{N_p}$	Decremented surrogate parameter space
$\mathbf{r}$	Residual vector
$\boldsymbol{R}$	Correlation matrix
$\tilde{s}_{\mathcal{J}}$	Standard deviation of predictive mean $\widetilde{\mathcal{J}}$ from GPRP
$\tilde{s}_E$	Standard deviation of predictive mean $\widetilde{E}$ from GPRC
t	Time step
T	Target value for GPRP improvement
$\mathbf{u}$	Displacement vector
$\mathbf{u}_r$	Reduced displacement vector
$\bar{\mathbf{u}}$	Macroscopic displacement vector
$\tilde{\mathbf{u}}$	Microscopic fluctuation displacement vector
$\boldsymbol{u}_{FOM}$	Matrix with full time-history of the displacements from the FOM solution
$\boldsymbol{u}_{ROM}$	Matrix with full time-history of the displacements from the ROM solution
$\mathbf{U}^e$	Elastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{U}}^e$	Elastic right-singular matrix of orthonormal basis vectors
$\mathbf{U}^i$	Inelastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{U}}^i$	Inelastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{U}}^g$	Global right-singular matrix of orthonormal basis vectors
$\boldsymbol{v}$	New 2nd raw moment for gradient ascent
$\hat{\boldsymbol{v}}$	2nd raw moment for gradient ascent
w	FEM integration weights
$\mathbf{x}$	Coordinates of a point in the mechanical model
$\bar{\mathbf{x}}$	Center of mass of the mechanical model
$\mathbf{X}$	Local snapshot matrix
$\mathbf{X}^e$	Local elastic snapshot matrix
$\widetilde{\mathbf{X}}^e$	Local elastic surrogate snapshot matrix
$\mathbf{X}^i$	Local inelastic snapshot matrix
$\widetilde{\mathbf{X}}^i$	Local inelastic surrogate snapshot matrix
$\mathbf{X}^g$	Global snapshot matrix
$\widetilde{\mathbf{X}}^g$	Global surrogate snapshot matrix
$\mathbf{Z}^e$	Elastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{Z}}^e$	Elastic right-singular matrix of orthonormal basis vectors
$\mathbf{Z}^i$	Inelastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{Z}}^i$	Inelastic right-singular matrix of orthonormal basis vectors
$\widetilde{\mathbf{Z}}^g$	Global right-singular matrix of orthonormal basis vectors

Greek Symbols

$\boldsymbol{\alpha}$	Reduced degrees of freedom vector
$\alpha_{\mathcal{J}}$	Z-score for confidence intervals for GPRP
α_E	Z-score for confidence intervals for GPRC

β	GPRP improvement termination factor
β_1	Exponential decay rate 1 for gradient ascent
β_2	Exponential decay rate 2 for gradient ascent
δ_{pq}	Kronecker delta
$\widetilde{\boldsymbol{\Delta\epsilon}}^M$	Random far field load path increment
$\boldsymbol{\epsilon}^M$	Far field load path
ϵ	Smoothing term for gradient ascent
ε^i	Inelastic energy tolerance
ε^g	Global energy tolerance
$\boldsymbol{\theta}_t$	Vector of hyperparameters to be optimized for GRPC
λ_{ROM}	Arc-length control load factor for the ROM
$\Delta\lambda_{ROM}$	Arc-length control load factor increment for the ROM
$\Delta\lambda_{FOM}$	Arc-length control load factor increment for the FOM
λ_l	Length-scale hyperparameter
μ	Parameter (or load location) in parameter space P
$\mu_{N_{it}}$	Chosen training parameter (or load location) in parameter space P in iteration N_{it}
$\widehat{\mu}$	Load location in surrogate parameter space $\widehat{P}^{N_p}$
μ^c	Candidate load location for GPRP improvement in parameter space P
$\mu^\star$	Test input parameters to construct GPRP
$\boldsymbol{\mu}^c$	Vector of candidate load locations
π	Acquisition function
σ_f	Signal variance hyperparameter
σ_n	Noise variance hyperparameter
$\boldsymbol{\Sigma}$	Singular value matrix
$\widetilde{\boldsymbol{\Sigma}}$	Truncated singular value matrix
$\boldsymbol{\Sigma}^e$	Elastic singular value matrix
$\widetilde{\boldsymbol{\Sigma}}^e$	Truncated elastic singular value matrix
$\boldsymbol{\Sigma}^i$	Inelastic singular value matrix
$\widetilde{\boldsymbol{\Sigma}}^i$	Truncated inelastic singular value matrix
$\boldsymbol{\tau}$	Vector of target factors for GPRP improvement
$\boldsymbol{\Phi}$	Final reduced basis matrix
$\boldsymbol{\Phi}_{N_{it}}$	Reduced basis matrix
φ	Normal cumulative distribution function
Ω	Domain of the finite element model
ω	ECM modified integration weights

Other Symbols and Conventions

$\|\cdot\|_2$	L2-norm of a vector
$\|\boldsymbol{x}\|_{\boldsymbol{K}_0}$	Stiffness-projected dot product
$size(\cdot)$	Number of elements in set
$\vert\cdot\vert$	Determinant of a matrix
$\mathrm{tr}(\cdot)$	Trace of a matrix
$\arg\max\limits_{x\in\mathbb{R}}(f(x))$	Argument $x \in \mathbb{R}$ that yields the maximum of function f

Acronyms

ANN	Artifical Neural Network
DEIM	Discrete Empirical Interpolation Method
ECM	Empirical Cubature Method
FE	Finite Element
FEM	Finite Element Method
FOM	Full-Order Model
GA	Gradient Ascent
GPR	Gaussian Process Regression
GPRP	Gaussian Process Regression for Prediction
GPRC	Gaussian Process Regression for Convergence
MOR	Model-Order Reduction
PDE	Partial Differential Equation
POD	Proper Orthogonal Decomposition
RB	Reduced Basis
ROM	Reduced-Order Model
RVE	Representative Volume Element
SPS	Surrogate Parameter Space
SVD	Singular Value Decomposition
SVM	Support Vector Machines

1 INTRODUCTION

1.1 General Overview

In the field of structural mechanics, the goal of model-order reduction (MOR) is to substitute a prohibitively expensive high-fidelity model with a significantly less complex reduced-order model (ROM) [26]. In the context of finite element modelling, reduction occurs in terms of fewer degrees of freedom or fewer integration points in which material response is computed. Methods have been developed to perform the actual dimensionality reduction, but they rely on information from the full-order model that has to be obtained in the so-called *offline* training phase. Assuming that a reduced model can be represented by a set of basis vectors, the task of the offline phase is to compute these vectors. These vectors are often determined by exhaustively sampling solutions from the full-order model, but that can be computationally expensive. Therefore, this thesis aims to develop and optimize the automatic and efficient sampling of full-order solutions to iteratively construct a ROM that accurately approximates the behaviour of a full-order nonlinear finite element model.

The case that is studied is a simply supported beam that is loaded in the downwards direction at an arbitrary point along the span. The ROM is constructed in the offline training phase by iteratively by running the full-order model (FOM) with samples of loads at different locations, and assembling a reduced basis (RB) using snapshots of the displacements in the entire beam due to each load case. When the ROM is accurate enough, the offline training phase is considered to be converged and is stopped. Then, in the *online* stage, the ROM can be run with a load at any location that will result in a response that accurately approximates the response of the FOM.

The high-level steps of the offline training phase procedure are summarized in Algorithm 1.1:

Algorithm 1.1: Offline training phase procedure

```
1  Select initial load location
2  Initialize reduced basis
3  while not converged do
4      Select load location
5      Run full-order model
6      Enrich reduced basis
7      Evaluate accuracy of the reduced-order model
8      if converged then
9          return Reduced basis
10     end
11 end
```

More specifically, this thesis will focus on lines 4 and 6 of Algorithm 1.1. A detailed procedure on iteratively updating the reduced basis while retaining and maximizing the amount of useful information (while remaining efficient) is developed. Furthermore, two methods for sampling the parameter space of the FOM are presented and compared.

1.2 Aim of the Research

The goal of this research is to develop and combine methods to reduce the computation time of finite element analyses that involve high-fidelity mechanical models by implementing an automatic and efficient procedure to sample the parameter space for load cases. This thesis will build upon some of the acceleration techniques implemented by Rocha [26] to minimize the computational cost of a multiscale/-multiphysics model. The results of the implementation using the aforementioned beam model can then be used as a stepping stone to implementing similar sampling methods to other high-fidelity models.

The following research questions have been formulated to guide and assess the research:

Research Questions

1. What is the best way to sample the parameter space so as to minimize the number of training cases?
2. Can training be performed in an optimized way by a heuristic algorithm that takes the maximum tolerable loss of accuracy as input?

1.3 Research Methodology

The focus of the literature study is to look into various methods of sampling (infinitely dimensional) parameter spaces, and determine which methods could be feasible to implement and investigate the performance of. Methods that make use of machine learning techinques are desirable for automatic and efficient load case selection for model-order reduction (MOR), so one of the chosen methods investigated in this thesis makes use of the Bayesian machine learning. The other method explored in this thesis does not make use of any machine learning techniques, but is an efficient way to explore a parameter space such as the one used in this thesis, which is the span of a beam where loads are applied in the downwards direction.

Following the literature review and choice of methods, computational tools are developed to implement two methods of greedily sampling the parameter space:

1. SPS Method: Greedy sampling of incrementally finer grids of points along the beam, where each individual grid is referred to as a *Surrogate Parameter Space* (SPS). Each grid is exhausted by every time selecting the load location that will add the most new information to the reduced model.
2. GPR Method: Greedy sampling using a Bayesian machine learning algorithm involving *Gaussian Process Regression* (GPR) to predict load locations that in each iteration add the most new information to the reduced model.

The computational implementation is programmed in C++ using the *Jem-Jive* toolkit, and the MOR framework developed by Rocha [26] is expanded to facilitate the iterative sampling of the parameter space. Following the implementation, the performance of the methods is assessed, and how the performance varies with different input parameters is investigated in a parametric study.

1.4 Thesis Outline

The thesis is structured such that all steps of the offline training phase procedure for each method are explained in a chronological manner. In chapter 2, the background and basics of MOR are presented, followed by brief explanations of the steps necessary to construct a reduced model. A large portion of the literature review is dedicated to the sampling of the parameter space. The following chapter, chapter 3, is kicked off with a necessary adjustment to the existing ROM framework in regards to how the reduced basis is iteratively assembled from the displacement snapshots. This is followed by thorough explanations of the two different methods of sampling the parameter space. The chapter is finished off by detailing the convergence modelling. Then, chapter 4 contains all the results that are gathered to answer the research questions, along with thorough discussions of observations that can be made from the results. Finally, chapter 5 summarises the findings of the thesis and the conclusions that can be made to answer the research questions, before finishing up with recommendations for further research.

2 LITERATURE REVIEW

2.1 High-Fidelity Models

In today's technological world numerical simulations have become an essential part of the design process in every field of engineering, and computing power has been increasing rapidly every year, enabling more and more complex simulations to be carried out. However, there are still problems that are too complex to be solved by conventional computers and algorithms, which has led to the rise of model-order reduction (MOR). MOR was originally developed in the area of systems and control theory in order to reduce the complexity of the problems, but is today a flourishing field of research used in all areas that involve numerical modelling, especially in electronics and fluid and structural mechanics [30]. The goal of MOR is to substitute an expensive high-fidelity numerical model with a reduced-order model (ROM) that accurately approximates the full-order model (FOM) and is inexpensive to compute. Figure 2.1 shows one of the simplest examples of model-order reduction techniques, where the mesh is made extremely coarse compared to the FOM, but still contains enough information to show that it is a model of bunny.

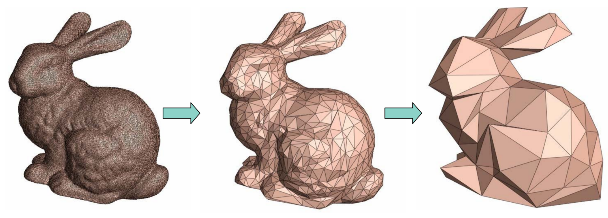


Figure 2.1: Graphical illustration of simple MOR [30]

An example of an application of MOR in structural mechanics is the numerical modelling of laminated composites for wind turbine blades performed by Rocha [26], and is shown in Figure 2.2. This is a highly heterogeneous material combination with large stiffness gradients, resulting in significantly complex mechanical behaviour. Additionally, due to the small diameter of the order of a few micrometers, the microscale behaviour of the material is important to understand in order to produce efficient designs. Wind turbines are subjected to highly dynamic loads, undergo considerable temperature changes, and are also exposed to moisture ingression. All of these effects combined result in a model of excessively high fidelity that is prohibitively expensive to compute for conventional computers and algorithms.

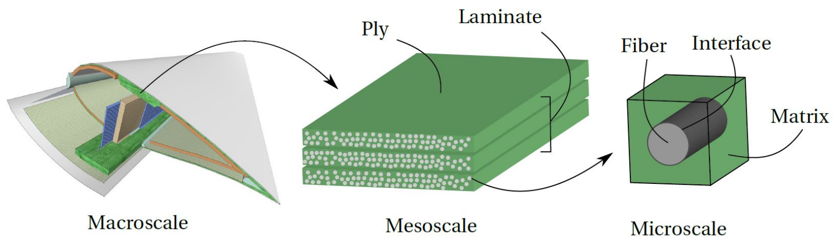


Figure 2.2: Laminated composite observed at macro-, meso-, and microscale [26]

The macroscale domain Ω of the laminated composite is formed by the repetition of a heterogeneous microstructure, as illustrated in Figure 2.3. The repeated microstructure is referred to as a representative volume element (RVE) that consists of unidirectional fibers surrounded by resin. In the MOR process for a multiscale model, the RVE is used to construct the reduced model, and is then homogenized to the macroscale.

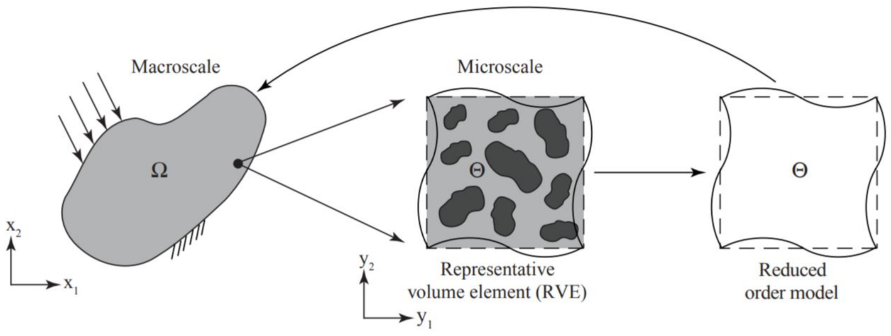


Figure 2.3: Macro- and microscale structures and RVE ROM [33]

Another example of multiscale model is a porous metal material, as illustrated by the RVE in Figure 2.4. This RVE is similar to that of a laminated composite, except that the fibers are replaced with pores, and there is no interface between the fibers and the resin. This model is thus likely to be significantly cheaper, but the presence of the pores still makes the finite element discretization complex enough to necessitate MOR.

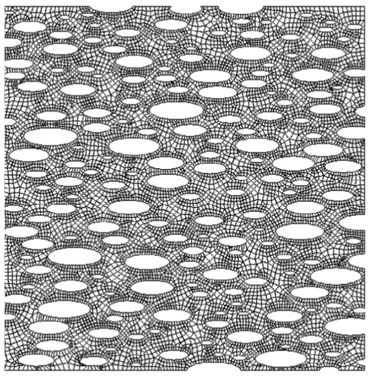


Figure 2.4: RVE for a porous metal material [11]

2.2 Model-Order Reduction

Finite element analysis using the types of models discussed previously can involve exceedingly high computation times, so development into methods to reduce the complexity of such models was prompted in order to accelerate such analyses. Model-order reduction techniques aim to construct reduced-order models (ROMs) of as low complexity as possible that still manage to represent full-order models (FOMs) as accurately as possible; the ROM should deliver a good approximation of the FOM for any set of parameters (e.g. load paths, flow fields). Projection-based reduced-order modeling techniques are widely used, and the most popular of these is Galerkin Projection used in combination with Proper Orthogonal Decomposition (POD) [28]. This method efficiently compresses snapshots of full-order solutions obtained using samples from the parameter space, and projects the full-order problem onto the so-called reduced basis. After constructing a reduced basis, further reduction techniques can be applied to this basis to create a hyper-reduced-order model (HROM), such as the Empirical Cubature Method (ECM) [28]. This hyper-reduction technique heuristically determines samples of the integration points of the reduced basis and computes the constitutive response.

2.2.1 Full-Order Finite Element Problem

This section briefly outlines the Finite Element Method (FEM) as described by Rocha [26], Rocha et al. [28] and Nikishkov [18]. FEM is a numerical method for solving field problems that are generally described by partial differential equations (PDEs). The first step is to subdivide the domain Ω of the FE model into a number of discrete elements that have a number of nodes that connect the elements to each other. The next step is to derive the interpolation functions that correspond with the geometry and physics of the FE model, before deriving the element equations using the Galerkin method, and using this to derive the overall system of equations. Boundary conditions are also applied to the boundary Γ of the domain Ω usually in the form of Neumann or Dirichlet boundary conditions to form the final part of the equilibrium problem. With the global equilibrium problem described, the final step is to solve for the unknowns at the nodes using direct or iterative methods (e.g.Newton-Raphson solver).

The global equilibrium problem in its full, weak form is expressed as:

$$\mathbf{r} = \mathbf{f}^{\Omega}(\mathbf{u}) - \mathbf{f}^{\Gamma} = \mathbf{0} \tag{2.1}$$

where $\mathbf{r} \in \mathbb{R}^N$ is the residual vector, $\mathbf{f}^{\Gamma} \in \mathbb{R}^N$ is the external force vector, and $\mathbf{f}^{\Omega} \in \mathbb{R}^N$ is the internal force vector. The solution of the global equilibrium problem is obtained by iteratively correcting the displacement vector until $\mathbf{r} = 0$ using the following expression:

$$\Delta\mathbf{u} = \mathbf{u}_{\mathrm{n}} - \mathbf{u}_{\mathrm{o}} = -\mathbf{K}_{\mathrm{o}}^{-1}\mathbf{r}_{\mathrm{o}} \tag{2.2}$$

where o and n indicates that values stem from the old or new analysis increments, respectively, and $\mathbf{K} \in \mathbb{R}^N$ is the global tangent stiffness matrix:

$$\mathbf{K} = \frac{\partial \mathbf{f}^{\Omega}}{\partial \mathbf{u}} \tag{2.3}$$

where the global internal force vector $\mathbf{f}^{\Omega}$ is computed as follows:

$$\mathbf{f}^{\Omega} = \int_{\Omega} \mathbf{f} d\Omega \approx \sum_{i}^{M} \mathbf{f}(\mathbf{x}_i) w_i \tag{2.4}$$

with $\mathbf{f} \in \mathbb{R}^N$ being a sparse internal force vector at a given material point, M is the number of integration points considered in the domain Ω, with w as the corresponding integration weight.

2.2.2 Proper Orthogonal Decomposition

POD is a statistical tool that was first introduced by Pearson [22], and it has been continuously developed since then by various authors. It is referred to in other field of applications as Principal Component Analysis [22], Karhunen-Loève transform [29], Hotelling transform [14], Empirical Orthogonal Functions [23], and more. In the field of structural engineering it is known as POD, and it is a way to explore the internal structure of given data and compress it to a signficiantly lower order representation of the same data. This is illustrated in Figure 2.5, where $\mathbf{\Phi}$ is the reduced basis, and $\mathbf{X}$ is a matrix of snapshots of the full-order solutions.

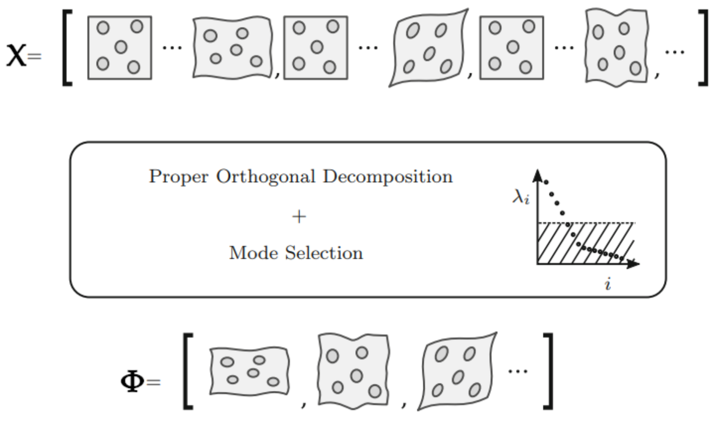


Figure 2.5: Outline of POD [34]

The original full-order FEM problem with N degrees of freedom (Eq. (2.1)) is projected onto the reduced basis $\mathbf{\Phi} \in \mathbb{R}^{N \times k^g}$ using the Galerkin projection constraint ($\mathbf{\Phi}^T \mathbf{r} = 0$), to obtain the reduced equilibrium problem:

$$\mathbf{\Phi}^T(\mathbf{f}^\Omega - \mathbf{f}^\Gamma) = 0 \Rightarrow \mathbf{f}_r^\Omega - \mathbf{f}_r^\Gamma = 0 \tag{2.5}$$

which reduces the problem to solving for $k^g \ll N$ mode contributions $\boldsymbol{\alpha} \in \mathbb{R}^k$. Then, the full-order displacement field is recovered as [28]:

$$\mathbf{u}_r = \mathbf{\Phi}\boldsymbol{\alpha} \tag{2.6}$$

$\boldsymbol{\alpha}$ contains the degrees of freedom of the reduced space and represents the contribution of k^e elastic and k^i inelastic displacement modes, and $k^g = k^e + k^i$. Eq. (2.6) shows that $\mathbf{u}$ is approximated as a linear combination of k^g basis modes, as illustrated in Figure 2.6.

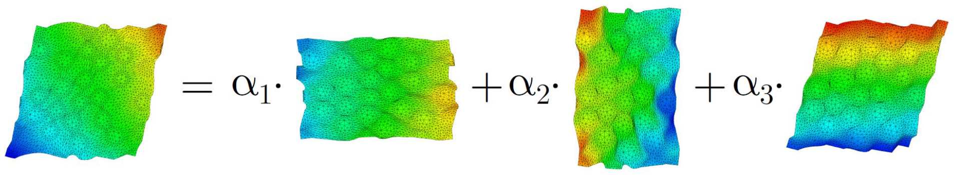


Figure 2.6: Reduced basis modes contributions [26]

A slightly modified version of the method can be used when dealing with a concurrent multiscale analysis (FE^2) where it is usual to split the displacement (or strain) field into two parts: a macroscopic part $\bar{\mathbf{u}}$ and a microscopic fluctuating part $\tilde{\mathbf{u}}$ [10, 11]:

$$\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}, t) + \tilde{\mathbf{u}}(\mathbf{x}, t) \tag{2.7}$$

with $\bar{\mathbf{u}}$ given by:

$$\bar{\mathbf{u}}(t) = \boldsymbol{\epsilon}^M(t)(\mathbf{x} - \bar{\mathbf{x}}) \tag{2.8}$$

where t is the time, $\mathbf{x}$ is the coordinate, $\bar{\mathbf{x}}$ is the center of mass of the RVE, and $\boldsymbol{\epsilon}^M$ describes the load path of the applied far field load. If utilizing this definition of the displacement field, then the reduced displacement field would turn into:

$$\mathbf{u} \approx \mathbf{u}_r = \bar{\mathbf{u}} + \boldsymbol{\Phi}\boldsymbol{\alpha} \tag{2.9}$$

which is also illustrated in Figure 2.7 in 2D. These macroscale contributions represent the linear elastic responses to uniaxial loads in x- and y-directions, and a shear load. This splitting ensures that the solutions from the elastic regime are captured and implemented sufficiently.

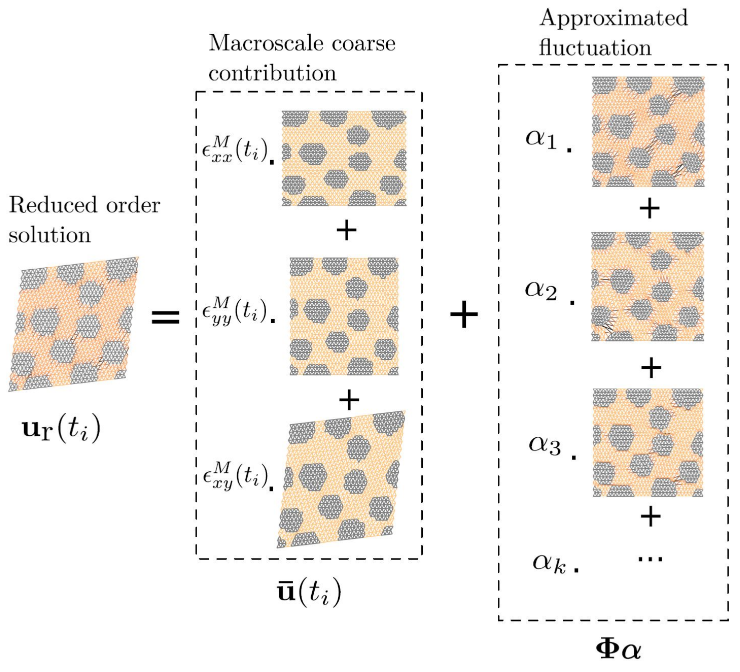


Figure 2.7: Decomposition of macroscale part and microscale fluctuation [10]

The result of POD is a reduced model of size $k^g \ll N$ that accurately approximates the constitutive response of the full-order model for any set of parameters. However, constructing the reduced basis requires sampling parameters from a potentially infinite-dimensional parameter space in an efficient way.

2.2.3 Hyper-reduction

While initial reduction steps can reduce the number of degrees of freedom by a factor of thousands, the computation of the internal force vector $\mathbf{f}_\mathrm{r}^\Omega$ can still be prohibitively expensive for complex mechanical models because of the high number of integration points in the finite element model that must be integrated over. Therefore, hyper-reduction techniques are being developed such as the Empirical Cubature Method (ECM) [12, 28]. This method aims to reduce the number of integration points considered in the domain that accurately represents the full integrand. The reduced set of integration points are determined along with the corresponding weights to optimize efficiency without a significant loss of accuracy. ECM reduces the computation of the global internal force vector to $m << M$ integration points:

$$\mathbf{f}^\Omega = \int_\Omega \mathbf{f} d\Omega \approx \sum_{i=1}^{m} \mathbf{f}(\mathbf{x}_i)\omega_i \tag{2.10}$$

This equation is very similar to Eq. (2.4), except M is replaced by m and w_i by ω_i, which are integration weights that are modified by the ECM.

An alternative hyper-reduction method is the Discrete Empirical Interpolation Method (DEIM) developed by Chaturantabut and Sorensen [4]. POD reduces the number of degrees of freedom and ECM reduces the number of integration points, but the evaluation of nonlinear responses remains a computational bottleneck [9]. To overcome this inefficiency, DEIM attempts to approximate a nonlinear function by combining POD-Galerkin projection with interpolation, resulting in an approximation for the full-order global internal force vector $\mathbf{f}^{\Omega}$.

2.3 Reduced Basis

2.3.1 Constructing the Reduced Basis

As mentioned in 2.2.2 Proper Orthogonal Decomposition, the reduced space is built from snapshots of the full-order solutions. The type of snapshots used depends on the problem being solved, but which type is used generally does not affect the performance of a reduced-model. Goury et al. [10] uses strain snapshots, Hernández et al. [12] uses stress snapshots for structural mechanics, and Kalashnikova et al. [16] uses pressure snapshots for fluid mechanics, but this thesis will use displacement snapshots as done in Rocha [26].

The displacement snapshots are basically the displacement values at every degree of freedom at every time step, and after solving the full-order model for a specific parameter, they are collected in a matrix $\mathbf{X}$ of size $N \times N_t$, where N is the number of degrees of freedom of the full-order model, and N_t is the number of time steps [21], and it assumed that the number of time steps remains the same for each training iteration. For now, one training iteration is defined as the process of selecting a parameter, collecting snapshots using the full-order model, and then updating the basis. $\mathbf{X}$ is referred to as a *local* snapshot matrix because it corresponds to the solutions for the current iteration N_{it}. As more snapshots are generated using N_{it} sets of parameters, the snapshot matrices from all the iterations of the offline training phase are collected in a *global* snapshot matrix $\mathbf{X}^g$:

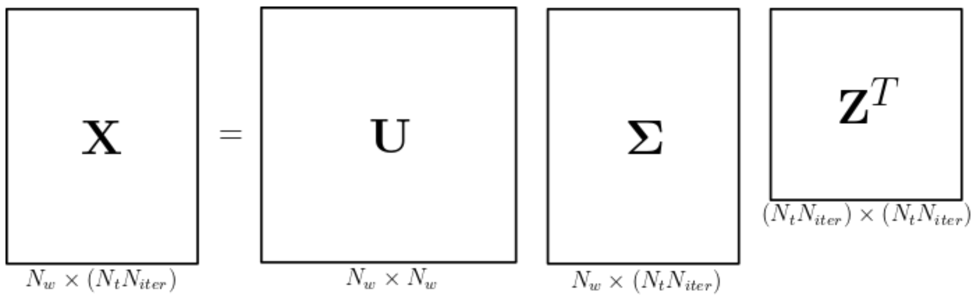
$$\mathbf{X}^g = [\mathbf{X}_0^g, \dots, \mathbf{X}_{N_{it}}^g] \tag{2.11}$$

Once the desired number of snapshots have been collected, they are compressed using the method of Singular Value Decomposition (SVD). This is the most important part of the POD procedure as this is the step where the most important data from the high-fidelity solutions are collected in the first k^g columns of $\mathbf{U}$, which is obtained by applying the SVD to the snapshot matrix $\mathbf{X}^g$:

$$\mathbf{X}^g = \mathbf{U\Sigma Z^T} \tag{2.12}$$

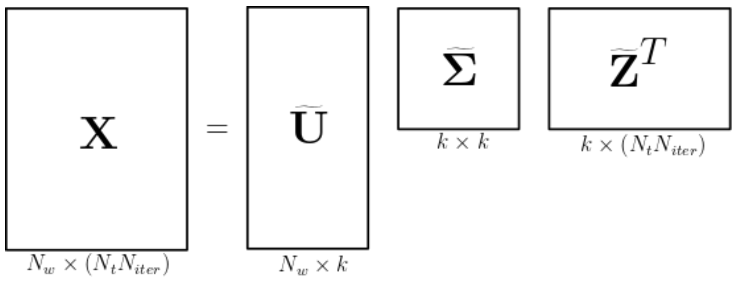
where $\mathbf{U}$ and $\mathbf{Z}$ are the left-singular and right-singular matrices of orthonormal eigenvectors (or basis vectors), respectively, and $\mathbf{\Sigma}$ is a rectangular diagonal matrix of singular values of $\mathbf{X}^g$ [28]. The singular values are arranged in descending order, and their magnitudes signify the importance of the data in the columns of the basis matrix, and at this point the matrices are still of very high order. Therefore, to construct an accurate and efficient reduced basis, only the k^g first columns will be used, as illustrated in Figure 2.8.

However, the SVD is a process that ignores the physical meaning of the data in the snapshots, but the reality is that one part of the snapshots corresponds to linear elastic solutions of the full order FE problem, and another part to inelastic solutions. In the SVD, which basis vectors are considered to be important is a purely statistical consideration, so in case the data from e.g. the elastic regime is not represented well in the snapshots, then the resulting basis vectors with the largest singular values (i.e. the most important modes) would not represent any information from this range. This would cause the reduced model to be either inaccurate or inefficient because a large number of basis vectors may have to considered in order to construct a basis that represents the elastic response well enough, which could greatly increase the computation cost of the reduced model [11].



$\mathbf{X}$
$=$
$\mathbf{U}$
$\mathbf{\Sigma}$
$\mathbf{Z}^T$
$N_w \times (N_t N_{iter})$
$N_w \times N_w$
$N_w \times (N_t N_{iter})$
$(N_t N_{iter}) \times (N_t N_{iter})$

(a) Full SVD



$\mathbf{X}$
$=$
$\widetilde{\mathbf{U}}$
$\widetilde{\mathbf{\Sigma}}$
$\widetilde{\mathbf{Z}}^T$
$N_w \times (N_t N_{iter})$
$N_w \times k$
$k \times k$
$k \times (N_t N_{iter})$

(b) Truncated SVD

Figure 2.8: Full and Truncated SVD

To prevent this, Hernández et al. [11] proposes to decompose the local snapshot matrix $\mathbf{X}_{N_{it}}$ into an elastic part ($\mathbf{X}^{\mathrm{e}}$) and an inelastic part ($\mathbf{X}^{\mathrm{i}}$). In the work of Rocha et al. [28], the SVD is performed on the two parts separately before combining them together into one basis: $\mathbf{\Phi} = [\mathbf{U}^{\mathrm{e}} \quad \mathbf{U}^{\mathrm{i}}]$.

2.3.2 Updating the Reduced Basis

Ideally, $\mathbf{\Phi}$ would be computed only once from a snapshot matrix $\mathbf{X}$ obtained from exhaustively sampling the parameter space. In practice, however, this would lead to a prohibitively expensive training phase if the parameter space is too large. The alternative is to resort to a greedy training procedure in which an initial basis is computed and is subsequently updated in a number of training iterations to include information from newly-obtained snapshots. An important issue that has to tackled when collecting more snapshots to add to the basis is exactly *how* to update the basis. The SVD is often one of the largest contributions to computation cost of the POD procedure [20] because the global snapshot matrix $\mathbf{X}^{g}_{N_{it}} \in \mathbb{R}^{N\times(N_t N_{it})}$, gets excessively large when carrying out more trainings with new parameters (Eq. (2.11)), especially when considering a large amount of time steps.

An alternative approach to updating the basis is to add an additional step in the POD-greedy process as proposed by Paul-Dubois-Taine and Amsallem [21], where a global *surrogate* snapshot matrix $\widetilde{\mathbf{X}}^{g}_{N_{it}}$ is assembled from local surrogate matrices $\widetilde{\mathbf{X}}_{N_{it}}$. The following explanation will not consider the elastic-inelastic decomposition outlined in the previous subsection.

First, the SVD is performed on the local snapshot matrix $\mathbf{X}_{N_{it}}$ and the resulting matrices are truncated to the first $k^{g}_{N_{it}}$ singular values:

$$\mathbf{X}_{N_{it}} = \mathbf{U}_{N_{it}}\mathbf{\Sigma}_{N_{it}}\mathbf{Z}^{T}_{N_{it}} \approx \widetilde{\mathbf{U}}_{N_{it}}\widetilde{\mathbf{\Sigma}}_{N_{it}}\widetilde{\mathbf{Z}}^{T}_{N_{it}} \tag{2.13}$$

Then, a local surrogate snapshot matrix $\widetilde{\mathbf{X}}_{N_{it}}$ is constructed that represents the original high-dimensional $\mathbf{X}_{N_{it}}$, but has significantly smaller dimension:

$$\mathbf{X}_{N_{it}} \approx \widetilde{\mathbf{X}}_{N_{it}} = \widetilde{\mathbf{U}}_{N_{it}}\widetilde{\mathbf{\Sigma}}_{N_{it}} \in \mathbb{R}^{N\times k^{g}_{N_{it}}} \tag{2.14}$$

Lastly, the global surrogate snapshot matrix $\widetilde{\mathbf{X}}^{g}_{N_{it}}$ is then assembled as:

$$\widetilde{\mathbf{X}}^{g}_{N_{it}} = \left[\widetilde{\mathbf{X}}_{N_1}, \ldots, \widetilde{\mathbf{X}}_{N_{it}}\right] \in \mathbb{R}^{N \times \left(\sum_{j=1}^{N_{it}} k_j\right)} \tag{2.15}$$

The truncation $k^{g}_{N_{it}}$ can be determined in a few different ways. Rocha et al. [28] used a constant number of elastic and inelastic modes to ensure that both types of behaviour were captured accurately. However, in that case the author utilized knowledge-based sampling, which requires a low number of number of training cases, so constructing the basis with a set number of modes is not that punishing in terms of computational efficiency. Additionally, the number of modes required to accurately describe a load case varies in each iteration based on the response. In the case that many training cases are required, the number of modes should be kept as low as possible for each type of response (elastic or inelastic) and in each iteration. The most common way is to determine the truncation by using an energy criterion which ensures that a certain fraction of the snapshot energy is retained in the surrogate snapshots [10, 16, 21, 31], where the energy of a matrix is the sum of its singular values. Using this criterion, $k^{g}_{N_{it}}$ is chosen to be the smallest integer such that:

$$\frac{\sum_{j=1}^{k^{g}_{N_{it}}} \Sigma_{jj}}{\sum_{j=1}^{N} \Sigma_{jj}} \geq \varepsilon \tag{2.16}$$

where $0 \leq \varepsilon \leq 1$ and represents the fraction of the retained snapshot energy desired in the reduced basis. ε is usually chosen to be at least 0.99, though it depends on the problem.

2.4 A Priori Sampling

Besides the issues of efficiently constructing and updating the basis as discussed previously, one of the most important issues during the offline training phase is how to efficiently sample the parameter space, which in many cases could be infinitely dimensional [10]. Efficient sampling methods must therefore address two challenges. Firstly, a systematic and smart strategy for selecting where and how many sample parameters to generate is necessary. Secondly, a strategy that is sufficiently scalable should be utilized, so that high-dimensional parameter spaces can be sampled efficiently [8]. One may utilize knowledge of the full-order problem to predict important and representative regions of the parameter space, as done in Rocha et al. [27], which addresses the first challenge and makes the second one negligible, but often such predictions about the behaviour of the full model cannot be made because such prior knowledge does not exist. It is therefore desirable to develop a reliable method of sampling the parameter space that does not rely on prior knowledge of the model that is to be reduced. This section explores various methods to sample the potentially infinite-dimensional parameter space that address one or both challenges. First, a priori sampling strategies will be explored, followed by progressive and greedy sampling strategies.

2.4.1 Knowledge-Based Sampling

This is by far the most efficient way of constructing a reduced basis because it saves a lot of computation time that would otherwise be spent to e.g. figure out which training cases add the most useful information. Using e.g. a greedy algorithm (2.6 Greedy Sampling), a reduced basis is made up of snapshot solutions that add as much new information as possible, but not all that information is actually useful for many load cases that will be applied to the ROM in the online phase. So with knowledge of which load cases will be applied to the online reduced model, only the training cases that add information that describes these load cases needs to be used.

Rocha et al. [27] had knowledge of what sort of elastic, plastic and fracture response was expected, so the load cases shown in Figure 2.9 were chosen to accurately represent the expected behaviour. A risk of this sort of sampling is that insufficient knowledge or flawed understanding of the expectations can result in inaccurate reduced models.

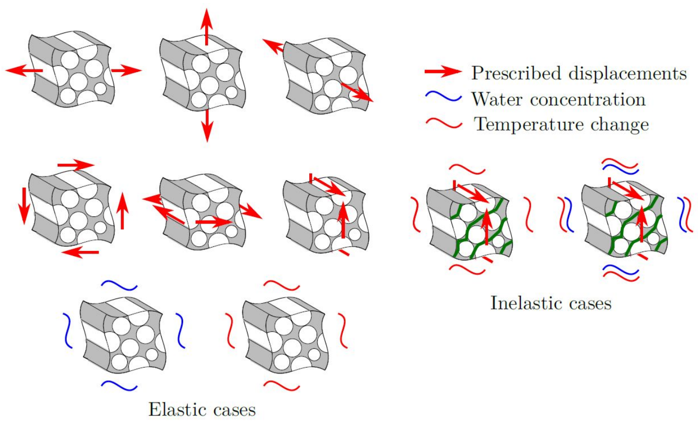


Figure 2.9: Load cases used by Rocha et al. [27] to train a ROM

2.4.2 Uniform Sampling

Uniform sampling is a straightforward way of sampling the parameter space, depending on the problem. It is a sure way of exhaustively sampling the parameter space, but for high-dimensional problems, the computational expense can get excessively high. There are many ways to discretize the parameter space uniformly, and two of them are shortly outlined in this section.

Simply Supported Beam
For a simply supported beam, the parameter space can be discretized into a grid of uniform spacing, as shown in Figure 2.10. The spacing depends on the desired accuracy with consideration of the computation time.

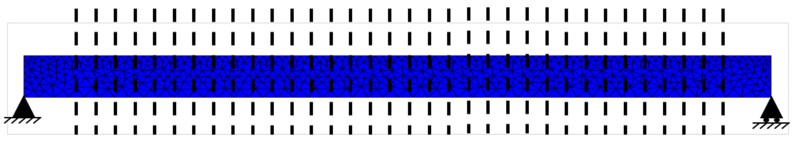
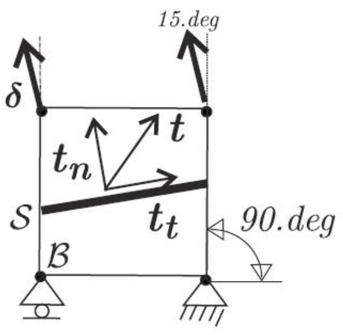
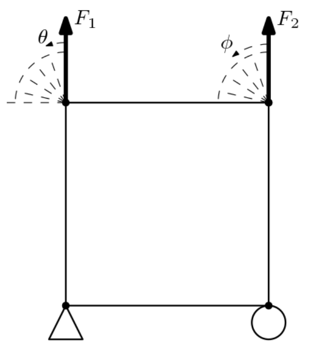


Figure 2.10: Uniformly spaced grid for a simply support beam

Representative Volume Element (RVE)
Figure 2.11a shows a typical way to apply loads to an RVE. Here, the parameter space is discretized by dividing the range of angles into a grid of uniform angle spacing, as shown in Figure 2.11b. The two angles can either vary together or separately. How the angles vary in relation to each other and the spacing depends on the desired accuracy with consideration of the computation time.



(a) Loads applied to an RVE [19]



(b) Uniformly spaced grid for load angles

Figure 2.11: RVE load example

2.4.3 Random Sampling

Random sampling is another simple way to probe the parameter space. For the beam model, this would be a number of randomly selected load in the downwards direction along the span, and for the RVE shown in the previous section it would be loads applied with random magnitudes in random directions. However a random sampling algorithm could easily fail to capture important regions in the parameter space [8]. Including enough samples to make sure that these regions are recognized could be too computationally expensive.

2.5 Progressive Non-Greedy Sampling

2.5.1 Constrained Random Sampling for RVE Modeling

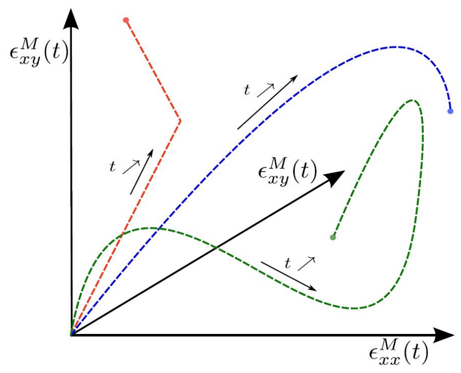
Goury et al. [10] proposes a method that is near-random, but employs a contraint that ensures that additional snaphots always add more information to the reduced basis. First of all, the parameters in Goury et al. [10] are represented by load paths (Figure 2.12a) described by:

$$\boldsymbol{\epsilon}^M(t) = \begin{pmatrix} \epsilon^M_{xx}(t) & \epsilon^M_{xy}(t) \\ \epsilon^M_{xy}(t) & \epsilon^M_{yy}(t) \end{pmatrix} \tag{2.17}$$

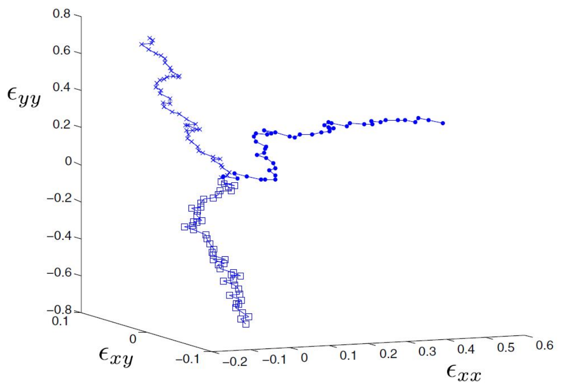
which are applied to the RVE as far field loads. To make sure that the further incrementation of the load paths add more information to the system, each random load increment, given by:

$$\widetilde{\boldsymbol{\Delta\epsilon}^M}(t_n) = \begin{pmatrix} \widetilde{\Delta\epsilon_{xx}}(t_n) & \widetilde{\Delta\epsilon_{xy}}(t_n) \\ \widetilde{\Delta\epsilon_{xy}}(t_n) & \widetilde{\Delta\epsilon_{yy}}(t_n) \end{pmatrix} \tag{2.18}$$

is forced to dissipate energy in the structure at every single time step t_n. This restriction is enforced by making sure that at each time step, the tension in either x- or y-direction has to increase, or the shear has to increase. Three examples of load paths generated by this procedure are shown in Figure 2.12b. With enough paths and enough iterations in each path, the parameter space can be sampled exhaustively, however this would be prohibitively expensive, but if efficiency is desired, the parameter space cannot be exhaustively probed.



(a) Examples of load paths for the nonlinear RVE problem



(b) Examples of load paths obtained using the constrained random sampling procedure

Figure 2.12: Load path examples for the RVE problem in Goury et al. [10]

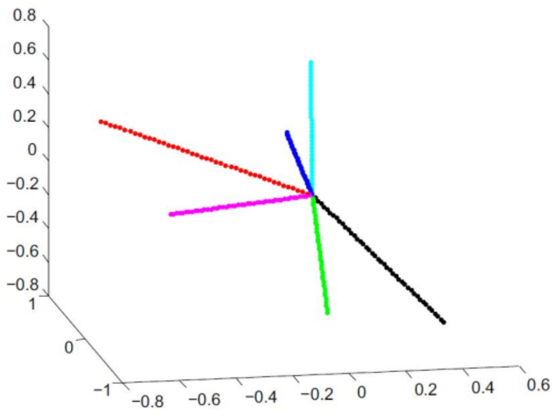
2.6 Greedy Sampling

To circumvent the issues in regards to computational effort that arise from sampling a high-dimensional parameter with the methods outlined in the previous sections, greedy sampling procedures have been proposed by various authors. A greedy algorithm is a mathematical process that iteratively finds locally optimal solutions that contribute to finding the globally optimal solution in the end. In the context of sampling parameters to construct a reduced basis, an optimal training set is constructed by sequentially choosing the parameters in the parameter space that yield the highest error at the current stage of the process, and thus would add the most new information to the reduced basis. The more accurately the greedy sampling chooses the parameter with the highest error, the more information is added in every iteration, which would generally lead to a more efficient offline training phase. However, depending on how the parameter is found, there could be cases where it is more efficient to rapidly select a suboptimal parameter in each iteration, and execute more iterations to add the same amount of information.

2.6.1 Surrogate Parameter Space

In the greedy sampling strategy proposed by Goury et al. [10], the high-dimensional parameter space P is subdivided into a sequence of low-dimension surrogate parameter spaces (SPS) of increasing dimension so that the parameter space can be sampled more efficiently. In that work, the parameter of interest is the far field load path (Eq. (2.17)). The initial SPS $\widehat{P}_0$ is made up of a number straight load paths as shown in Figure 2.13a. What varies are the two angles that describe the direction of the load, and thus the dimension of this SPS is 2. The following SPS $\widehat{P}_1$ is made up of load paths with two successive straight paths and the angles describing the second path are different than those for the first path, as shown in Figure 2.13b. This SPS has five dimensions because there are two angles for each path, and one additional dimension for the distance to where the second path is initiated.

Each additional increment of this hierarchical sequence of low-dimensional SPSs $\widehat{P}_{N_p}$ will have $N_p + 1$ distinct straight load paths and $2 \times N_p + (N_p - 1)$ dimensions. If a reduced model constructed from snapshots from SPS $\widehat{P}_{N_p}$ describes well every solution in SPS $\widehat{P}_{N_p+1}$, then the reduced model is assumed to be satisfactory, and no further increments are required.


(a) Examples of load paths in $\widehat{P}^1$

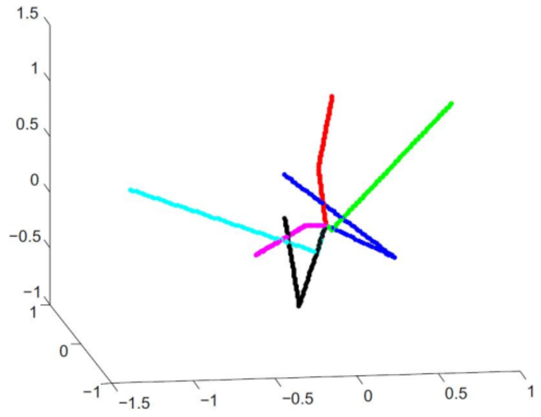

(b) Examples of load paths in $\widehat{P}^1$

Figure 2.13: Examples of load paths in two levels of the SPS in Goury et al. [10]

In the work by Goury et al. [10], the load paths are chosen using Gaussian Process Regression. For the beam model used in this thesis, this SPS method can be used as a greedy sampling strategy by itself due to the simplicity of the parameter space. For such a model, the parameter of interest is the location of the load applied in the downwards direction along the top of the beam. The SPSs $\widehat{P}_{N_p}$ can thus be built up by simply subdividing the beam into grids with uniform spacing that become progressively finer, as shown in Figure 2.14. The greedy sampling is performed in every iteration N_{it} and every SPS $\widehat{P}_{N_p}$ after the intial ones by computing an a posteriori error indicator at each point on grid, and selecting the load location with the maximum error indicator to enrich the basis further in N_{it}:

$$\mu_{N_{it}} = \underset{\mu \in \widehat{P}_{N_p}}{\arg\max} \mathcal{I}(\mu; \mathbf{\Phi}) \tag{2.19}$$

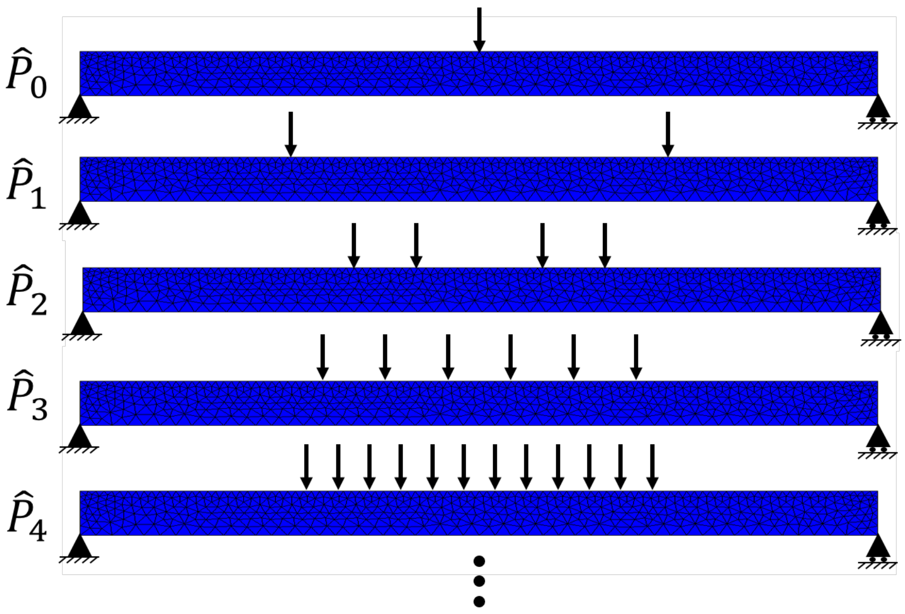


Figure 2.14: SPSs for the simply support beam model

As mentioned, an a posteriori error indicator $\mathcal{I}(\mu)$ is used when selecting the next training case because the exact error cannot be computed without the high-fidelity solution, which is only being obtained when looking to enrich the reduced basis. Goury et al. [10] proposes an error indicator that is a time-independent norm of the full-order residual computed by the reduced model before projection:

$$\mathcal{I}(\mu;\mathbf{\Phi}) = \sqrt{\frac{\sum_{k=0}^{k=N_t} \left\|\mathbf{r}(t_k,\ \mu;\ \mathbf{\Phi})\right\|_2^2}{N_t+1}} \tag{2.20}$$

where N_t is the total number of timesteps $t_k = 1,\dots,N_t$, $\|\cdot\|_2$ is the L2-norm, and $\mathbf{r}$ is the full-order residual at timestep t_k given by:

$$\mathbf{r} = \mathbf{f}^{\Omega} - \mathbf{f}^{\Gamma} \tag{2.21}$$

2.6.2 Gaussian Process Regression

In order to more optimally choose the next parameter (or load location) to use in the next training iteration, Bayesian machine learning using Gaussian Process Regression (GPR) can be utilized. A regression is simply a prediction of variables based on the measurement of other variables, and GPR relies on Bayesian probability theory to make more accurate predictions [3] than e.g. linear, parabolic or sinusoidal regression, because the latter methods most often rely on predetermined model structures such as parametric functions. GPR is a popular non-parametric regression method that is constructed based almost solely on the given data [13], and it takes into account uncertainty in every aspect of the model, which makes it a good regression technique for problems with significant uncertainties, such as the estimation of the error of a ROM. GPR also has the added benefit that it not only provides an estimate of the mean of the regression, but it also generates a variance (gray area in Figure 2.15, which is valuable when determining the accuracy of a ROM. Figure 2.15 shows an example of a regression computed with data generated with roughly sinusoidal behaviour, as well as its 70% and 95% confidence intervals, which are roughly one and two standard deviations away from the mean, respectively. The figure shows how the error at the edges increases because there are much fewer observations made there.

In order to determine the parameter μ to use in the next training iteration $(N_{it}+1)$, the error indicator $\mathcal{I}$ (Eq. (2.20)) will be predicted as a function of the parameters in the parameter space P. The initial step in the computation of the regression is to gather an initial set of N_m observations of the error indicator

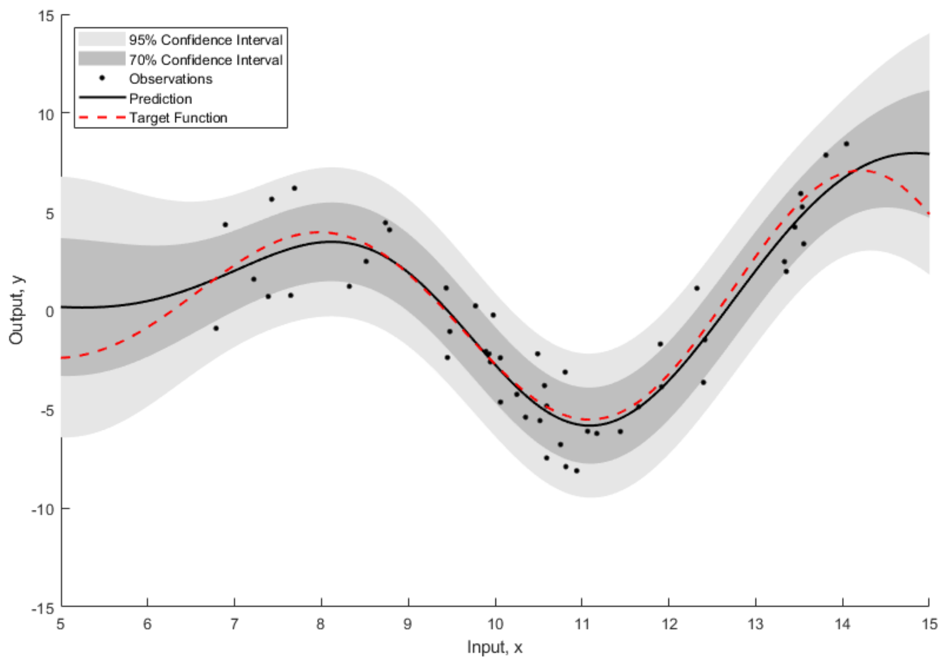


Figure 2.15: Example of Gaussian Process Regression

using a few random values of the input parameter. The approximation of these observations by the GPR can be expressed as:

$$\mathcal{I}(\mu;\mathbf{\Phi}) = f(\mu) + \epsilon(\mu) \tag{2.22}$$

where $f(\mu)$ is the unknown function that is to be approximated, and $\epsilon(\mu)$ is the observation noise [1].

The parameter that leads to the highest error indicator at iteration N_{it} of the training procedure will be selected as:

$$\mu_{N_{it}} = \arg\max_{\mu \in P} \mathcal{I}(\mu;\mathbf{\Phi}) \tag{2.23}$$

as it is the one that will add the most new information to the reduced model. In the case of a homogeneous simply supported beam, the parameter space is simply the area along the top where the load may be applied in the vertical direction. Initially, the error indicator is computed at a few points along the beam, and those samples are interpreted as a sample from some multivariate infinite-dimensional data set, but what establishes the relationship between the various observations is the *covariance function* [7]. A popular choice of the covariance function (often referred to as *kernel function*) is the *squared exponential* [24]:

$$\operatorname{cov}\left(\mathcal{I}_p, \mathcal{I}_q\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{(\mu_p - \mu_q)^2}{\lambda_l^2}\right] + \sigma_n^2 \delta_{pq} \tag{2.24}$$

$$\operatorname{cov}\left(\mathcal{I}_p, \mathcal{I}_q\right) = k(\mu_p, \mu_q) + \sigma_n^2 \delta_{pq} \tag{2.25}$$

or collected in matrix form:

$$\mathbf{C}\left(\mathcal{I}\right) = \sigma_f^2 \boldsymbol{R}\left(\boldsymbol{\mu}, \boldsymbol{\mu}\right) + \sigma_n^2 \boldsymbol{I} = \boldsymbol{K}\left(\boldsymbol{\mu}, \boldsymbol{\mu}\right) + \sigma_n^2 \boldsymbol{I} \tag{2.26}$$

where σ_f^2 is the maximum allowable input signal variance, λ_l^2 is a length-scale, σ_n^2 is the variance of the noise of the input, δ_{pq} is the Kronecker delta function, and μ_p and μ_q are the parameter inputs of the measurement outputs. As $\mu_p \approx \mu_q$, k approaches its maximum σ_f^2, meaning that the two points are neighbors and both have a great effect on the regression in that area. Likewise, as $|\mu_p - \mu_q|$ increases,

k becomes smaller and smaller, which means that the two points are far away from each other and the regression computed at one point has a negligible effect on the regression computed at the other. How much the points affect each other depends on the length-scale λ_l^2, which is a hyperparameter that needs to be optimized. The two other hyperparameters of the GPR are σ_f^2 and σ_n^2, which are also determined using an optimization procedure.

The covariance function is computed with all possible combinations of the input μ and collected in a covariance matrix $\mathbf{C}(\boldsymbol{\mathcal{I}})$, which is used to ultimately compute the predictive mean and variance as functions of test inputs $\mu^\star \subset P$ [32]:

$$\tilde{\mathcal{I}}(\mu^\star) = \overline{m} + \tilde{\mathbf{k}}^T \left(\boldsymbol{K}\left(\boldsymbol{\mu}, \boldsymbol{\mu}\right) + \sigma_n^2 \boldsymbol{I}\right)^{-1} \left(\boldsymbol{\mathcal{I}} - \mathbf{1}\overline{m}\right) \tag{2.27}$$

$$\tilde{s}(\mu^\star)^2 = \sigma_f^2 - \tilde{\mathbf{k}}^T \left(\boldsymbol{K}\left(\boldsymbol{\mu}, \boldsymbol{\mu}\right) + \sigma_n^2 \boldsymbol{I}\right)^{-1} \tilde{\mathbf{k}} + \sigma_n^2 \tag{2.28}$$

where $\tilde{\mathbf{k}}^T = [k(\mu_1, \mu^\star),\ k(\mu_2, \mu^\star), \dots,\ k(\mu_{N_c}, \mu^\star)]$ is the vector of correlation functions computed with the N_c observations and the test input $\mu^\star$, and $\overline{m}$ represents the mean of the observations and is a fourth hyperparameter that has to be optimized.

The last step before selecting the parameter according to Eq. (2.23) is to improve the regression with at least N_c^{min} additional observations of the a posteriori error indicator. This is done by searching the parameter space P for the parameter value that has the highest probability of improving the current maximum of the prediction by some target $T(\tilde{\mathcal{I}}^{max})$. A high target enables a global search of the space which helps to find other potential local maxima, one of which could be the global maximum. A low target will cause a local search that refines the current maximum, which could already be the global maximum. Therefore, without a global search, the regression might completely miss out on the global maximum, and without the local search, the regression can only get so close to the global maximum, given that it has already been roughly located. The probability of improving the regression is computed at every test input $\mu^\star$ using the acquisition function $\pi(\mu^\star; T)$:

$$\pi(\mu^\star; T) = \varphi\left(\frac{\tilde{\mathcal{I}}\left(\mu^\star\right) - T\left(\tilde{\mathcal{I}}^{max}\right)}{\tilde{s}(\mu^\star)}\right) \tag{2.29}$$

where φ is the normal cumulative distribution function (cdf) [10, 15, 21].

2.6.3 Sampling Convergence – Error Modeling

After every time the reduced basis has been enriched, it must evaluated whether or not the reduced model is accurate enough or not. When the ROM is judged to be accurate enough, the training is said to have converged. The error indicator $\mathcal{I}$ is what drives the sampling process in the two previous sections, however it is merely an indicator, so there is no knowledge of what the magnitude of the exact error E between the FOM and the ROM actually is. The most reliable way to evaluate the accuracy of the ROM is of course to compare the results of the high-fidelity model and the ROM with many different parameters, but this is of course prohibitively expensive. However, we do have access to FOM solutions at trained parameters, so the ROM can be run with the same sets of parameters, which is inexpensive, and then the exact error can be computed using [10]:

$$E = \frac{\sum_{j=0}^{N_t} \|\boldsymbol{u}_{FOM}(t_j) - \boldsymbol{u}_{POD}(t_j)\|_{\boldsymbol{K}_0}^2}{(N_t + 1)\|\boldsymbol{u}_{FOM}(t_{N_t})\|_{\boldsymbol{K}_0}^2} \tag{2.30}$$

where N_t is the total number of timesteps $t_j = 1, \dots, N_t$, $\boldsymbol{u}_{FOM} \in \mathbb{R}^{N \times N_t}$ is the full time-history of the displacement field from the FOM solution, and $\boldsymbol{u}_{POD} \in \mathbb{R}^{N \times N_t}$ is the full time-history of the displacement field from the POD solution. $\|\boldsymbol{u}\|_{\boldsymbol{K}_0} = \sqrt{\boldsymbol{u}^T \boldsymbol{K}_0 \boldsymbol{u}}$ is the stiffness-projected dot product, where $\boldsymbol{K}_0$ is the initial structure stiffness. This dot product is used along with the displacements to more accurately measure structure specific behaviour [10].

The error indicator $\mathcal{I}$ and the exact error E can thus be computed with all trained parameters before and after training, allowing us to create a mapping between the two, linking the value of $\mathcal{I}$ to E. A log-linear model can be used [21] for this mapping, but it has been shown for multiscale models that this does not work well so instead Gaussian process regression (GPR) can be employed [5, 10] to capture irregular and oscillatory behaviour in the exact error. Both methods are inexpensive, and the computation of a GPR is already investigated in 2.6.2 Gaussian Process Regression so the GPR method will be used in this thesis for modelling the error of the ROM. With a mapping of how the error indicator relates to the exact error, a convergence criterion that takes a user-defined error tolerance can be utilized to determine when the offline training phase should be stopped.

3 PARAMETER SAMPLING METHODS

The focus of the remainder of the thesis is how the parameter samples are chosen to construct the reduced basis, and how that affects the efficiency and accuracy of the ROM. However, it is important to ensure that the actual enrichment of the basis is efficient, so the first section of this chapter details how the basis is assembled and updated. The two parameter sampling methods using the SPS method or GPR method are then explained in detail before the sampling convergence is discussed. Note that GPR was also discussed in 2.6.3 Sampling Convergence – Error Modeling as a way to create a mapping between the error indicator and the exact error, so to avoid any confusion, two additional abbreviations are introduced to distinguish what the GPR is used for: Gaussian process regression for prediction (GPRP), and Gaussian process regression for convergence (GPRC).

To implement these methods, the simply support beam model shown in Figure 3.1 will be used. It is a rectangular beam of a homogeneous, elastoplastic material. The load is applied along the top of the beam and the load location is defined by the parameter μ and the span of the beam is 20 mm long. However, the actual loading span is limited to $5 \leq \mu \leq 15$, and this span is referred to as the parameter space P. The beam will only be subjected to one point load at a time, and is applied in the numerical analysis using arc-length control.

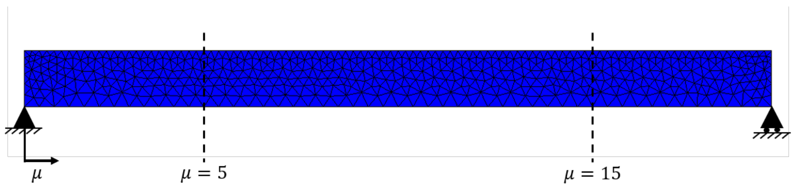
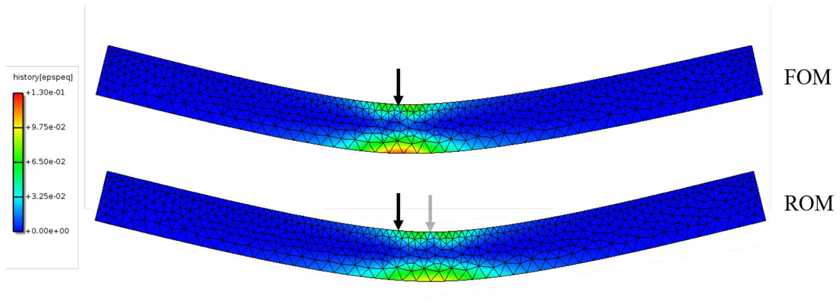
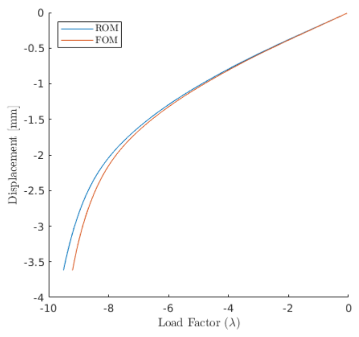


Figure 3.1: Simply supported beam model

The parameter space is in theory infinitely large as the load can be placed anywhere, however it is more simple to exhaust this parameter space sufficiently compared to those of RVE models (as outlined in 2.5.1 Constrained Random Sampling for RVE Modeling), because this parameter space can be discretized easily with grids. This is the reason why using the SPS method as a sampling method has the potential to be efficient and accurate. However, we do not know how fine the grid should be in order to construct an accurate ROM, which is why the SPS $\widehat{P}_{N_p}$ becomes incrementally finer. Figure 3.2 illustrates the effect of having a deficient ROM. The top beam shows the equivalent plastic strains in the FOM when a load is applied at $\mu = 9$. The bottom beam shows the same type of strains, but of a ROM in an online analysis with the load at the same location ($\mu = 9$), for which the reduced basis was assembled with only one load case at $\mu = 10$ during the offline phase. Both analysis are run with the same amount of time steps to a displacement of 3.62 mm, as seen in Figure 3.2b, and the black arrow for the ROM indicate the load applied in the online analysis, and the gray arrow is the one load case used to construct the reduced basis. It is clear that the ROM is severely deficient when the online load is only 5% away from the offline load as the error in the final load factor λ when comparing the FOM with the ROM is 3.3%. Furthermore, when observing the distribution of the equivalent plastic strains, the ROM experiences the same distribution as the FOM solution that was used in the offline phase to construct the basis. This occurs because the ROM has no information about how strains can be distributed differently when the loads are applied elsewhere. This highlights the importance of constructing a reduced basis that has sampled the parameter space exhaustively enough to obtain accurate approximations in the online phase.



(a) ROM with deficient basis and corresponding FOM



(b) Force-displacement graph

Figure 3.2: Illustration of the effect of deficient basis

The reduced model is constructed by iteratively enriching the reduced basis through greedy sampling of the parameter space P. The high-level steps of the offline training phase are summarized in Algorithm 3.1.

Algorithm 3.1: Offline training phase procedure

```
1  Select initial load location
2  Set $N_{it} = 0$
3  Initialize reduced basis $\mathbf{\Phi}_0$
4  while not converged do
5      Update $N_{it} = N_{it} + 1$
6      Greedily sample the parameter space $P$ using SPS or GPRP
7      Run the full-order model
8      Enrich reduced basis $\mathbf{\Phi}_{N_{it}}$
9      Evaluate accuracy of the ROM using GPRC
10     if converged then
11         return Reduced basis $\mathbf{\Phi}$
12     end
13 end
```

3.1 Assembling the Reduced Basis

To obtain the displacement snapshots $\mathbf{X}$, a full-order model (FOM) is run with the parameter value determined by one of the greedy sampling methods (SPS or GPRP). The snapshots are obtained from the FOM solution in each time step and collected in a *local* snapshot matrix $\mathbf{X}_{N_{it}}\ in\mathbb{R}^{N\times N_t}$, which is subsequently decomposed into an elastic part ($\mathbf{X}^e_{N_{it}}$) and an inelastic part ($\mathbf{X}^i_{N_{it}}$) as proposed by Hernández et al. [11]:

$$\mathbf{X}_{N_{it}} = \left[\mathbf{X}^e_{N_{it}}, \mathbf{X}^i_{N_{it}}\right] \tag{3.1}$$

For the sake of brevity and readability, the subscript N_{it} will be left out from now on for the local matrices, which are generated anew in every iteration. This decomposition ensures that elastic load cases are always exactly reproduced, providing a baseline for the accuracy of the ROM. The next step is to perform POD-compression using SVD on both snapshot matrices separately:

$$\mathbf{X}^e = (\mathbf{U\Sigma Z}^T)^e \in \mathbb{R}^{N\times N_e} \qquad \mathbf{X}^i = (\mathbf{U\Sigma Z}^T)^i \in \mathbb{R}^{N\times N_i} \tag{3.2}$$

where N_e is the number of elastic snapshots, and N_i is the number of inelastic snapshots. The resulting matrices are then truncated to $k_{N_{it}}$ modes that is either a fixed number or determined by the energy criterion (Eq. (2.16)). The elastic snapshots will be compressed into one mode ($k^e = 1$) because all of the linear elastic behaviour can easily be captured in just one mode. For the inelastic behaviour, however, it is not as simple. Depending on where the load is applied, the inelastic behaviour may vary considerably. In some cases, there will be significant localized plasticity, but in other cases it will be more spread out. Therefore, the number of inelastic modes $k^i_{N_{it}}$ must be determined separately for each case, and this will be done using the energy criterion such that a certain fraction $0 \leq \varepsilon^i \leq 1$ of the energy is retained in the compressed matrix:

$$\frac{\sum_{j=1}^{k^i_{N_{it}}} \Sigma^i_{jj}}{\sum_{j=1}^{N_i} \Sigma^i_{jj}} \geq \varepsilon^i \tag{3.3}$$

When the number of modes to truncate is set, the compression is approximated as:

$$\mathbf{X}^e \approx \left(\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{Z}}^T\right)^e \in \mathbb{R}^{N\times N_e} \qquad \mathbf{X}^i \approx \left(\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{Z}}^T\right)^i \in \mathbb{R}^{N\times N_i} \tag{3.4}$$

Using the output from the truncated SVD procedure, two local *surrogate* snapshot matrices are constructed [21]:

$$\widetilde{\mathbf{X}}^e = \left(\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\right)^e \in \mathbb{R}^{N\times k^e} \qquad \widetilde{\mathbf{X}}^i = \left(\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\right)^i \in \mathbb{R}^{N\times k^i_{N_{it}}} \tag{3.5}$$

where $\widetilde{\mathbf{U}}^e$ and $\widetilde{\mathbf{U}}^i$ have the same dimensions as $\mathbf{X}^e$ and $\mathbf{X}^i$, respectively. This construction is also illustrated in Figure 3.3. Now we have obtained local surrogate snapshot matrices that are of significantly smaller dimension than their corresponding full-order matrices. For the very first iteration ($N_{it} = 0$), these two matrices are combined to make the *global* surrogate snapshot matrix $\widetilde{\mathbf{X}}$:

$$\widetilde{\mathbf{X}}^g_0 = [\widetilde{\mathbf{X}}^e, \widetilde{\mathbf{X}}^i] \in \mathbb{R}^{N\times(k^e + k^i_{N_{it}})} \tag{3.6}$$

For every iteration after the first one ($N_{it} > 0$), the elastic and inelastic surrogate snapshot matrices will be combined with the previous global surrogate snapshot matrix $\widetilde{\mathbf{X}}^g_{N_{it}-1}$:

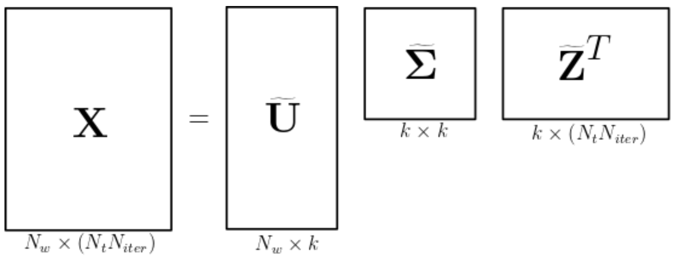
$$\widetilde{\mathbf{X}}^g_{N_{it}} = [\widetilde{\mathbf{X}}^g_{N_{it}-1}, \widetilde{\mathbf{X}}^e, \widetilde{\mathbf{X}}^i] \in \mathbb{R}^{N\times\left(\sum_{j=1}^{N_{it}}(k^i_j + k^e_j)\right)} \tag{3.7}$$

Then, the SVD is applied to $\widetilde{\mathbf{X}}^g_{N_{it}}$, and the resulting matrices are truncated in the same way as with the inelastic snapshots (Eq. (3.3)), but now using a different global energy tolerance ε^g:

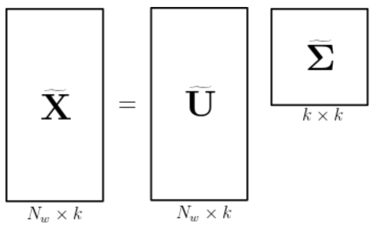
$$\widetilde{\mathbf{X}}^g \approx \left(\widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{Z}}^T\right)^g_{N_{it}} \tag{3.8}$$

Finally, we obtain:

$$\mathbf{\Phi}_{N_{it}} = \widetilde{\mathbf{U}}^g_{N_{it}} \tag{3.9}$$



(a) Truncated SVD



(b) Surrogate matrix construction

Figure 3.3: Truncated SVD and surrogate construction

Computational Complexity
When using the global snapshot matrix X^g that contains all displacement snapshots collected in all iterations to compute the reduced basis $\mathbf{\Phi}_{N_{it}}$, the computational complexity is [21]:

$$O\left(\sum_{i=1}^{N_{it}} N(N_t i)^2\right) = O(NN_t^2 N_{it}^3) \tag{3.10}$$

For the global surrogate snapshot matrix, however, the complexity is significantly reduced. If we for the sake of simplicity consider $k^e + k^i_{N_{it}}$ to constantly be k, then the computational complexity of computing the basis $\mathbf{\Phi}_{N_{it}}$ with this approach is for the first N_{it} iterations:

$$O\left(\sum_{i=1}^{N_{it}} \left(NN_t^2 + N\left(ik\right)^2\right)\right) = O\left(NN_{it}\left(N_t^2 + k^2 N_{it}^2\right)\right) \tag{3.11}$$

which shows that when $k \ll N_t$, the SVD performed on the global surrogate snapshot matrix involves matrices of significantly lower order, and thus drastically reducing the computation time. However, this improved efficiency comes at the cost of all the information in the basis vectors that are dropped during the truncated SVD, which reduces the accuracy of the reduced model. Therefore, it is important to take care when choosing the energy tolerances to ensure that enough information is kept.

The entire procedure of updating the reduced basis as described in this section is summarized in Algorithm 3.2:

Algorithm 3.2: POD-compression procedure for $N_{it} > 0$

```
Input  : Elastic snapshots $\mathbf{X}^{e}_{N_{it}}$
         Inelastic snapshots $\mathbf{X}^{i}_{N_{it}}$
         Global surrogate snapshots $\widetilde{\mathbf{X}}^{g}_{N_{it}-1}$
         Number of desired elastic modes $k^{e}$
         Inelastic energy tolerance $\varepsilon^{i}$
         Global energy tolerance $\varepsilon^{g}$
Output : Global surrogate snapshots $\widetilde{\mathbf{X}}^{g}_{N_{it}}$
         Reduced basis $\mathbf{\Phi}_{N_{it}}$
1 Compress $\mathbf{X}^{e}_{N_{it}}$ using SVD to $k^{e}$ modes
2 Compress $\mathbf{X}^{i}_{N_{it}}$ using SVD to $k^{i}_{N_{it}}$ modes based on energy tolerance $\varepsilon^{i}$
3 Reconstruct elastic surrogate snaphots $\widetilde{\mathbf{X}}^{e}_{N_{it}}$
4 Reconstruct inelastic surrogate snapshots $\widetilde{\mathbf{X}}^{i}_{N_{it}}$
5 Update $\widetilde{\mathbf{X}}^{g}_{N_{it}} = [\widetilde{\mathbf{X}}^{g}_{N_{it}-1}, \widetilde{\mathbf{X}}^{e}, \widetilde{\mathbf{X}}^{i}]$
6 Compress $\widetilde{\mathbf{X}}^{g}_{N_{it}}$ using SVD to $k^{g}_{N_{it}}$ modes based on energy tolerance $\varepsilon^{g}$ to obtain $\mathbf{\Phi}_{N_{it}}$
7 return $\widetilde{\mathbf{X}}^{g}_{N_{it}}, \mathbf{\Phi}_{N_{it}}$
```

3.2 Greedy Sampling

3.2.1 Surrogate Parameter Space

The parameter space P can in the case of the RVE (2.1 High-Fidelity Models) be infinitely dimensional, and in the case of the beam model there is an infinite number of points where the load can be placed. Constructing surrogate parameter spaces (SPS) $\widehat{P}_{N_p}$ is a way to explore the full-order parameter space in a structured and efficient way, where N_p denotes the SPS iteration. For the beam model, this can be done very simply by discretizing P into incrementally finer grids of uniform spacing, where each individual grid is $\widehat{P}_{N_p}$, as shown in Figure 3.4. With the shown grid, the parameter space should intuitively be searched in the most efficient way, because starting from $\widehat{P}_2$, the grid fills the gaps in the previous SPSs $\widehat{P}_0$ and $\widehat{P}_1$, and it continues like that for each SPS.

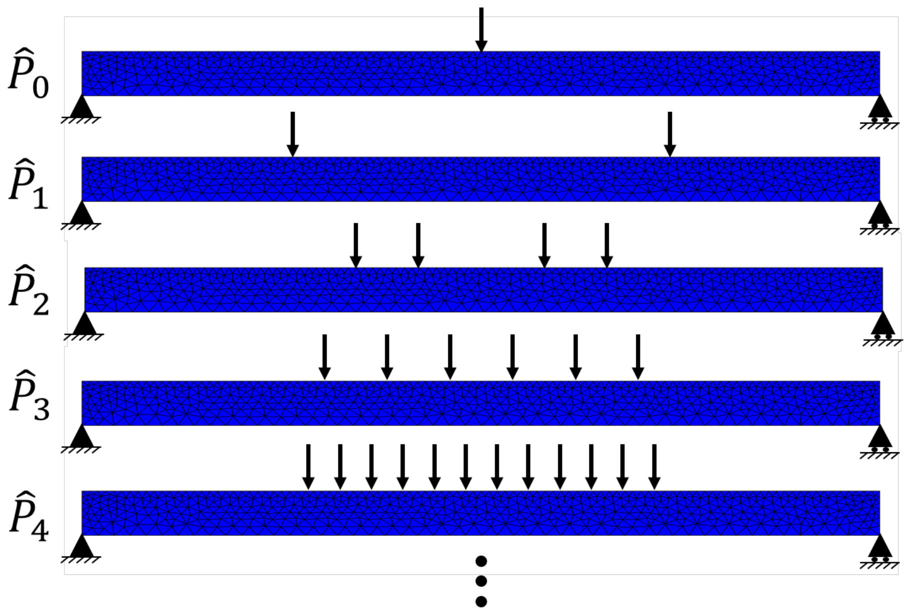


Figure 3.4: SPSs for the simply support beam model

The following explanation of the procedure to sample the SPSs is summarized in Algorithm 3.3. For the first SPS ($\widehat{P}_0$), the load is simply placed at the only point, and the training procedure moves to the next SPS, which has two points: one on each end. What is desired in a greedy algorithm in every single iteration is to train the point which has the highest error, so when moving to a new SPS, the current reduced model is run using all the surrogate parameters $\widehat{\mu}$, and the error indicator $\mathcal{I}$ is computed at every single point in the grid using the formula:

$$\mathcal{I}(\widehat{\mu};\mathbf{\Phi}) = \sqrt{\frac{\sum_{k=0}^{k=N_t} \left\|\mathbf{r}(t_k,\ \widehat{\mu};\ \mathbf{\Phi})\right\|_2^2}{N_t + 1}} \tag{3.12}$$

where N_t is the total number of timesteps $t_k = 1, \dots, N$, $\|\cdot\|_2$ is the L2-norm, and $\mathbf{r}$ is the full-order residual at timestep t_k given by:

$$\mathbf{r} = \mathbf{f}^{\Omega} - \mathbf{f}^{\Gamma} \tag{3.13}$$

The location with the maximum error indicator will be chosen as the parameter for the next load case:

$$\mu_{N_{it}} = \underset{\widehat{\mu}\in\widehat{P}_{N_p}}{\arg\max}\,\mathcal{I}(\widehat{\mu};\mathbf{\Phi}) \tag{3.14}$$

and this maximum error indicator is set as $\mathcal{I}^{p}_{N_{it}}$, where p indicates that it is from before updating the basis in N_{it}. Using the parameter $\mu_{N_{it}}$, the full-order model (FOM) solution is computed to collect snapshots to enrich the reduced basis (RB) matrix further. However, before the basis is updated, the exact error $E^{p}_{N_{it}}$ of the ROM solution (with $\mathbf{\Phi}_{N_{it}-1}$) is computed at this point using the displacement field time-history that was just obtained from the FOM solution. This error will be used along with the error indicator $\mathcal{I}^{p}_{N_{it}}$ to model the error for convergence purposes, which will be explained in detail in 3.3 Sampling Convergence – Error Modeling. The next step is then to enrich the basis $\mathbf{\Phi}_{N_{it}}$, and compute the ROM solution at the same parameter $\mu_{N_{it}}$ so that the error indicator $\mathcal{I}^{a}_{N_{it}}$ and exact error $E^{a}_{N_{it}}$ can be computed again, both of which are now significantly lower than before updating the basis. However, neither of them will be zero despite the fact that the ROM now contains a load case at the exact same load location, which is because some information is inevitably lost in the compression stage due to the truncation.

Now that error indicators and exact errors are computed before and after enriching the basis, these results are stored in an error pool $\mathcal{E}$, which will be fed into the convergence error model to determine whether or not the reduced model is sufficiently accurate to describe any load applied on the beam. If the training has not converged yet, the procedure starts over again with computing the error indicator at every remaining untrained point in SPS $\widehat{P}$. The SPS containing untrained points is denoted in Algorithm 3.3 as $\widehat{P}^{*}_{N_p}$, where the $*$ signifies that the SPS has been decremented and does not contain already trained points. When the SPS is completely exhausted, we move to the next SPS $\widehat{P}_{N_p+1}$.

The efficiency of this method depends on the inelastic (ε^{i}) and global (ε^{g}) energy tolerances. A high tolerance means that more information is retained in the SVDs, but it also means that each ROM will be more expensive to run, so energy tolerances have to be chosen such as to achieve the efficiency desired with respect to an accepted loss of accuracy.

Algorithm 3.3: Surrogate Parameter Space Method Procedure

```
Input   : Parameter space $P$
          Exact error tolerance $E^{max}$
          Inelastic energy tolerance $\varepsilon^i$
          Global energy tolerance $\varepsilon^g$
Output  : Reduced basis (RB) $\mathbf{\Phi} \in \mathbb{R}^{N \times n}$
1  Go to SPS $\widehat{P}_0$
2  Set $N_{it} = 0$
3  Choose initial parameter $\mu_0 \in \widehat{P}_0$
4  Compute the FOM solution $\mathbf{u}(t, \mu_0)$
5  Construct the initial RB $\mathbf{\Phi}_0$ using POD-compression with energy tolerances $\varepsilon^i$ and $\varepsilon^g$
6  Set $N_p = 1$
7  Go to SPS $\widehat{P}_{N_p}$ and set $\widehat{P}^*_{N_p} = \widehat{P}_{N_p}$
8  Initialize error pool $\mathcal{E} = \emptyset$
9  while not converged do
10     Update $N_{it} = N_{it} + 1$
11     if $size(\widehat{P}^*_{N_p}) \geq 2$ then
12         Compute ROM solutions $\mathbf{u}_r(t, \widehat{\mu}; \mathbf{\Phi}_{N_{it}-1})$ for all remaining untrained points $\widehat{\mu} \in \widehat{P}^*_{N_p}$
13         Compute the associated error indicators $\mathcal{I}(\widehat{\mu})$ for all $\widehat{\mu} \in \widehat{P}^*_{N_p}$
14         Select $\mu_{N_{it}} = \arg\max_{\widehat{\mu} \in \widehat{P}^*_{N_p}} \mathcal{I}(\widehat{\mu})$
15     else
16         Select $\mu_{N_{it}}$ as the last untrained point $\widehat{\mu} \in \widehat{P}^*_{N_p}$
17         Compute ROM solution $\mathbf{u}_r(t, \widehat{\mu}; \mathbf{\Phi}_{N_{it}-1})$
18     end
19     Compute FOM solution $\mathbf{u}(t, \mu_{N_{it}})$
20     Set $\mathcal{I}^p_{N_{it}} = \mathcal{I}(\mu_{N_{it}})$ and compute the exact error $E^p_{N_{it}}$
21     Update the RB to $\mathbf{\Phi}_{N_{it}}$ using POD-compression with energy tolerances $\varepsilon^i$ and $\varepsilon^g$
22     Compute ROM solution $\mathbf{u}_r(t, \mu_{N_{it}}; \mathbf{\Phi}_{N_{it}})$
23     Compute the associated error indicator $\mathcal{I}^a_{N_{it}}$ and exact error $E^a_{N_{it}}$
24     Update $\mathcal{E} = \mathcal{E} \cup \left(\mathcal{I}^p_{N_{it}}, E^p_{N_{it}}\right) \cup \left(\mathcal{I}^a_{N_{it}}, E^a_{N_{it}}\right)$
25     Compute the GPRC error model $\widetilde{E}\left(\mathcal{I}^\star\right)$ and associated error bound $\tilde{s}_E\left(\mathcal{I}^\star\right)$ using the error
         pool $\mathcal{E}$
26     if converged then
27         return $\mathbf{\Phi}$
28     else if $size(\widehat{P}^*_{N_p}) \geq 2$ then
29         Remove $\mu_{N_{it}}$ from $\widehat{P}^*_{N_p}$
30     else
31         Update $N_p = N_p + 1$
32         Go to SPS $\widehat{P}_{N_p}$ and set $\widehat{P}^*_{N_p} = \widehat{P}_{N_p}$
33     end
34 end
```

3.2.2 Gaussian Process Regression for Prediction

For the SPS method to be efficient, the parameter space P must be simple enough to be split into sets that enable exploration of the high-dimensional parameter space in an efficient manner. In the case of the beam it is very simple, as it can be subdivided into 1-D grids, but even with such a division, there is no a priori knowledge of how fine the grid should be, which is why this thesis looked into incrementally finer grids. However, as outlined in 2.5.1 Constrained Random Sampling for RVE Modeling, the parameter space can be infinite-dimensional, and thus the same method of subdividing the space cannot be utilized, so more sophisticated methods of sampling the space should be implemented, which is where the Gaussian process regression for prediction (GPRP) comes in. This form of supervised Bayesian machine learning infers a probability distribution of the prediction for any parameter μ in the parameter space.

The following explanation of the procedure to sample the parameter space using GPRP is summarized in Algorithm 3.5. The offline training phase is initiated at the zeroth training iteration (Algorithm 3.1) by running a full-order model (FOM) with the load at a random point. The displacement field history is collected in a snapshot matrix $\mathbf{X}_0$ and compressed into $\mathbf{\Phi}_0$. With the initial reduced basis ready, the first step towards building the GPRP can begin, which is running the reduced-order model (ROM) at N_c^0 initial random candidate parameters $\mathcal{C} = \left\{\mu_1^c, \dots, \mu_{N_c^0}^c\right\} \subset P$, and then compute the corresponding error indicators $\mathcal{I}^c(\mathcal{C})$:

$$\mathcal{I}_i^c(\mu_i^c; \mathbf{\Phi}) = \sqrt{\frac{\sum_{k=0}^{k=N_t} \left\|\mathbf{r}(t_k,\ \mu_i^c;\ \mathbf{\Phi})\right\|_2^2}{N_t + 1}} \tag{3.15}$$

With the initial observations of the error indicator $\mathcal{I}^c(\mathcal{C})$, the squared exponential covariance function can be computed:

$$\operatorname{cov}\left(\mathcal{I}_p^c,\ \mathcal{I}_q^c\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{(\mu_p^c - \mu_q^c)^2}{\lambda_l^2}\right] \tag{3.16}$$

$$\operatorname{cov}\left(\mathcal{I}_p^c,\ \mathcal{I}_q^c\right) = k(\mu_p^c,\ \mu_q^c) \tag{3.17}$$

or collected in matrix form:

$$\mathbf{C}\left(\mathcal{I}^c(\boldsymbol{\mu}^c)\right) = \sigma_f^2 \boldsymbol{R}\left(\boldsymbol{\mu}^c,\ \boldsymbol{\mu}^c\right) = \boldsymbol{K}\left(\boldsymbol{\mu}^c,\ \boldsymbol{\mu}^c\right) \tag{3.18}$$

where $\boldsymbol{\mu}^c$ is a vector containing all the candidate parameters in $\mathcal{C}$. Note that the observation noise variance σ_n^2 that was introduced in Eq. (2.24) is left out. This is because the GPR is used to predict values of the error indicator along the beam, and there is no uncertainty in the observations made at the candidate parameters, so $\sigma_n^2 = 0$. For GPRP, the covariance function will be referred to as $\boldsymbol{K}\left(\boldsymbol{\mu},\ \boldsymbol{\mu}\right)$ to more easily distinguish it from the covariance function $\mathbf{C}$ for GPRC that does contain the noise variance. The observations of the error indicator are described by the likelihood, or probability density, given the training input parameters [24]

$$\mathcal{I}(\mu) \sim \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{K}|}} \exp\left(-\frac{1}{2}(\mathcal{I}^c - \mathbf{1}\overline{m})^T \boldsymbol{K}^{-1} (\mathcal{I}^c - \mathbf{1}\overline{m})\right) \tag{3.19}$$

which can be rewritten more simply as:

$$\mathcal{I}(\mu) \sim \mathcal{N}\left(\overline{\boldsymbol{m}}(\mu),\ \boldsymbol{K}\left(\boldsymbol{\mu}^c,\ \boldsymbol{\mu}^c\right)\right) \tag{3.20}$$

The hyperparameters $\overline{m}$, σ_f^2, and λ_l^2 are optimized by maximizing the log-likelihood function:

$$\log(\mathcal{L}) = -\frac{N_c}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{R}|) - \frac{1}{2}\frac{(\mathcal{I}^c - \mathbf{1}\overline{m})^T \boldsymbol{R}^{-1} (\mathcal{I}^c - \mathbf{1}\overline{m})}{\sigma_f^2} \tag{3.21}$$

The maximum is found by setting the derivative equal to zero and solving for the different hyperparameters. In the case without noise, $\overline{m}$ and σ_f^2 can be solved analytically:

$$\overline{m} = \frac{\mathbf{1}^T \boldsymbol{R}^{-1} \boldsymbol{\mathcal{J}}^c}{\mathbf{1}^t \boldsymbol{R}^{-1} \mathbf{1}} \tag{3.22}$$

$$\sigma_f^2 = \frac{(\boldsymbol{\mathcal{J}}^c - \mathbf{1}\overline{m})^T \boldsymbol{R}^{-1} (\boldsymbol{\mathcal{J}}^c - \mathbf{1}\overline{m})}{N_c} \tag{3.23}$$

After substituting Eq. (3.22) and Eq. (3.23) into Eq. (3.21) and ignoring constant terms, we are only left with one term of the log-likelihood to maximize in order to determine the last hyperparameter λ_l^2:

$$\lambda_l^2 = \arg\max \left(-\frac{1}{2} \log \left| \boldsymbol{R} \right| \right) \tag{3.24}$$

Since there is only one variable to optimize for, it is sufficient to perform a simple grid search to find the optimal value. The next step is to include a large set of test inputs $\mu^\star \subset P$, and along with the training inputs, joint distribution of the two can be described as:

$$\begin{bmatrix} \mathcal{J} \\ \widetilde{\mathcal{J}} \end{bmatrix} \sim \mathcal{N}\left(\overline{\boldsymbol{m}}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{\mu}^c, \boldsymbol{\mu}^c) & \boldsymbol{K}(\boldsymbol{\mu}^c, \widetilde{\boldsymbol{\mu}}) \\ \boldsymbol{K}(\widetilde{\boldsymbol{\mu}}, \boldsymbol{\mu}^c) & \boldsymbol{K}(\widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\mu}}) \end{bmatrix} \right) \tag{3.25}$$

From this distribution, the predictive mean and associated variance as functions of test inputs are:

$$\widetilde{\mathcal{J}}(\mu^\star) = \overline{m} + \widetilde{\mathbf{k}}^T \boldsymbol{K}\left(\boldsymbol{\mu}^c, \boldsymbol{\mu}^c\right)^{-1} \left(\boldsymbol{\mathcal{J}}^c - \mathbf{1}\overline{m}\right) \tag{3.26}$$

$$\tilde{s}_{\mathcal{J}}^2(\mu^\star) = \sigma_f^2 - \widetilde{\mathbf{k}}^T \boldsymbol{K}\left(\boldsymbol{\mu}^c, \boldsymbol{\mu}^c\right)^{-1} \widetilde{\mathbf{k}} \tag{3.27}$$

where $\widetilde{\mathbf{k}}^T = [k(\mu_1^c, \mu^\star), k(\mu_2^c, \mu^\star), \ldots, k(\mu_{N_c^0}^c, \mu^\star)]$. It is also common to use the zero mean regression:

$$\widetilde{\mathcal{J}}(\mu^\star) = \widetilde{\mathbf{k}}^T \boldsymbol{K}\left(\boldsymbol{\mu}^c, \boldsymbol{\mu}^c\right)^{-1} \boldsymbol{\mathcal{J}}^c \tag{3.28}$$

but is not used in this thesis in order to follow the procedures in Goury et al. [10] and Paul-Dubois-Taine and Amsallem [21]. In general, N_c^0 is low and will lead to poor a poor regression, so another step is required in order to improve it further. During the following improvement procedure, the ROM will be run a number of times to gather more observations that will improve the GPR. These locations where these observations are obtained is chosen by searching the parameter space P for the location that has the highest probability of improving the current maximum of the GPR $\widetilde{\mathcal{J}}^{max}$ by a certain target value $T(\widetilde{\mathcal{J}}^{max})$. At every test input $\mu^\star$, the prediction $\widetilde{\mathcal{J}}$ has a normal distribution , so the probability of improving the prediction is computed using the acquisition function $\pi(\mu^\star; T)$:

$$\pi(\mu^\star; T) = \varphi\left(\frac{\widetilde{\mathcal{J}}\left(\mu^\star\right) - T\left(\widetilde{\mathcal{J}}^{max}\right)}{\tilde{s}_{\mathcal{J}}(\mu^\star)} \right) \tag{3.29}$$

where φ is the normal cumulative distribution function. Several target values T_i are calculated using the expression:

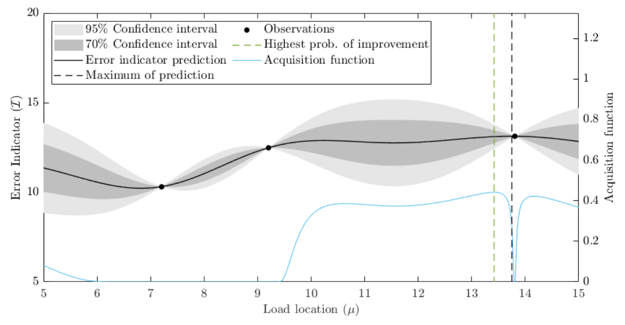
$$T_i = \widetilde{\mathcal{J}}^{max} + \tau_i \left(\widetilde{\mathcal{J}}^{max} - \widetilde{\mathcal{J}}^{min} \right) \tag{3.30}$$

where $\boldsymbol{\tau} = \left\{\tau_1, \ldots, \tau_{N_T}\right\}$ are target factors that determine how global or local the search should be. A low target leads to a local search for a higher maximum, as shown in Figure 3.5a, and a high target enables a global search (Figure 3.5b. In order to both refine the current local maximum and to find potentially other local maxima, both a local and a global search is performed by choosing many values of the target factor τ_i. As shown in Figure 3.6, a low target seeks to refine the current maximum, and when the target gets high enough, the search skips to a point far away that could potentially be the global maximum.

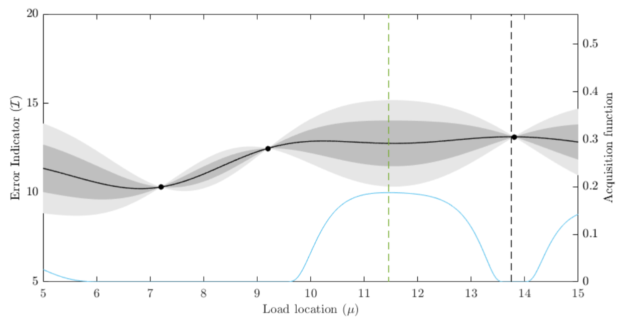
There are two clusters of points that could have the global maximum, and in each cluster the locations associated with the highest targets are chosen to improve the regression. This improvement is carried out until there are at least N_c^{min} points to compute the regression, or when there is a high certainty that the current maximum is the global one. We can be certain of this when the following condition is satisfied:

$$\left(\tilde{\mathcal{I}}^{max} - \tilde{\mathcal{I}}^{min}\right)\beta > \max\left(\tilde{\mathcal{I}}(\mu^\star) + \alpha_{\mathcal{I}}\tilde{s}_{\mathcal{I}}(\mu^\star)\right) \tag{3.31}$$

where $\alpha_{\mathcal{I}}$ is the z-score, or the number of standard deviations from the mean, and β is a factor that determines how much above the current maximum the mean plus $\alpha_{\mathcal{I}}$ standard deviations must be to continue the GPRP improvement. For few observations in $\mathfrak{I}^c$, the computation of the GPRP itself is negligible (which will always be the case in this thesis) when not considering the time taken to run the ROM. Therefore, N_c^0 should be as low as possible so that a poor but reasonable regression can be computed and then subsequently improved. It is better to as soon as possible cheaply compute points that are likely to improve the regression using the procedure described above and in Algorithm 3.4. The number of additional improvements is a parameter that greatly affects the efficiency of the offline training phase.



(a) Low target, 1%



(b) High target, 25%

Figure 3.5: GPRP improvement with only low target and only high target

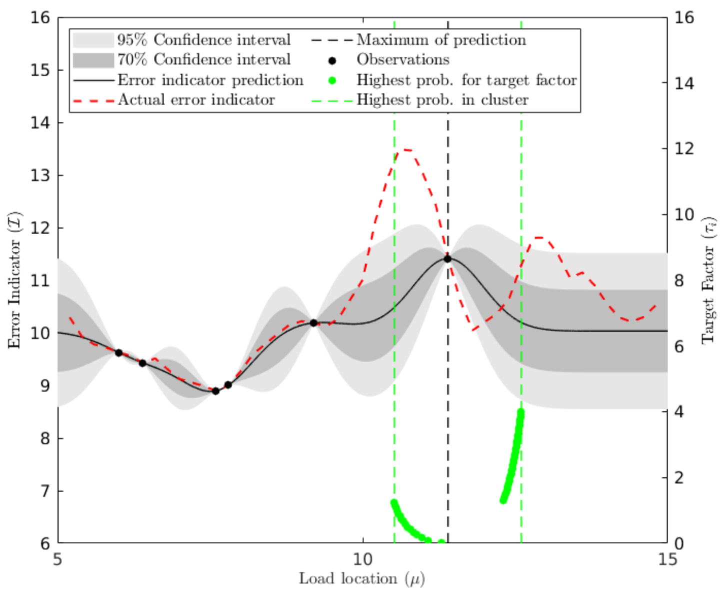


Figure 3.6: GPRP improvement with many targets

When the improvement is complete, the training location for the current iteration N_{it} is the one that yields the maximum error indicator:

$$\mu_{N_{it}} = \underset{\mu^\star \subset P}{\arg\max} \widetilde{\mathcal{I}}(\mu^\star) \tag{3.32}$$

and this error indicator is set as $\mathcal{I}^p_{N_{it}}$. Using the parameter $\mu_{N_{it}}$, the FOM solution is computed to collect snapshots to enrich the reduced basis further. However, before the basis is updated, the exact error $E^p_{N_{it}}$ of the ROM (with $\mathbf{\Phi}_{N_{it}-1}$) is computed at this point using the displacement field time-history that was just obtained from the FOM solution. After the basis is updated, the ROM solution is computed again, and the error indicator $\mathcal{I}^a_{N_{it}}$ and the exact error $E^a_{N_{it}}$ are also computed. All these error indicators and exact errors that are computed in every iteration are stored in an error pool $\mathcal{E}$, which is used to build the error model that evaluates whether or not the the ROM is sufficiently accurate. If the training has not converged yet, the procedure starts over again with computing a new GPRP to find the next training location.

Like with the SPS method, the efficiency of the GPR method also depends on the inelastic (ε^i) and global (ε^g) energy tolerances, but there is a very important third parameter that has a great effect as well, which is the minimum number of improvements made to the GPRP N_c^{min}. A high number ensures that the GPRP almost always finds the global maximum, but that requires running the ROM many times, which can be costly. A low number poses the risk of choosing the next training location at a suboptimal location, which can cause the offline phase to require more training iterations in order to construct a sufficiently accurate ROM. So all three input parameters have to be chosen such as to achieve the efficiency desired with respect to an accepted loss of accuracy.

Algorithm 3.4: Computing points with highest probability of improving the GPR

```
Input  : Parameter space $P$
         GPRP predictive mean $\widetilde{\mathcal{I}}(\mu^\star)$
         GPRP error $\tilde{s}_{\mathcal{I}}(\mu^\star)$
Output : Candidate parameters $C^m$
1  Define $N_T$ target factors $\boldsymbol{\tau} = \left(\tau_1, \dots, \tau_{N_T}\right)$
2  for $i = 1, \dots, N_T$ do
3      Compute $T_i = \widetilde{\mathcal{I}}^{max} + \tau_i\left(\widetilde{\mathcal{I}}^{max} - \widetilde{\mathcal{I}}^{min}\right)$
4      Find parameter $\mu_i^c$ with maximum probability of improving the regression by factor $\tau_i$:
           $$\mu_i^c = \underset{\mu^\star \subset P}{\arg\max}\left(\varphi\left(\frac{\widetilde{\mathcal{I}}(\mu^\star) - T\left(\widetilde{\mathcal{I}}^{max}\right)}{\tilde{s}_{\mathcal{I}}(\mu^\star)}\right)\right)$$
5  end
6  Determine $m$ parameter clusters based on spatial proximity
7  for $j = 1, \dots, m$ do
8      Set $\mu_j^c$ as the parameter in each cluster associated with the highest target
9  end
10 return $C^m = \left\{\mu_1^c, \dots, \mu_m^c\right\} \subset P$
```

Algorithm 3.5: Gaussian Process Regression Method Procedure

```
Input  : Parameter space $P$
         Exact error tolerance $E^{max}$
         Inelastic energy tolerance $\varepsilon^i$
         Global energy tolerance $\varepsilon^g$
         Initial number of error indicator observations $N_c^0$
         Minimum number of additional error indicator observations $N_c^{add}$
Output : Reduced basis (RB) $\mathbf{\Phi} \in \mathbb{R}^{N \times n}$
1  Set $N_{it} = 0$
2  Choose a random initial parameter $\mu_0 \in P$
3  Compute the FOM solution $\mathbf{u}(t, \mu_0)$
4  Construct the initial RB $\mathbf{\Phi}_0$ using POD-compression with energy tolerances $\varepsilon^i$ and $\varepsilon^g$
5  Initialize error pool $\mathcal{E} = \emptyset$
6  while not converged do
7      Choose $N_c^0$ initial random candidate parameters $\mathcal{C} = \left\{\mu_1^c, \dots, \mu_{N_c^0}^c\right\} \subset P$
8      Initialize set of candidate error indicators $\mathcal{I}^c = \emptyset$
9      for $i = 1, \dots, N_c^0$ do
10         Compute the ROM solutions $\mathbf{u}_r(t, \mu_i^c; \mathbf{\Phi}_{N_{it}-1})$
11         Compute the corresponding error indicator $\mathcal{I}_i^c(\mu_i^c)$
12         Update $\mathcal{I}^c = \mathcal{I}^c \cup \mathcal{I}_i^c(\mu_i^c)$
13     end
14     Set $N_c = 0$ and $N_C^{min} = N_c^{add} + N_c^0$
15     while $N_c \leq N_c^{min}$ do
16         Set $m = 0$
17         Compute the GPRP predictive mean $\widetilde{\mathcal{I}}(\mu^\star)$ and associated error $\tilde{s}_{\mathcal{I}}(\mu^\star)$ using $\mathcal{C}$ and $\mathcal{I}^c$
18         if $\left(\widetilde{\mathcal{I}}^{max} - \widetilde{\mathcal{I}}^{min}\right)\beta > \max\left(\widetilde{\mathcal{I}}(\mu^\star) + \alpha_{\mathcal{I}}\tilde{s}_{\mathcal{I}}(\mu^\star)\right)$ then
19             break
20         end
21         Select $m$ candidate parameters $\mathcal{C}^m = \left\{\mu_1^c, \dots, \mu_m^c\right\} \subset P$ according to Algorithm 3.4
22         Update $N_c = N_c + m$
23         Update $\mathcal{C} = \mathcal{C} \cup \mathcal{C}^m$
24         for $i = 1, \dots, m$ do
25             Compute the ROM solution $\mathbf{u}_r(t, \mu_i^c; \mathbf{\Phi}_{N_{it}-1})$
26             Compute the corresponding error indicator $\mathcal{I}_i^c(\mu_i^c)$
27             Update $\mathcal{I}^c = \mathcal{I}^c \cup \mathcal{I}_i^c(\mu_i^c)$
28         end
29     end
30     Select $\mu_{N_{it}} = \arg\max_{\mu^\star \subset P} \widetilde{\mathcal{I}}(\mu^\star)$
31     Compute FOM solution $\mathbf{u}(t, \mu_{N_{it}})$
32     Compute ROM solution $\mathbf{u}_r(t, \mu_{N_{it}}; \mathbf{\Phi}_{N_{it}-1})$
33     Compute the associated error indicator $\mathcal{I}_{N_{it}}^p$ and exact error $E_{N_{it}}^p$
34     Update the RB to $\mathbf{\Phi}_{N_{it}}$ using POD-compression with energy tolerances $\varepsilon^i$ and $\varepsilon^g$
35     Compute ROM solution $\mathbf{u}_r(t, \mu_{N_{it}}; \mathbf{\Phi}_{N_{it}})$
36     Compute the error indicator $\mathcal{I}_{N_{it}}^a$ and exact error $E_{N_{it}}^a$
37     Update $\mathcal{E} = \mathcal{E} \cup \left(\mathcal{I}_{N_{it}}^p, E_{N_{it}}^p\right) \cup \left(\mathcal{I}_{N_{it}}^a, E_{N_{it}}^a\right)$
38     Compute the GPRC exact error model $\widetilde{E}\left(\mathcal{I}^\star\right)$ and associated error bound $\tilde{s}_E\left(\mathcal{I}^\star\right)$ using the
        error pool $\mathcal{E}$
39     if converged then
40         return $\mathbf{\Phi}$
41     end
42 end
```

3.3 Sampling Convergence – Error Modeling

In both the SPS and the GPR method, error indicators $(\mathcal{I}^p_{N_{it}}, \mathcal{I}^a_{N_{it}})$ and exact errors $(E^p_{N_{it}}, E^a_{N_{it}})$ are computed at the load locations $\mu_{N_{it}}$ that are used to construct the reduced basis $\boldsymbol{\Phi}_{N_{it}}$. These results are collected in an error pool $\mathcal{E}$ that is used in the error modelling to evaluate whether or not the offline training phase has converged. The error model is built using GPRC, but now noise in the observations is taken into account because the error indicator is merely an indication of the exact error. The covariance function including observation noise σ_n^2 is then:

$$\mathrm{cov}\big(E_p, E_q\big) = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{(\mathcal{I}_p - \mathcal{I}_q)^2}{\lambda_l^2}\right] + \sigma_n^2\delta_{pq} \tag{3.33}$$

$$\mathrm{cov}\big(E_p, E_q\big) = k(\mathcal{I}_p, \mathcal{I}_q) + \sigma_n^2\delta_{pq} \tag{3.34}$$

or collected in matrix form:

$$\mathbf{C}\big(\boldsymbol{E}\big) = \sigma_f^2\boldsymbol{R}\big(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{I}}\big) + \sigma_n^2\boldsymbol{I} = \boldsymbol{K}\big(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{I}}\big) + \sigma_n^2\boldsymbol{I} \tag{3.35}$$

Using this covariance function, the predictive mean $\widetilde{E}$ of the GPRC and associated variance $\tilde{s}^2_E(\mathcal{I}^\star)$ as functions of test inputs $\mathcal{I}^\star$ are:

$$\widetilde{E}(\mathcal{I}^\star) = \overline{m} + \tilde{\mathbf{k}}^T\mathbf{C}^{-1}\big(\boldsymbol{E} - \mathbf{1}\overline{m}\big) \tag{3.36}$$

$$\tilde{s}^2_E(\mathcal{I}^\star) = \sigma_f^2 - \tilde{\mathbf{k}}^T\mathbf{C}^{-1}\tilde{\mathbf{k}} + \sigma_n^2 \tag{3.37}$$

The procedure to arrive at the above functions is similar to the procedure described in 3.2.2 Gaussian Process Regression for Prediction, except the hyperparameter optimization is considerably more complicated. Due to the inclusion of the observation noise, only the hyperparameter $\overline{m}$ can now be determined analytically, and the others $(\sigma_f^2, \sigma_n^2, \lambda_l^2)$ have to be optimized numerically and is often done using a gradient ascent algorithm [3]. In order to perform such an optimization, the derivative of the log-likelihood is needed. The log-likelihood (when including noise) and its derivative is as follows:

$$\log\left(\mathcal{L}\right) = -\frac{size(\mathcal{E})/2}{2}\log\left(2\pi\right) - \frac{1}{2}\log\left(|\mathbf{C}|\right) - \frac{1}{2}\frac{(\boldsymbol{E} - \mathbf{1}\overline{m})^T\mathbf{C}^{-1}(\boldsymbol{E} - \mathbf{1}\overline{m})}{\sigma_f^2} \tag{3.38}$$

$$\frac{\partial\log\left(\mathcal{L}\right)}{\partial\theta_i} = \frac{1}{2}\left(\mathrm{tr}\Big(\boldsymbol{\rho}\boldsymbol{\rho}^T - \mathbf{C}^{-1}\Big)\frac{\partial\mathbf{C}}{\partial\theta_i}\right) \tag{3.39}$$

where $\mathrm{tr}(\cdot)$ is the trace of a matrix, $\boldsymbol{\rho} = \mathbf{C}^{-1}(\boldsymbol{E}^c - \mathbf{1}\overline{m})$, and θ_i is simply the hyperparameters to be optimized: $\sigma_f^2, \sigma_n^2, \lambda_l^2$. With this expression, all we have to do is find $\partial\mathbf{C}/\partial\theta_i$, so in the end the hyperparameters to be optimized depend on the covariance function that was chosen. The derivatives with respect to each hyperparameter are as follows [3]:

$$\frac{\partial\mathbf{C}}{\partial\sigma_f^2} = \boldsymbol{I} \qquad \frac{\partial\mathbf{C}}{\partial\lambda_l^2} = \boldsymbol{R} \qquad \frac{\partial\mathbf{C}_{pq}}{\partial\sigma_n^2} = \frac{1}{2}k(\mathcal{I}_p, \mathcal{I}_q)\left(\frac{\mathcal{I}_p - \mathcal{I}_q}{\lambda_l^2}\right)^2 \tag{3.40}$$

With all these derivatives, the AMSGrad algorithm proposed by Reddi et al. [25] for gradient ascent can be used to optimize the hyperparameters. This method was chosen because the authors claim it consistently outperforms many other methods both in terms of speed and convergence behaviour. The optimization using more basic gradient ascent methods was very often not able to find the global maximum because there is a significant number of local minima, and in many cases the optimization did not converge within a reasonable number of iterations. The procedure for the numerical optimization of the hyperparameters collected in a vector $\boldsymbol{\theta}$ is shown in Algorithm 3.6 [17, 25].

Algorithm 3.6: AMSGrad for hyperparameter vector $\boldsymbol{\theta}$ (all operations on vectors are elementwise)

```
Input  : Initial hyperparameter vector $\boldsymbol{\theta}_1$
         Step size vector $\boldsymbol{a}$
         Exponential decay rates $\beta_1, \beta_2$
         Smoothing term $\epsilon$
         Objective function $\mathbf{C}$
Output : $\boldsymbol{\theta}$
1 Set $\boldsymbol{m}_0 = 0$, $\boldsymbol{v}_0 = 0$ and $\hat{\boldsymbol{v}}_0 = 0$
2 for $t = 1, \dots, N_{GA}$ do
3     $\boldsymbol{g}_t = \nabla_\theta \mathbf{C}(\boldsymbol{\theta}_t)$                         (Gradients to use for optimization: Eq. (3.40))
4     $\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$   (Update biased 1st moment estimate)
5     $\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2$   (Compute new biased 2nd raw moment estimate)
6     $\hat{\boldsymbol{v}}_t = \max(\hat{\boldsymbol{v}}_{t-1}, \boldsymbol{v}_t)$            (Select max biased 2nd raw moment estimate)
7     $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\boldsymbol{a}\boldsymbol{m}_t}{\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon}$           (Update hyperparameters)
8 end
9 return $\boldsymbol{\theta} = \boldsymbol{\theta}_T$
```

The gradient ascent algorithm takes as input the covariance function $\mathbf{C}$, a step size vector $\boldsymbol{a}$, two exponential decay rates β_1, $\beta 2$, a smoothing term ϵ and an initial hyperparameter vector $\boldsymbol{\theta}_1$. As suggested by the authors, the following values are set: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The step sizes and initial hyperparameter values, on the other hand, are significantly more problem-dependent, so they are calculated based on the input. In Reddi et al. [25] and Kingma and Ba [17] (which [25] is based on), the same step size is used for all three hyperparameters, but in this thesis they are different because the length-scale λ_l^2 has considerably higher magnitude. Therefore, the step sizes are calculated from the *initial* hyperparameter values:

$$\boldsymbol{a} = \left[\sigma_{n,1}^2,\ \lambda_{l,1}^2,\ \sigma_{n,1}^2\right] \times 10^{-3} \tag{3.41}$$

The optimization problem is strongly non-concave (non-convex when minimizing) because there a significant number of local maxima of the covariance function $\mathbf{C}$. In such cases a large number of different sets of initial values have to be chosen in order to find the local maximum that is also the global maximum. Figure 3.7 shows an example of optimizing an arbitrary non-concave function C that depends on two hyperparameters, θ_1 and θ_2. It has three local maxima in the range where it is defined. The three red lines show the progression of a gradient ascent algorithm with three different starting points. It shows that the result of the optimization depend greatly on the starting point. This highlights how important it is to have a large number of sets of initial hyperparameters in order to find the global maximum, especially with a strongly non-concave function such as the covariance function that has to be optimized in this problem.

With the hyperparameters optimized, the GPRC can finally be computed using Eq. (3.36) and Eq. (3.37), which Figure 3.8 shows an example of. Using this mapping of the exact error, we can compute a *pessimistic* exact error $\hat{E}^+$ (or upper bound for the error [21]):

$$\hat{E}^+ = \widetilde{E}(\mathcal{I}^\star) + \alpha_E \tilde{s}(\mathcal{I}^\star) \tag{3.42}$$

where α_E is the number of standard deviations away from the mean. For this thesis $\alpha_E = 1.036$, which corresponds to the 70% confidence interval. The offline training phase is converged when the *three* last error indicators computed *before* updating the basis (E^p) that were added to the error pool $\mathcal{E}$ are below an error tolerance E^{max} (Figure 3.8b), which is set as $E^{max} = 0.5\%$.

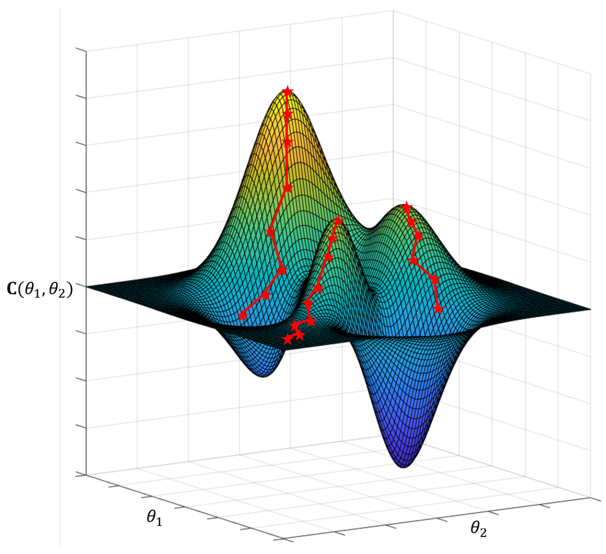
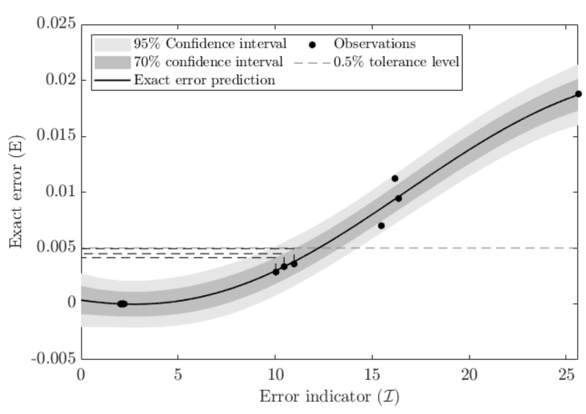
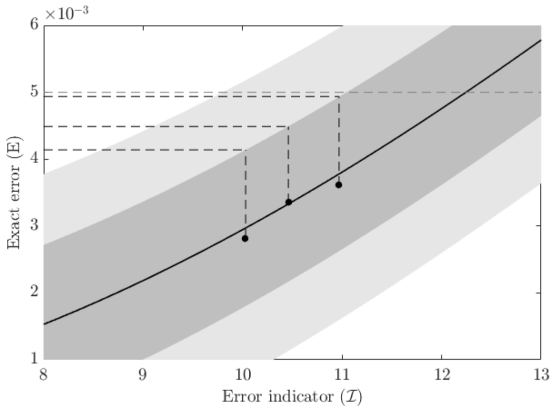


Figure 3.7: Gradient ascent example



(a) Full error mapping



(b) Zoomed portion of error mapping

Figure 3.8: Example of error mapping using GPRC at convergence

3.3.1 Hyperparameters

In order to obtain a good regression for both GPRP and GPRC, it is essential that the hyperparameters are optimized properly. The more hyperparameters that have to be numerically optimized, the more expensive it becomes to compute the optimization, but following is a description of how the hyperparameters affect the regression and the importance of optimizing them well. Figure 3.9 shows a regression computed from a number of arbitrary observations y that are based on inputs x. It is judged that this regression has well-tuned hyperparameters in terms of maximizing the log-likelihood function, and in the remainder of this section, this regression will referred to as a "good" regression. The following figures show regressions made from the same observations, but with each hyperparameter adjusted to be lower or higher in order to see the effect they have on the regression. For reference in order to understand how the hyperparameters play a role in computing the regression, the squared exponential covariance function C for input x and output y is:

$$\mathbf{C} = \sigma_f^2 \exp\left[-\frac{1}{2}\frac{(x_p - x_q)^2}{\lambda_l^2} \right] + \sigma_n^2 \delta_{pq} \tag{3.43}$$

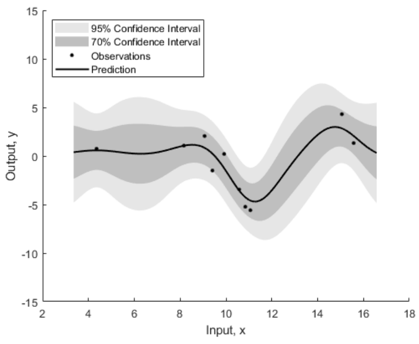
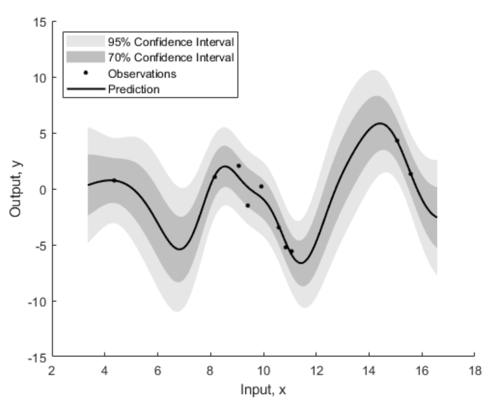


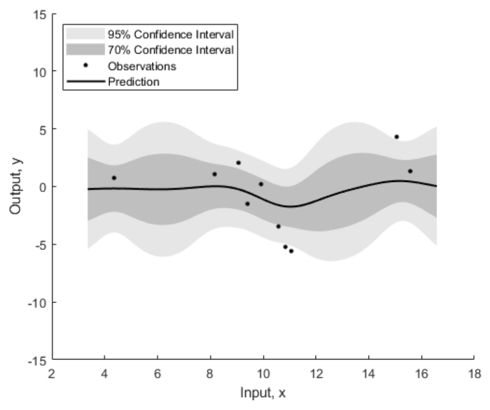
Figure 3.9: Example of GPR with good hyperparameter optimization

Signal variance σ_f^2
The signal variance σ_f^2 is often referred to as the *amplitude* of the predictive mean function because it describes the distance from the mean of the observations, which in this case is roughly zero [32]. Figure 3.10a shows that a high variance leads to high amplitudes along the domain of the input. When comparing this regression with the good one, roughly the same peaks are present, but the adjusted σ_f^2 makes the amplitude of these peaks considerably higher. In the case when σ_f^2 is lowered, the peaks are much closer to the mean, and the function really only moves away from the mean when there are points that are very far away from the mean.

Length-scale λ_l^2
The length-scale λ_l^2 (also referred to as *bandwidth*[2]) specifies the width of the kernel and thus describes the smoothness of the regression. In other words, it determines how quickly the correlation between points drops off as one moves along the domain x. Figure 3.11 shows heatmaps of a covariance matrix C with 1000 arbitrary observations that are generated in the same way as the observations used for all the other figures in this section. The value indicates the correlation between the points collected in the matrix, where a value of 1 is the highest and indicates very high correlation. It is clear that a higher length-scale increases the correlation between the points, and almost all the points affect each other to a great extent. Such a high λ_l^2 results in a smooth function, as seen in Figure 3.12a. Furthermore, the higher the length-scale, the more the regression is able to extrapolate the data reliably. In general, we cannot extrapolate more than λ_l away from the outer observations [6]. Figure 3.11c shows that the high values along the diagonal quickly, and points that are not in the immediate vicinity are not correlated at all. Figure 3.12b shows that a low λ_l^2 makes the regression always tend to go back the mean if there are no other points nearby.

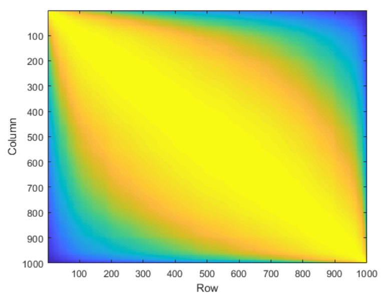


(a) High σ_f^2

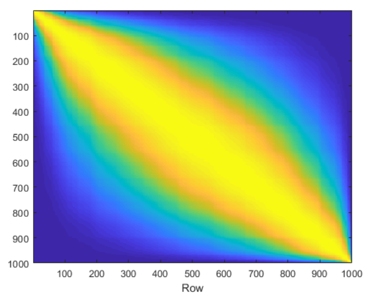


(b) Low σ_f^2

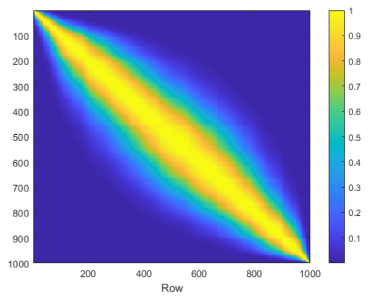
Figure 3.10: Examples of GPR with high and low σ_f^2



(a) High λ_l^2

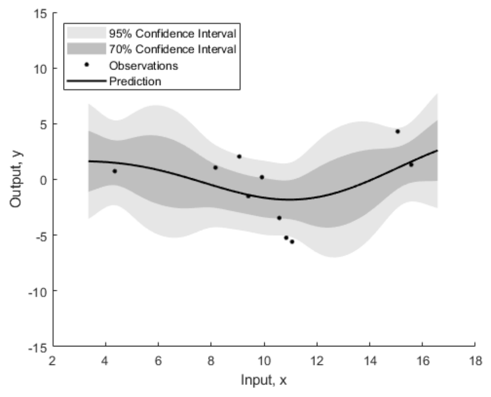


(b) Good λ_l^2

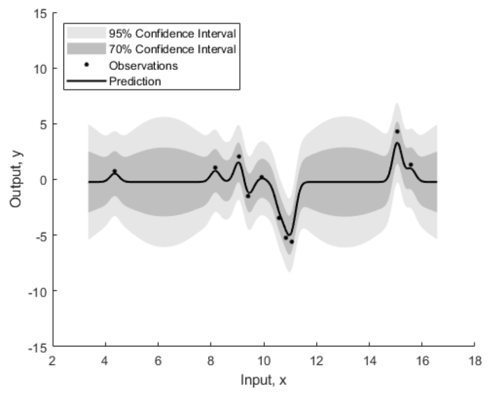


(c) Low λ_l^2

Figure 3.11: Heatmap of covariance matrix for various length-scales



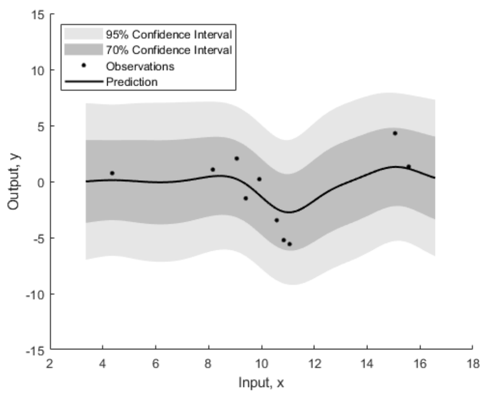
(a) High λ_l^2



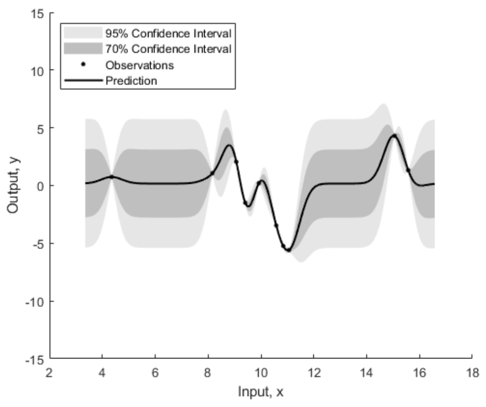
(b) Low λ_l^2

Figure 3.12: Examples of GPR with high and low λ_l^2

Observation noise σ_n^2
The observation noise σ_n^2 describes the uncertainty in the observations and thus determines how well the regression fits the points. If we believe that there is noise in the observed values, then this observation noise has to be implemented. However, in some cases, such as the GPRP, the error indicator $\mathcal{I}$ is computed at certain locations μ, and are therefore not uncertain at all, which is why no noise is included in that case. Figure 3.13b shows an example of this, and it can be seen that once we move away an area with several observations, the function goes back toward the mean, just like when there is a low length-scale. However, in this case there is a constant predictive variance when the regression is at the mean, but Figure 3.12b shows that the variance varies over the same ranges. It only makes sense to exclude the noise in a situation when there are few observations because the combination of low noise and many observations yields a regression like in Figure 3.14. A high observation noise, as shown in Figure 3.13a implies that there is a lot of uncertainty in the observations, yielding a smoother regression with a lot of variance, and actually a mostly constant variance over the domain x.



(a) High σ_n^2



(b) Low σ_n^2

Figure 3.13: Examples of GPR with high and low λ_l^2

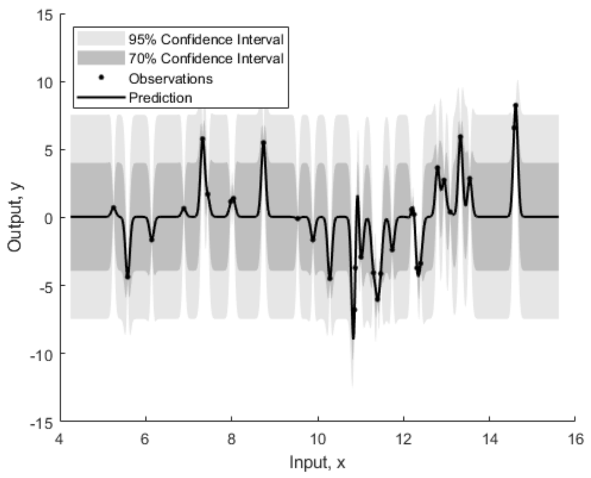
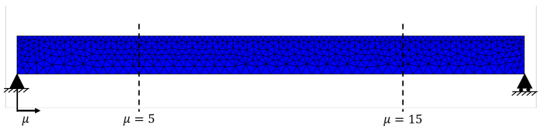


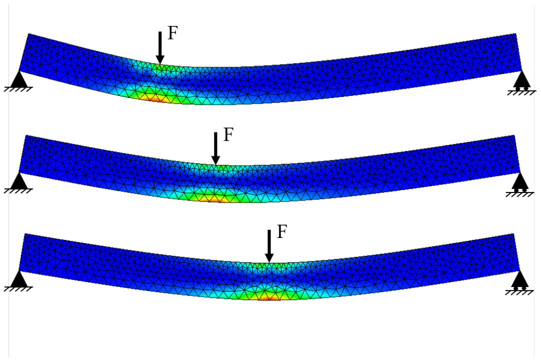
Figure 3.14: Example of GPR with low σ_n^2 and many observations

4 RESULTS AND DISCUSSION

In this thesis two methods of constructing a reduced-order model (ROM) were investigated. The ROM should accurately describe the behaviour of a simply supported beam that is loaded in the downwards direction (Figure 4.1). In this chapter, a short description and values are given for all parameters used in the implementation of the methods that have been mentioned throughout the thesis, then the progression of the two methods are presented to show how each of them behave. Lastly, the results of a parametric study is shown. This study investigates the performance of both methods with varying input parameters, where performance is assessed in terms of efficiency and accuracy. For the SPS these input parameters are the energy tolerances ε^i and ε^g, and for the GPR method they are the same energy tolerances as well as the number of additional improvements to the GPRP N_c^{add}.



(a) Beam model with no load



(b) Examples of FOM responses for beam model

Figure 4.1: Simply supported beam model

Mechanical Model
The model used in this thesis is a simply supported beam that is a solid, rectangular and made of a homogenous, elastoplastic material. The full span of the beam is 20 mm, but the actual loading span is limited to $5 \leq \mu \leq 15$ mm. Only one point is loaded at a time, and the load is applied using arc-length control. The magnitude of the downwards load is described by the load factor λ which in each time step is incremented by $\Delta\lambda$. The FOM is run while $\Delta\lambda_{FOM} \geq 10^{-4}$ is satisfied, meaning that the response is well into the plastic regime (Figure 3.2b), and the load is barely increasing anymore. This ensures that sufficient displacement snapshots are gathered to assemble the reduced basis. We desire a ROM that is suitable for significant plastic responses, but due to erroneously high error indicators along the edges of the parameter space, the stopping criterion for the ROM had to be fine-tuned in order to get a reasonable distribution of the error indicator in the entire parameter space. Therefore, the ROM is run while $\Delta\lambda_{ROM} \geq 10^{-4} \ \wedge \ \lambda_{ROM} \leq 9.5$ is satisfied.

$\Delta\lambda_{FOM}$	$\geq$	10^{-4}
$\Delta\lambda_{ROM}$	$\geq$	1.75×10^{-3}
λ_{ROM}	$\leq$	9.5
Length [mm]		20
Parameter space span [mm]		$5 \leq \mu \leq 15$

Table 4.1: Arc-length control load settings

Reduced Basis Construction

The input parameters that affect how the reduced basis is constructed are the number of elastic modes k^e, and the energy tolerances ε^i and ε^g. $k^e = 1$ because all the information from the elastic regime is captured well by just one mode. ε^i and ε^g are the two of the input parameters that are varied in the parametric study for both sampling methods. They greatly affect the efficiency and accuracy of the offline training phase, as they determine how much information is kept in the SVD performed on the snapshot matrices. The values used for each energy tolerance are listed in Table 4.2 below.

ε^i	0.9995, 0.9999, 0.99995, 0.99999, 0.999995, 0.999999
ε^g	0.9999, 0.99995, 0.99999, 0.999995, 0.999999
k^e	1

Table 4.2: Reduced Basis Construction Parameters

Gaussian Process Regression for Prediction (GPRP)

There are various parameters that affect the performance of the GPR method during the computation of the GPRP, and they are all related to the improvement of the regression. The search parameter $\boldsymbol{\tau}$ ranges from 0.05 to 4.00 (see Table 4.3) in order to perform both a local and a global search to improve the regression. The number of searches is high because it is inexpensive to compute. The number of initial observations used to construct the GPRP N_c^0 is set as 3 because that is the lowest number of observations required to make a poor but reasonable regression. From this regression, N_c^{add} additional improvements are made. N_c^{add} is the third input parameter in the GPR method that is varied in the parametric study, and ranges from 1 to 6. β and $\alpha_{\mathcal{I}}$ are the factors that determine when it can be judged that the maximum error indicator has been found by the GPRP, and they are set as 5% and 1.96, respectively. The latter value corresponds to a 95% confidence interval of the regression.

$\alpha_{\mathcal{I}}$	1.96
β	0.05
N_c^0	3
N_c^{add}	1, 2, 3, 4, 5, 6
$\boldsymbol{\tau}$	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.25, 3.50, 3.75, 4.00]

Table 4.3: GPRP Parameters

Gaussian Process Regression for Convergence (GPRC)

The GPRC is computed in the same way for both the SPS and the GPR method, so this specific process of the offline phase should take the same amount of time in each iteration for both methods. The GPRC is first computed in $N_{it} = 5$, which is to reduce the computational cost a little. In the early iterations it is certain that the offline training process will not converge either way, so the computation of the GPRC can be initiated later. Furthermore, in the initial computation of the GPRC, only the error indicators $\mathcal{I}$ and exact errors E computed in the last two iterations, $N_{it} = 4$ and $N_{it} = 5$, are used, meaning that $\mathcal{I}$ and E from training iterations 1, 2 and 3 are not used. These three iterations are left out because in the beginning of the offline training phase, the load locations are selected far away from each other, and with a poor basis the error indicators fluctuate significantly, and also depend on whether the GPRP selects points around the middle or at the edges first. Furthermore, the erroneously high error indicators at the edges makes these fluctuations worse. In almost every case, two of the first load cases are at the edges, and one is somewhere around the middle, so at this point the distribution of the error indicator has stabilized since the basis includes information from a few points that range the parameter space. This is why the first three iterations are left out. For the convergence criterion explained in 3.3 Sampling Convergence – Error Modeling, the error tolerance E^{max} is set as 0.5%, and $\alpha_E = 1.036$, which corresponds to a 70% confidence interval.

E^{max}	0.5%
α_E	1.036
N_{it} to begin GPRC	$N_{it} = 5$
N_{it} used in 1st GPRC	$N_{it} = 4, 5$

Table 4.4: GPRC Parameters

Gradient Ascent Algorithm

For the AMSGrad gradient ascent algorithm, some parameters are set based on suggestions from the authors, while others are highly problem-dependent and have to be set by the user. The suggestions from the authors are: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ [17, 25]. The step sizes are more problem-dependent because of the varying magnitudes of the hyperparameters that are optimized, which is why they are calculated based on the initial values used in one run of the optimization algorithm. The initial values also depend on the provided observations, so for each hyperparameter an *estimate* is computed: $\sigma^2_{f,est}$, $\lambda^2_{l,est}$ and $\sigma^2_{n,est}$. Then, seven different initial values $(\sigma^2_{f,1}, \lambda^2_{l,1}, \sigma^2_{n,1})$ are calculated for each hyperparameter as shown in Table 4.5, and the gradient ascent algorithm is run for a maximum of $N_{GA} = 100$ iterations with every combination of initial values.

$\boldsymbol{a}$	$\left[\sigma^2_{n,1}, \lambda^2_{l,1}, \sigma^2_{n,1}\right] \times 10^{-3}$
β_1	0.9
β_2	0.999
ϵ	10^{-8}
N_{GA}	100
$\lambda^2_{l,est}$	$max(\mathcal{I}_p \cup \mathcal{I}_a) - min(\mathcal{I}_p \cup \mathcal{I}_a)$
$\lambda^2_{l,1}$	$\lambda^2_{l,est}, \frac{\lambda^2_{l,est}}{5}, \frac{\lambda^2_{l,est}}{10}, \frac{\lambda^2_{l,est}}{25}, \frac{\lambda^2_{l,est}}{50}, \frac{\lambda^2_{l,est}}{75}, \frac{\lambda^2_{l,est}}{100}$
$\sigma^2_{f,est}$	$\sqrt{var(E^p \cup E^a)}$
$\sigma^2_{f,1}$	$10\sigma^2_{f,est}, \sigma^2_{f,est}, \frac{\sigma^2_{f,est}}{10}, \frac{\sigma^2_{f,est}}{50}, \frac{\sigma^2_{f,est}}{100}, \frac{\sigma^2_{f,est}}{500}, \frac{\sigma^2_{f,est}}{1000}$
$\sigma^2_{n,est}$	$var(E^p \cup E^a)$
$\sigma^2_{n,1}$	$10\sigma^2_{n,est}, \sigma^2_{n,est}, \frac{\sigma^2_{n,est}}{10}, \frac{\sigma^2_{n,est}}{50}, \frac{\sigma^2_{n,est}}{100}, \frac{\sigma^2_{n,est}}{500}, \frac{\sigma^2_{n,est}}{1000}$

Table 4.5: Gradient Ascent Parameters

4.1 Greedy Sampling Progression

4.1.1 Surrogate Parameter Space Method

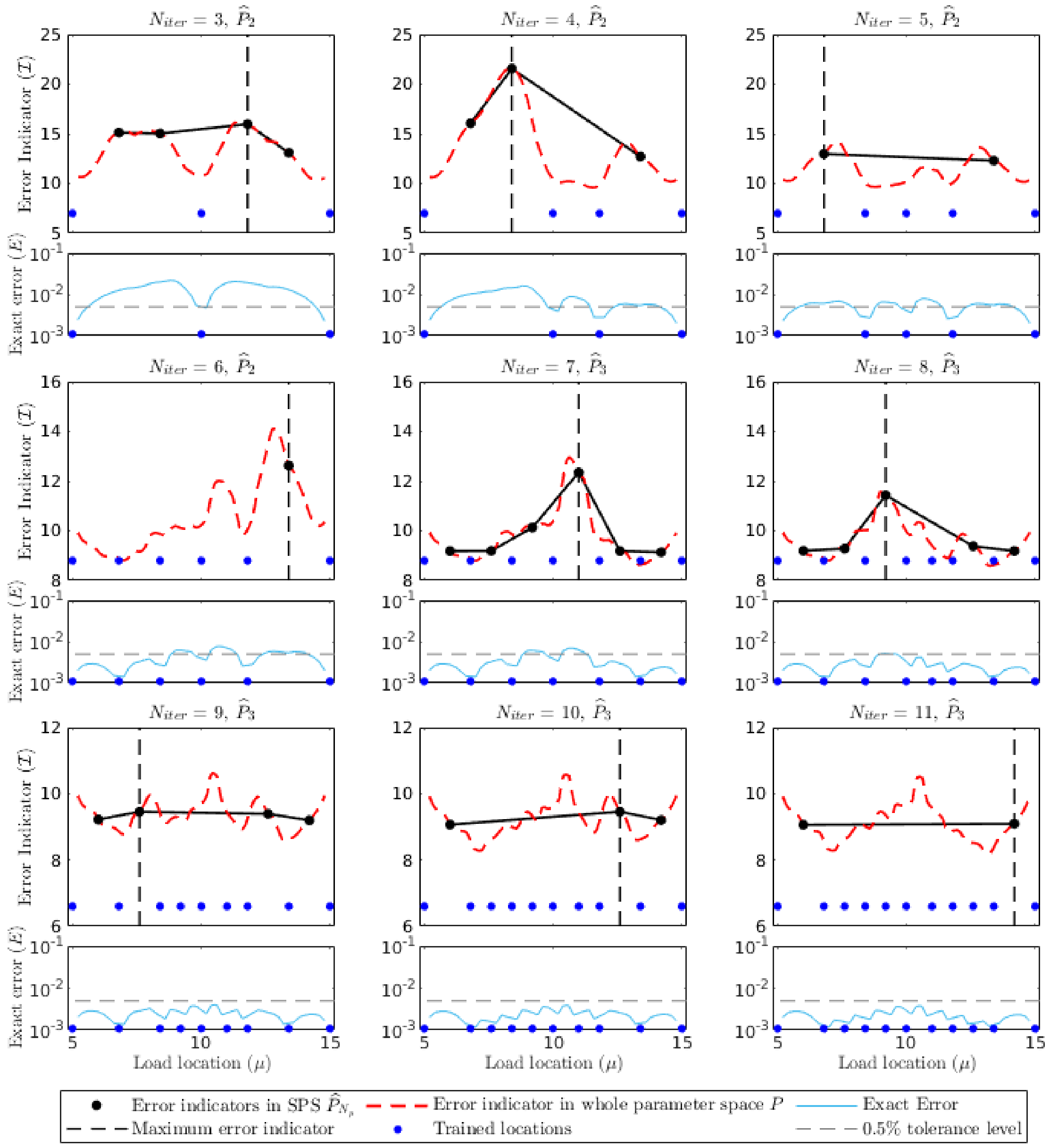


Figure 4.2: Iterative progression of the SPS method

Figure 4.2 shows how the load cases $\mu_{N_{it}}$ are selected in each iteration and the SPSs $\widehat{P}_{N_p}$. It also compares the approximation of the error indicator $\mathcal{I}$ that can be made from those computed in $\widehat{P}_{N_p}$ with the distribution of the error indicators in the full-order parameter space P. The figure also shows how the exact error curve develops as data from more load cases is added to the reduced basis.

The figure starts at $N_{it} = 3$ and shows the sampling progression until convergence in $N_{it} = 11$. In the top plot in each iteration, the black dots are the error indicators $\widehat{\mathcal{I}}$ computed at the points that have not yet been trained in the current SPS $\widehat{P}_{N_p}$, and the dashed red line shows the error indicator in the parameter space P. The distribution of the error indicator in the rest of the parameter space is inaccessible during the offline training phase, but it has been obtained here for illustration purposes. In $N_{it} = 3$, the training phase has just moved on to SPS $\widehat{P}_2$, in which there are four points. In each iteration one point is trained and the error indicator will not be computed in a trained point in the following iteration, so in $N_{it} = 4$ there are only three untrained points as shown. In $N_{it} = 6$ there is only one point left, so this point can be selected as the next load case without the need to compute the error indicator. However, the error indicator is actually computed in this location before enrichment of the basis either way because it will be used for the error modelling used to evaluate whether the training phase has converged or not. In $N_{it} = 7$, the training phase moves on to the next SPS $\widehat{P}_3$ because the previous one has been exhausted. The dashed black line shows the location of the maximum error indicator in each iteration, which is the point that will be trained next, and the blue dots show the points that have already been trained prior to the respective iteration. The solid black line is the best approximation available of the distribution of the error indicator in the SPS method.

The bottom plot in each iteration shows the development of the exact error E in the parameter space P as the training progresses. Just like the error indicator in P, this exact error curve is also not accessible during the offline training phase. The dashed gray line is the $E^{max} = 0.5\%$ error tolerance, and it is the goal of the offline training phase to construct a ROM that ensures that every single point along the beam has an exact error lower than this tolerance. Neither the error indicator nor the exact error in the parameter space P (red and blue curves) are computed in the already trained points because that would only serve to visually disturb the plots.

Before $N_{it} = 9$, the points in the SPSs are always at or close to the maximum error indicator in the beam. This occurs because the grids in each SPS are made such that the beam will be trained at a number of locations with uniform distance from each other when the SPS is exhausted. For a simply supported beam of homogenous material, one could intuitively think that training such a uniformly spaced grid is the best and most efficient way to construct a reduced basis, given that we know the spacing required in order to bring the exact error below a certain tolerance everywhere. While the reduced basis is indeed constructed well and relatively efficiently, the selection of the load cases in each iteration is often suboptimal, which is clearly shown in $N_{it} = 9$ and the following iterations. Before discussing the suboptimality of the load cases selected, we first have to address the high error indicators at the very edges. It can be seen starting in $N_{it} = 6$ that the error indicator spikes towards the edges and is erroneously high. Figure 4.3 shows the distribution of the ROM residual (Eq. (3.13)) after running the ROM in the online phase with loads at $\mu = 10.8$, $\mu = 12.8$ and $\mu = 14.8$. The residual is an indication of the ROM error as the MOR aims to minimize this residual. The locations loaded are all next to points that have been trained in the offline phase shown in Figure 4.2 ($\mu_8 = 11.0$, $\mu_{10} = 12.6$ and $\mu_3 = 15.0$). The figure shows that when loads are placed closer to the middle of the beam, the residual is significantly spread out in the beam, and has a generally higher magnitude throughout. This is likely because the load is transferred to both supports through larger portions of the beam, making it more difficult for the reduced model to accurately capture the behaviour in the elements. As the load moves more towards the edges, the stress is transferred through smaller portions of the beam, causing the residual to be less spread out and generally smaller in magnitude. However, when getting too close to the support, there is significant resistance from the closest support, impeding deflection and causing residual to build up significantly around the load location. While the residual is now less spread out and generally smaller in magnitude, the residual built up around the loaded point now dominates, causing a higher error indicator. This high build-up of the residual around the load locations is due to the fact that we are using a point load. It can be seen in all three cases that the residual is significantly higher around the load location than anywhere else. When the load is placed closer to the middle, the residual in the rest of the beam dominates, but when moving too much to the edges, the residual around the load location dominates. The exact error is computed based on the displacement field, and Figure 4.2 shows in the last few iterations that the exact error dips down at the edges, while the error indicator goes up, illustrating that this significant build-up of the residual is a numerical error that we do not want to capture.

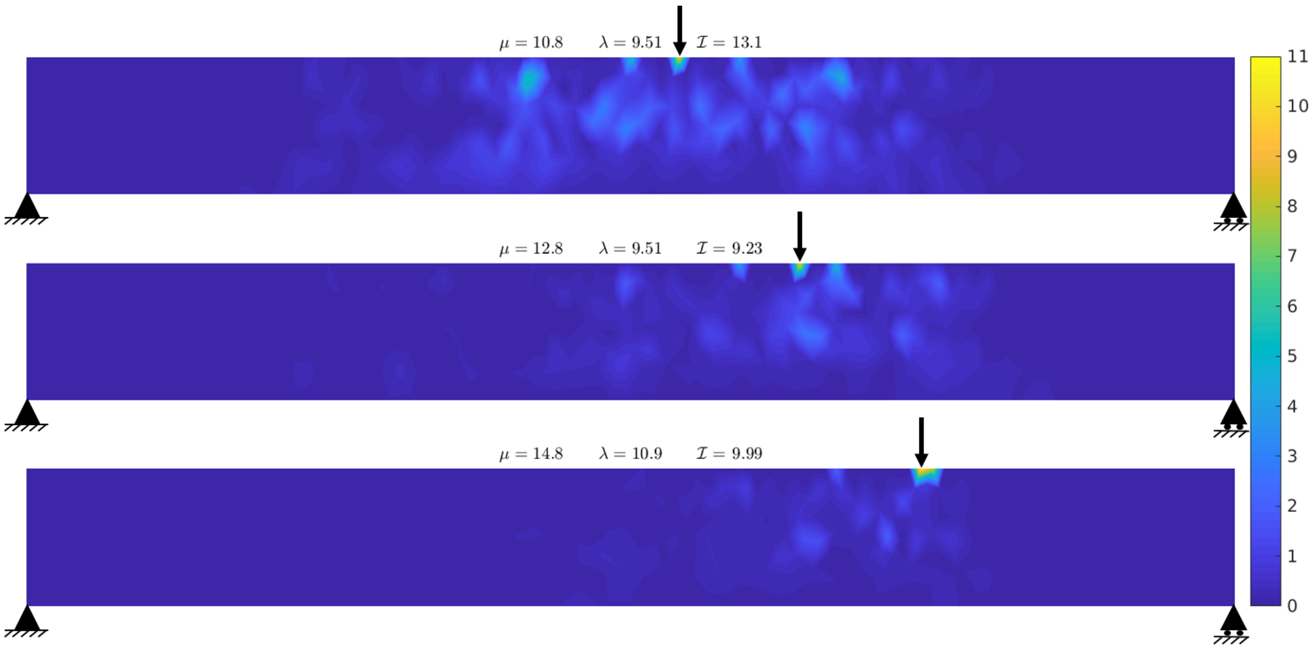


Figure 4.3: Distribution of residual for various load locations

When looking at the error indicator curve in $N_{it} = 11$, the value goes down as we move away from the middle, and even though the small areas on both sides have yet to be trained, both the error indicators and the exact errors there are lower than in the middle. This shows that further away from the middle, fewer load cases are needed in those areas to describe the behaviour of the beam. So it is evident that even for a simple model like this, the load cases can be selected in a more optimal way. Furthermore, one would also expect the error indicator curve to be symmetric when symmetric load cases have been used, but this is not the case as clearly shown in iterations 3, 5, 7, and 9. This occurs because of how the SVD keeps information from the snapshots. Depending on the order of the iterations, the SVD might not deem information obtained from one side of the beam as important as when the other side was trained due to all the other information that has already been stored in the global surrogate snapshot matrix. However, this asymmetry is less present in the exact error, so in terms of the accuracy of the ROM itself, enough good information is kept in the SVD.

The exact error plots show that one training case significantly decreases the error in the vicinity of the training location. However, this vicinity is very small. In $N_{it} = 3$ it can be seen that the exact error is above 0.5% when moving just 0.2 mm away, and in $N_{it} = 4$ the affected area of the new load case is slightly larger. This shows that load cases around the middle add useful information to the reduced model in only a small vicinity, indicating again that the middle should be more densely trained. The reason for this is likely because the load is transferred through larger portions of the beam when loaded close to the middle, as mentioned above. When more elements in the finite element model undergo nonlinear behaviour, the ROM has troubles interpolating between the modes in the reduced basis, and cannot accurately capture the behaviour.

Furthermore, Figure 4.4 shows an example of how $\mathcal{I}$ is merely an indicator, and why the offline training phase requires a relatively conservative convergence criterion. The top four curves shows the distribution of the error indicator in iterations $7 - 10$, and the bottom four curves are the corresponding exact errors. The dots show the trained locations in the indicated iterations. What we can see in this figure is that while the exact error consistently decreases in every point as more information is added in each iteration, this is not the case for the error indicator. E.g. as we go from iteration 7 to 8, the error indicator around $\mu = 9$ increases considerably when the ROM is enriched by a load case at $\mu = 11$. However, the exact error in the same area does go down quite significantly. This likely occurs because the snapshots are of the displacements in the FOM solutions, so the ROM more accurately approximates the displacement field than the internal forces.

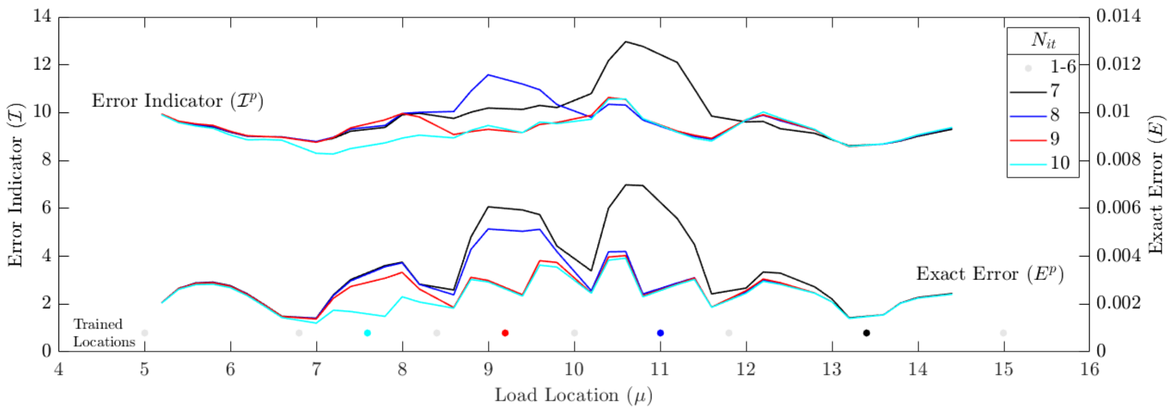


Figure 4.4: Development of error indicator and exact error for a few iterations

Lastly, the exact error plots in Figure 4.2 also show that after $N_{it} = 9$, the exact error is in fact below the 0.5% tolerance in the entire parameter space, but the training goes on for two additional iterations due to the convergence criterion.

4.1.2 Gaussian Process Regression Method

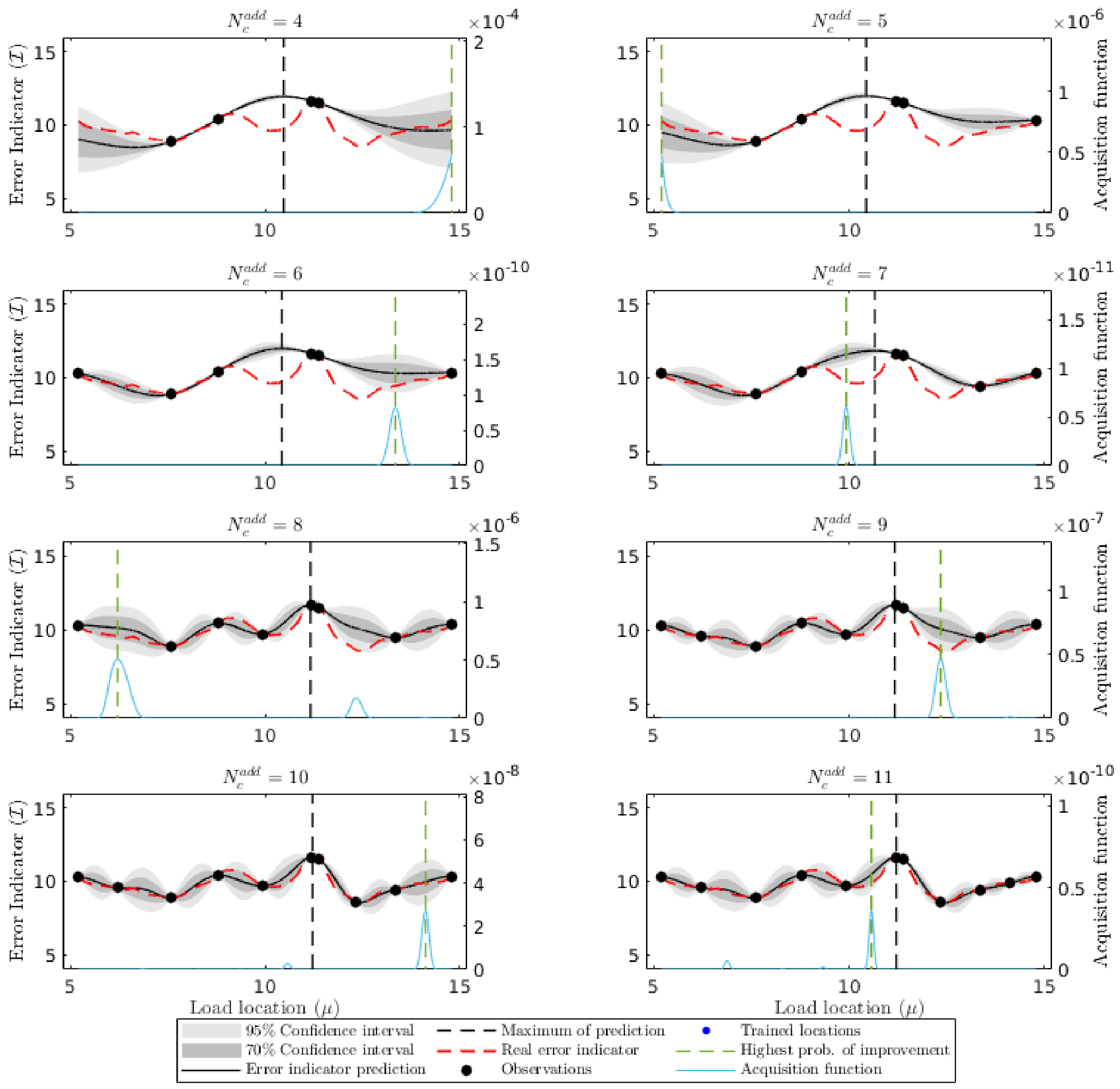


Figure 4.5: GPRP improvement progression using one target

Figure 4.5 shows how the GPRP is built with N_c improvements and illustrates how the accuracy of the regression is affected by the number of improvements. The goal of the GPRP is, in each iteration, to find the true maximum error indicator in the beam as efficiently as possible. The first plot shows the GPRP after $N_c^0 = 4$ initial observations are computed in random locations. The left axis in each plot is the error indicator $\mathcal{I}$, and the right axis is the acquisition function which indicates the probability of improving the maximum of the regression by a certain target. The x axis is the span of the beam represented by the parameter μ. The black dots are the observations from which the regression is computed, the black line is the predictive mean, and the dark and light gray areas show the 70% and 90% confidence intervals of the regression, respectively. The dashed red line is the actual error indicator, which is normally inaccessible, but has been obtained for illustration purposes. The actual error indicator is computed in many points along the beam, but not in the already trained points because there would be significant dips in the graph that are not of much use for the purpose of these plots. The blue dots show the locations in the beam that have already been trained in the previous iterations, and the dashed black line shows the location of the current maximum of the prediction.

The blue line is the acquisition function that gives the probability of improving the regression by a certain target. In this illustrative case, only one high target is used to enable a global search. In the actual implementation, 29 different targets are used that compute the probability of improving the maximum by factors ranging from $\tau_1 = 0.05$ to $\tau_{29} = 4.0$. We use such a large range because the computational cost of this step is negligible. The dashed green line simply shows where the next observation of the error indicator will be obtained.

Coincidentally in this case, one of the first random observations actually turned out to be at or very close to the actual maximum error indicator in this iteration. However, the regression for $N_c = 4$ predicts the maximum to be slightly to the left and very close to a point that has already been trained. If the improvement had stopped there, then the chosen training case would be very close to an already trained point, which is generally a waste of time. As more improvements are made, the regression becomes more accurate and resembles more the actual error indicator. The fifth improvement $N_c = 7$ gives an observation that produces a regression that predicts the true maximum, however because there is no access to the actual error indicator, it could very well be that the maximum is actually in the area on the left with high uncertainties. The method recognizes this, and therefore chooses a point in this area as the next location to improve the regression.

What can also be seen in these steps of constructing the regression is that the uncertainty is sometimes underestimated, especially with few, scattered observations. In this case, it appears that the GPRP finds a smooth curve with a shape that resembles a sine function that easily describes the four points, so there is only significant uncertainty outside the area enclosed by the observations. However, when more observations are added the shape becomes more complex and contains more inflection points, causing the uncertainty to increase in the middle section. After $N_c = 7$, almost the entire actual error indicator curve is captured by the 95% confidence interval.

Figure 4.6 shows the final GPRP in each training iteration N_{it}, and is the GPR method's counterpart of Figure 4.2. It shows the quality of the predictions also how the exact error curve develops as data from more load cases is added to the reduced basis. The figure begins with $N_{it} = 3$ and ends at $N_{it} = 11$ when the offline training phase has converged. To compute these regressions, $N_c^0 = 4$ observations of the error indicator are made at random locations, and improvements are made until there is nowhere in the beam where the $\alpha_{\mathcal{I}} = 95\%$ confidence interval indicates that the regression could be $\beta = 5\%$ higher than the current predicted maximum. Basically, the improvement stops when there is high certainty that the predicted maximum is the true maximum.

In every iteration except $N_{it} = 9$ and $N_{it} = 11$, the GPRP predicts the location of the maximum either exactly at the true maximum or very close to it. In $N_{it} = 9$ we see an example of a poor regression that occurs sometimes when there are only a few observations included. In some poor regressions such as this one, the predictive mean goes to the mean $\overline{m}$ as soon as we move away from the observations. In this case, an observation was made at the edge that was almost as high as the true maximum, and the 95% confidence intervals along the rest of the beam did not indicate that there could be another location where the value could be 5% higher than the current maximum, so the improvement was stopped.

In $N_{it} = 3$ it can be seen that the exact error goes above 0.5% when moving just $0.3 - 0.4$ mm away from the load case that is more in the middle. In $N_{it} = 4$ the affected area of the new load case is even smaller, indicating that the middle part of the beam must be trained more densely than towards the sides, which is in fact what occurs in the later iterations.

It can also be seen in the development of the exact error that all load cases do lower the error very slightly in the rest of the beam. This is seen in the iterations from $N_{it} = 3$ to $N_{it} = 7$ in the gap between the two load cases on the right. Before $N_{it} = 8$, no load case is added in that gap, but the exact error is

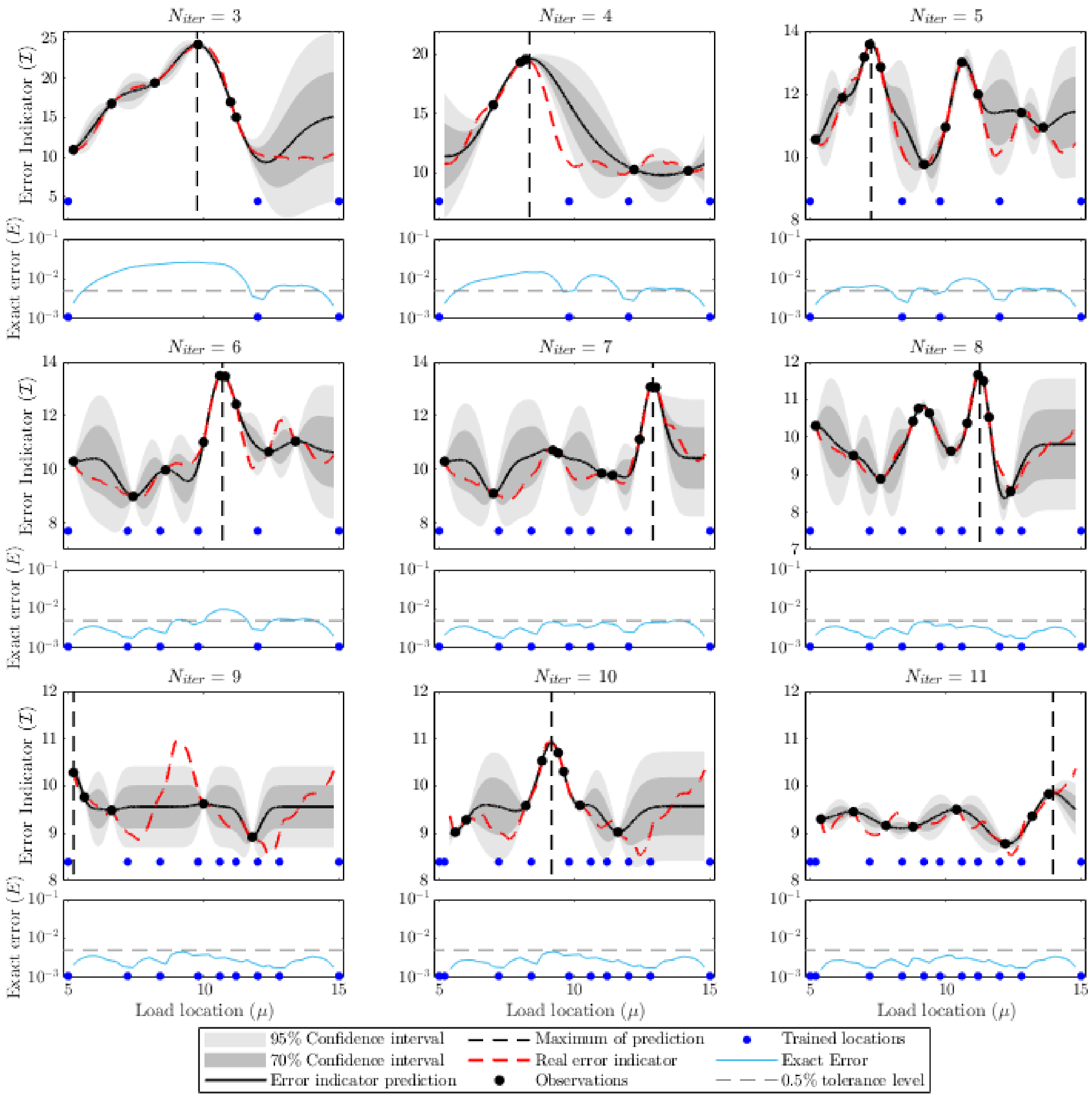


Figure 4.6: Iterative progression of the GPR method

still going down. In $N_{it} = 7$ the error in that gap is even brought below the tolerance. And of course, the load cases applied closer to this area have the greatest affect, as can be seen going from iteration 3 to iteration 4 and iteration 6 to iteration 7, as opposed to the very small decreases from iteration 4 to 5 and 5 to 6. The exact error plots also show that after $N_{it} = 7$, the exact error is in fact below the 0.5% tolerance in the entire parameter space, but the training goes on for four additional iterations due to the convergence criterion.

Just like in the SPS method, the error indicator is erroneously high toward the edges of the parameter space, which disturbs the sampling process of the GPR. In $N_{it} = 9$, the actual error indicator and the exact error follow roughly the same trend, except at the edges where the error indicator goes up quite a bit. This leads to the GPRP predicting a maximum at the left edge right next to a previously trained point, where the exact error is actually already very low and has been below 0.5% since $N_{it} = 3$. These erroneously high error indicators at the edges occurs in every offline training phase, and Figure 4.7 shows how often those edges are trained because of that. This figure shows how the training locations are distributed throughout the beam for all the cases in the parametric study. The high density is because the edges are trained every single time, and also because in almost every training, at least one edge is trained twice, as is the case in Figure 4.6 on the left edge. So after all it was actually lucky in this occasion that the GPRP predicted the wrong maximum in $N_{it} = 11$ because the exact error is higher at the predicted maximum error indicator than the true maximum. Figure 4.7 also shows the middle is more densely trained because the GPRP understands which areas needs to be more trained for the reason discussed in 4.1.1 Surrogate Parameter Space Method.

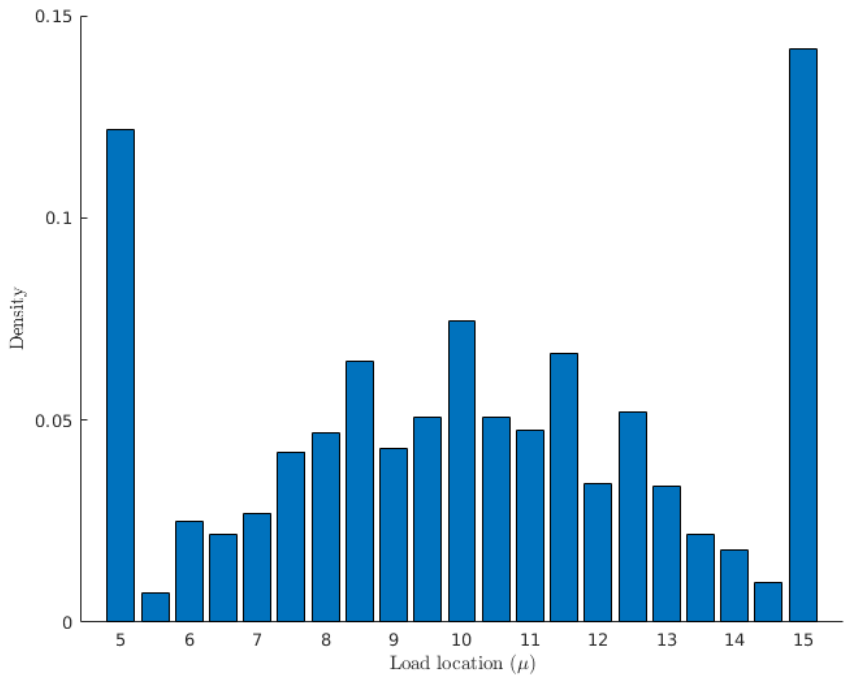


Figure 4.7: Density of training locations

4.2 Convergence Progression

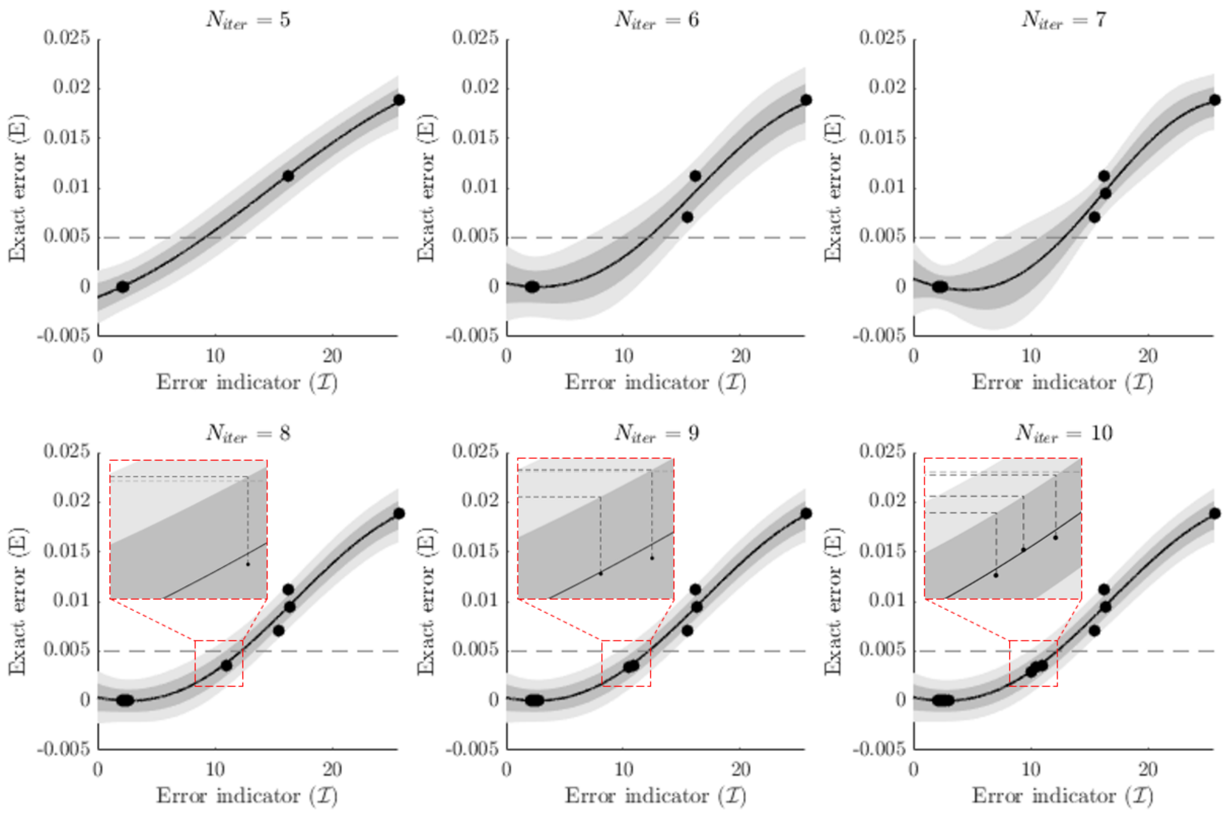


Figure 4.8: Iterative progression of error mapping using GPRC

Figure 4.8 shows the progression of the error mapping and illustrates at which point convergence is achieved. It shows the GPRC computed using observations of the exact errors and error indicators before (E^p, $\mathcal{I}^p$) and after (E^a, $\mathcal{I}^a$) enrichment of the basis in each training iteration from $N_{it} = 5$ to $N_{it} = 10$. The cluster on the left are all observations made shortly after enrichment, so those errors are low, as expected. Like in the previous figures in this chapter, the black dots are the observations from which the regression is computed. The black line is the mean of the prediction, and the dark and light gray areas are the 70% and 95% confidence intervals of the regression, respectively. The dashed gray line is the $E^{max} = 0.5\%$ error tolerance.

The convergence criterion (3.3 Sampling Convergence – Error Modeling) is meant to ensure that the exact error along the parameter space is actually below $E^{max} = 0.5\%$. There are a few reasons why we choose this criterion instead of letting it converge the first time the pessimistic error is below the tolerance. One important reason is the erroneously high error indicator at the edges. This could cause the GPRP to choose the next load location to be at the edge where a training has already been performed, which will yield a low exact error, while the error could still be too high elsewhere in the beam. Another reason is that sometimes the regression simply turns out to be poor, like in $N_{it} = 9$ in Figure 4.6, which may again cause the regression to select a load location that is already well described by the reduced model. This convergence criterion could be changed to converging when a certain number of observations of the pessimistic error are below the tolerance instead of the last few observations. This could work better because e.g. if the GPR predicted two locations that were close to some previously trained locations, the two observations arising from these trainings would have very low errors. However, the next prediction could be a good one, yielding an observation with a pessimistic error above the tolerance, making the count starting over again. At this point, we need 3 more predictions of the pessimistic error, when in fact it could be that every single location along the beam would give pessimistic errors below the tolerance. If using this alternative criterion, the number of observations that have to be below the tolerance has to be higher that of the orginal criterion.

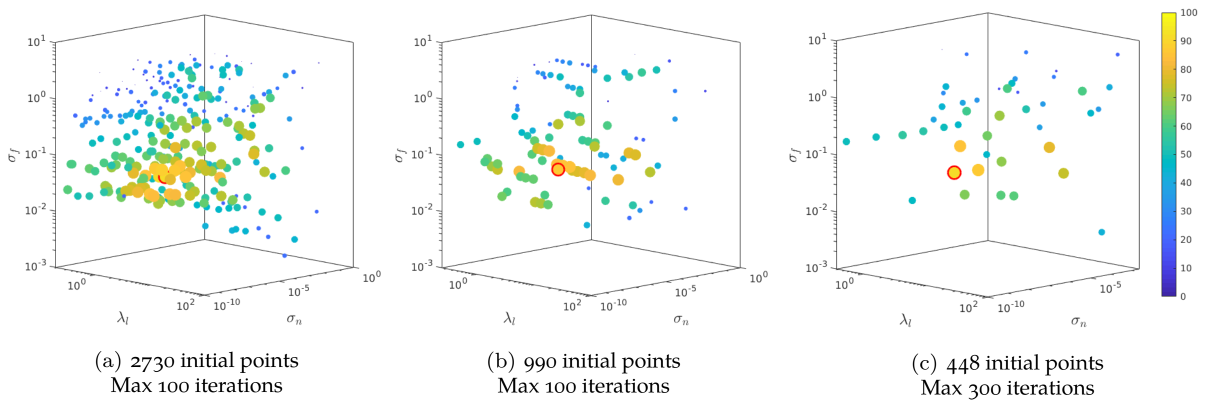
As is expected from a good offline training phase, there is a decreasing trend in both the exact error and the error indicator. As more iterations are made, more observations below the exact error tolerance of 0.5% are made. However, in $N_{it} = 8$, the pessimistic error of the new observation is actually slightly above the tolerance with a value of 0.5074%, and in $N_{it} = 9$ the same observation has a pessimistic error of 0.5022%. It is not until $N_{it} = 10$ that the uncertainty has been lowered such that the pessimistic error of the last three observations is below 0.5%, thus satisfying the convergence criterion.

In $N_{it} = 7$, the predicted mean is actually slightly negative, and in every iteration the confidence intervals go into the negative zone. Obviously, the way the error indicator and the exact error are computed, they can never be negative, but the GPRP method implemented in this thesis does not take that into account. The regression is unconstrained, and we judge that it would take time to try to implement constrained GPR, and in the case of this thesis it turned out to be unnecessary as well because the part of the regression that is interesting for the convergence criterion would likely not be affected significantly by the constraint.

4.3 Parametric Study on Performance

The parametric study mainly focuses on how the performance of the offline training phase is affected when changing the input parameters ε^i, ε^g, and N_c^{add}. However, this section will first take a look at how the hyperparameter optimization depends on the number of maximum iterations N_{GA} versus the number sets of initial values, before comparing the time-wise performance of each of the two methods spend on the main MOR processes, followed by a discussion of the parametric study.

4.3.1 Hyperparameter Optimization



(a) 2730 initial points
Max 100 iterations

(b) 990 initial points
Max 100 iterations

(c) 448 initial points
Max 300 iterations

Figure 4.9: Local and global maxima obtained by the hyperparameter optimization for various settings

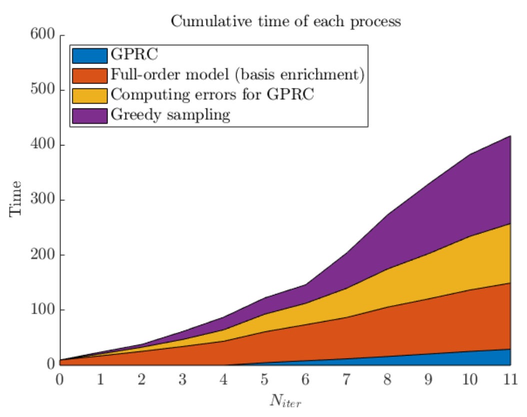
Figure 4.9 shows all the local maxima of the log-likelihood function Eq. (3.38) found by the gradient ascent algorithm. Each dot is a local maximum, and the dot with the red border is the point determined to be global maximum. For illustration purposes, both the size and color of the dots denote the value of the log-likelihood function, where bigger and more yellow denote a higher value. The three axes are the hyperparameters σ_f, σ_n and λ_l. Figure 4.9a was a numerical optimization done with 2730 different starting points and a maximum of 100 iterations. Figure 4.9b was done with 990 points and maximum 100 iterations. Figure 4.9c was done with 448 points and maximum 300 iterations.

From the first and third optimizations, the global maxima obtained were 91.82, while for the second optimization it was 86.83. So to obtain the global maxima it works to use many initial points with not so many maximum iterations, because one of the guesses is likely to be close to the global maximum, and it also works to use fewer initial points but allow for many more iterations so that the guesses don't have to be as close to the global maximum. Either way, the figure shows that an extensive search of the hyperparameter space is needed to find the global maximum, but it is also cheap with such a low number of observations as the matrices in the log-likelihood function are small.

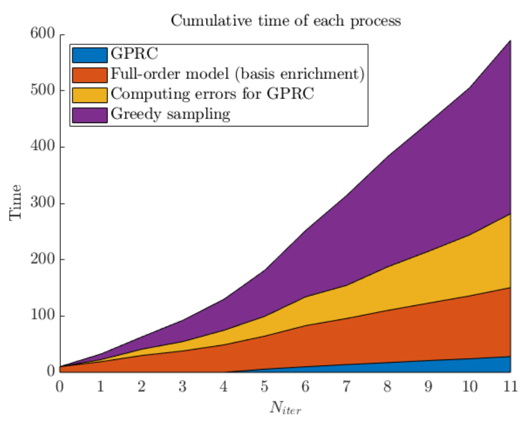
4.3.2 Expense of MOR Processes

Figure 4.10 compares the time-wise performance of the SPS and GPR methods by showing the cumulative time taken by the four main processes of the MOR as the number of iterations increase. The inelastic energy tolerance and global energy tolerance used in both methods are 99.95% and 99.99%, respectively, and the least number of improvements of the GPRP (N_c^{add}) is 2. Both trainings converged at the end of $N_{it} = 11$.

The figure shows that all processes except the greedy sampling take approximately the same amount of time in both methods. The greedy sampling consists of all the ROMs that are run to determine the next training location. For the SPS method, the amount of ROMs that have to be run in total for the greedy sampling up until and including $N_{it} = 11$ is 31, and this number is constant for all energy tolerances because the same SPSs $\widehat{P}_{N_p}$ are being searched every time. For the GPR method, in this case a total of 58



(a) SPS



(b) GPR

Figure 4.10: Cumulative time of each MOR process in the offline phase for SPS and GPR

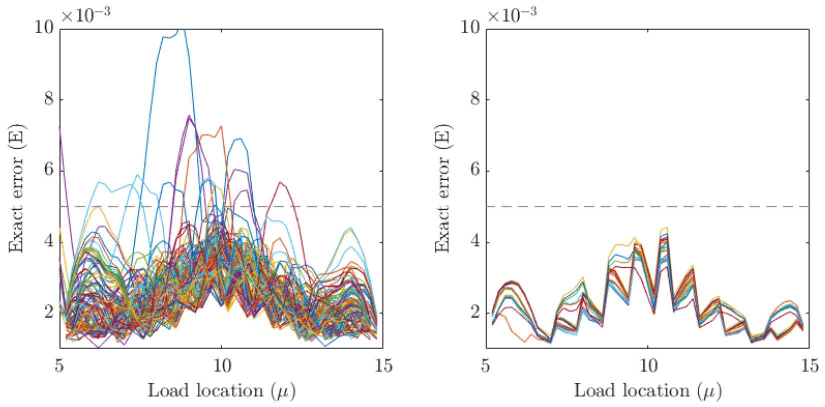
ROMs were run as part of the greedy sampling process, which is what causes this method to take longer. However, while it does take longer, the GPR method selects better loads cases when comparing it with an SPS training phase with the same amount of iterations, as the exact error throughout the beam is on average lower.

Furthermore, the very first load in the GPR method is random and the first three observations in the GPRP are random, so there is quite some uncertainty in this method, causing the time taken to vary considerably. For this specific set of parameters of the GPR method, the minimum time it has taken is the 589 seconds that is shown in Figure 4.10b, and the maximum time is 1167 seconds. The significant variations in time are caused by the uncertainty in the convergence modelling explained in 4.2 Convergence Progression, where it was mentioned that sometimes it could happen that the GPRP leaves out some areas in the beam for a few iterations and instead trains locations that are already well described by the reduced model, and thus yields pessimistic errors that are below the tolerance. But then when it finds this neglected area, the pessimistic exact error could be significantly higher than the tolerance, and the count starts over again, leading to a much longer offline training phase than it has to be.

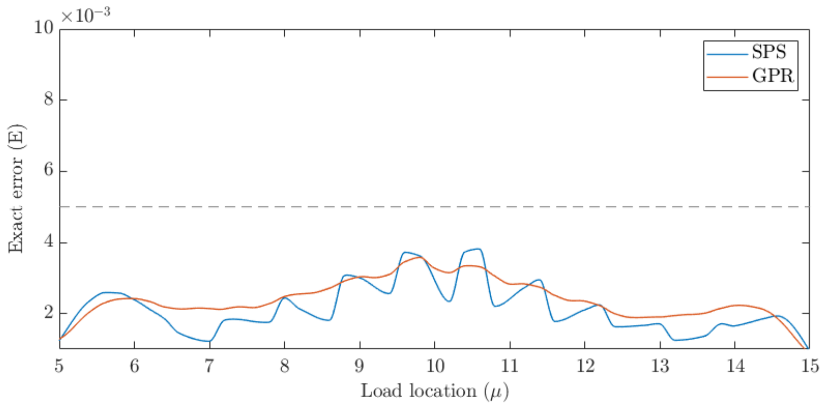
Additionally, the time taken for each reduced model increases with every iteration because the reduced basis matrix becomes bigger. This leads to the proportion of the time taken for the greedy sampling to increase exponentially for the GPR method. For the SPS method the proportion also increases, but not as much because as the SPS is exhausted there are fewer reduced models to run. Though if we move to SPS $\widehat{P}_4$, the time would increase considerably because there are significantly more error indicators to compute, but that is never needed for the 0.5% error tolerance in the convergence criterion. Additionally, the increased size of the reduced basis also causes the computation of error indicators and exact errors for the convergence modelling to take longer.

The SPS method works very well for a beam model when the parameter space is the span of the beam where loads are placed in the downwards direction. This space can easily be subdivided into incrementally finer grids that can be used to train a reduced model to sufficient accuracy very efficiently. However, for other models, such as those mentioned in the literature review, this method will not peform as efficiently. The amount of reduced models that have to be run is the real bottleneck for both methods, and especially for the GPR method. Therefore it is crucial to choose input parameters that minimize the amount of reduced models that have to be run and also that minimizes the size of reduced basis, while at the same time making sure that the GPRP is constructed well and that not too much information is lost in the inelastic and global SVDs. Furthermore, the actual efficiency of the offline and online phases are highly problem-dependent, as it also depends a lot on how much faster the reduced-order model is compared to the full-order model, which also varies significantly from case to case.

4.3.3 Online Accuracy



(a) Exact error distributions for various input parameters for GPR (left) and SPS (right)



(b) Average of the exact error distributions

Figure 4.11: Exact error distributions of ROMs after the offline training phase

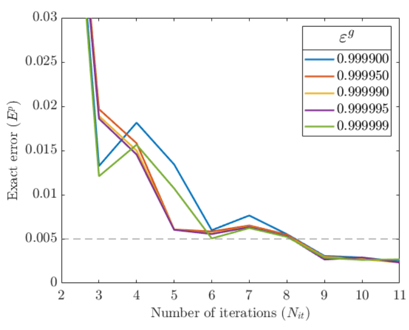
Figure 4.11 shows how the exact errors E^p are distributed throughout the beam for both methods after the offline training phase has converged. The top plots shows the results from using many different sets of input parameters ε^i, ε^g, and N_c^{add}, while the bottom plot shows the average of all the curves for GPR and SPS methods respectively. The dashed gray line is the 0.5% error tolerance that each curve should be below after convergence.

Figure 4.11a shows how the exact error curves from GPR trainings can have many different shapes, which is of course expected due to the nature of how the the parameter space is sampled. For SPS trainings, on the other hand, the shapes are much more similar to each other. This is because the same points are almost always trained, and often in the same order, depending on the energy tolerances. The shape also changes with the number of iterations required for convergence.

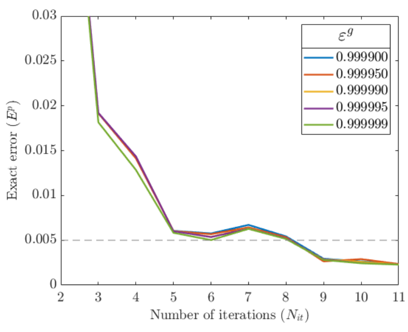
Figure 4.11b shows that the GPR curve is a little more flat than the SPS curve. The reason the SPS curve is more jagged is because the location where those dips are is where the trainings are performed every time, as mentioned before. The GPR curve is more smooth because those load cases can be anywhere. In the areas roughly from $\mu = 6$ to $\mu = 8$ and from $\mu = 12$ to $\mu = 14$, the GPR curve does not follow the same trend because the GPRP sampling method recognizes that fewer trainings are needed to describe this area, which is also seen in Figure 4.7. This results in higher errors in these areas, but still far below the tolerance on average. In the SPS method, the sampling simply exhausts a uniformly distributed grid that becomes incrementally finer, so this method "overtrains" those two areas. At the edges the averages of both go down significantly because those locations are always trained. In the middle, the GPR curve is slightly below the peaks of the SPS curve because the GPR sampling recognizes that the middle part requires more trainings to describe the more significant plastic response due to loads applied there.

Figure 4.11a also shows that for the GPR trainings, some of reduced models are deficient even after convergence, as the exact error is above 0.5% in some parts. So the current convergence criterion has failed to ensure that the goal of the offline training phase was accomplished in every case. The figure shows results from 129 unique sets of input parameters in total for both SPS and GPR, and in 14 of those cases the exact error exceeds the tolerance. The sets of input parameters chosen for the parametric study are those that are most likely to be the most efficient. The type of sets that are expected to and have shown to be efficient are those with either one or two of the input parameters being low, or those with input parameters that are neither low nor high. Those with one or more low input parameters risk converging too quickly. This is because if one of the energy tolerances are too low, then not enough information is kept from each load case, which makes it more likely for the GPRP to select a load location close to a location that is already trained. With a low N_c^{add}, the GPRP is often unable to find the location true maximum error indicator, as shown in Figure 4.5, making it again more likely for the GPRP to select a load location close to an already trained point.

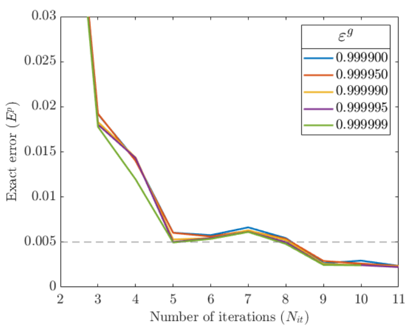
4.3.4 Energy Tolerances and Accuracy



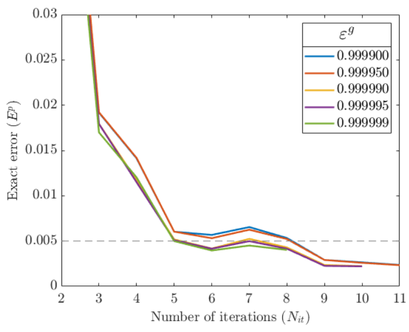
(a) $\varepsilon^i = 99.95\%$



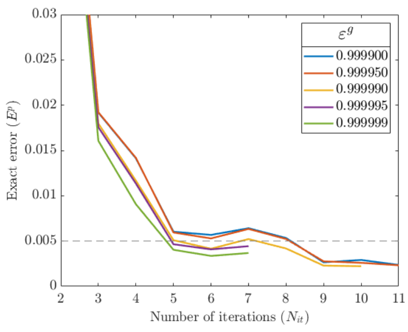
(b) $\varepsilon^i = 99.99\%$



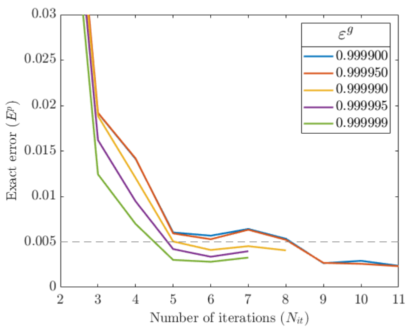
(c) $\varepsilon^i = 99.995\%$



(d) $\varepsilon^i = 99.999\%$



(e) $\varepsilon^i = 99.9995\%$



(f) $\varepsilon^i = 99.9999\%$

Figure 4.12: Effect of energy tolerances on exact error in each iteration

Figure 4.12 shows how the exact error develops during the offline training phase of the SPS method for the all the different combinations of energy tolerances. Each plot has a constant inelastic tolerance ε^i and varying global energy tolerance ε^g. All results shown are from offline training phases that are run until convergence. We only show the effect of the energy tolerance in combination with the SPS method because there is nothing random that occurs in any of the processes of the method, and these results do not depend on a third parameter, which in the case of GPR is the number of improvements of the GPRP. In this way it is easier to see the relationship between the exact error development and the different energy tolerances. The amount of local inelastic and global surrogate snapshots constructed depends not only on the tolerances, but also on the information gathered from the FOM, so their sizes vary in each iteration.

Depending on the load location, the fraction of snapshots that are elastic and inelastic changes. A higher fraction of inelastic snapshots yields a greater number of inelastic surrogate snapshots. Table 4.6 shows a list of how many local inelastic surrogate snapshots are constructed on *average* depending on the inelastic energy tolerance, and also how many global surrogate snapshots are *added* in each iteration depending on the global energy tolerance. The number of global surrogate snapshots added also depend on the number of inelastic surrogate snapshots. If the inelastic SVD yields 3 snapshots, the global one cannot give more than 4 (1 elastic + 3 inelastic) addtional snapshots. Therefore, the values given below are in fact the average maximum number of additional snapshots, given that there are enough elastic and inelastic surrogate snapshots. For example if the SVD performed on the local snapshot matrices yields 5 local surrogate snapshots in total, then a global tolerance of 99.999% will most likely add 5 global surrogate snapshots.

ε^i or ε^g	Local Inelastic Surr. Snapshots	Additional Global Surr. Snapshots
99.95%	3	–
99.99%	4	3
99.995%	5	4
99.999%	8	7
99.9995%	11	10
99.9999%	20	14

Table 4.6: *Average* local and global surrogate snapshots for various energy tolerances

We only show the effect of the energy tolerance in combination with the SPS method because there is nothing random that occurs in any of the processes of the method, and these results do not depend on a third parameter, which in the case of GPR is the number of improvements of the GPRP. In this way it is easier to see the relationship between the exact error development and the different energy tolerances.

The inelastic and global energy tolerances should not go below 99.95% and 99.99%, respectively, because such tolerances greatly affect the quality of the prediction of the GPRP. One of two scenarios tend to occur relatively often. The first scenario is when the GPRP chooses locations that are relatively close to points that are already trained, yielding a low error, and then subsequently choosing a good point that gives a higher error. With low tolerances, this tends to recur, causing significant uncertainty in the GPRC, causing the offline phase to become excessively lengthy as it needs to obtain more cases where the pessimistic error is below the error tolerance in order to satisfy the convergence criterion. This is illustrated in Figure 4.14a. Other times in this scenario, convergence is never achieved. The second scenario that occurs that reduces the performance is that the GPRP keeps choosing points that are close to already trained locations (yielding low errors) enough times to trick the GPRC into thinking that convergence is achieved when in fact the reduced model is severely deficient. The convergence behaviour is illustrated in Figure 4.14b, and Figure 4.6 shows an example where such a load location is selected once.

Furthermore, Figure 4.14a shows that when using the GPRC to compute a mapping of the exact error, an error tolerance lower than $E^{max} = 0.5\%$ could require a high number of iterations to converge. This is because when the ROM becomes more and more accurate, the relative change in $\mathcal{J}^p$ and E^p becomes smaller and they experience considerable fluctuations, especially the exact error as seen in Figure 4.13. While the figure does show that the ROM can be constructed to achieve a lower accuracy than 0.5% in the whole beam, the fluctuations in $\mathcal{J}^p$ and E^p causes a cluster to be formed as seen in Figure 4.14a. This causes issues when it comes to satisfying the convergence criteria for lower error tolerances than 0.5%, resulting in extremely inefficient offline training phases. Therefore, $E^{max} = 0.5\%$ is judged as the lowest error tolerance that is viable for this problem, and it is also judged to be highly accurate.

As shown in Figure 4.11a several of the cases where the global tolerance is below 99.999%, the ROM proves to be deficient. Despite this, we have chosen to also look into those tolerances because the first scenario described in the paragraph above does not occur in these cases, and when the second scenario does occur, the error is often not exceeded significantly. Therefore, the issue with said cases could po-

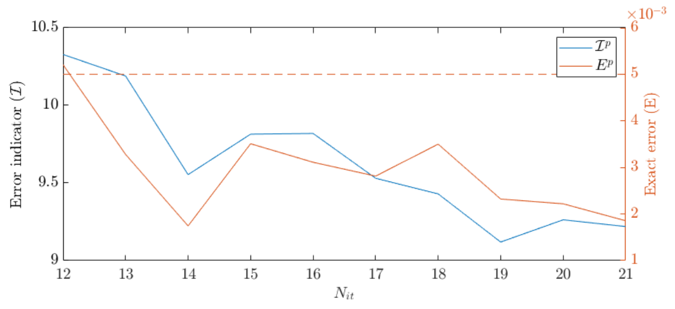
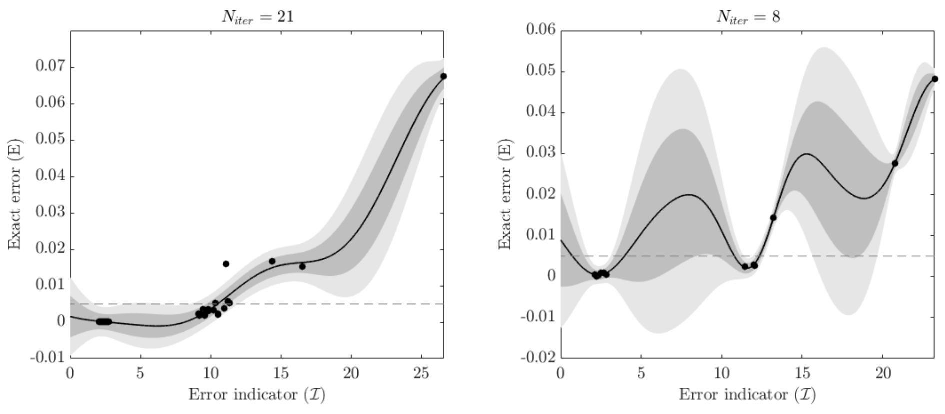


Figure 4.13: E^p and $\mathcal{I}^p$ for last 10 iterations of the case shown in Figure 4.14a

tentially be circumvented by implementing another convergence criterion, as has been mentioned in 4.2 Convergence Progression.

For lower inelastic tolerances there is not that much difference between the curves, given that they choose the same order of the load locations, which is always the case, except in Figure 4.12a for global energy tolerances of 99.99% and 99.9999%. In iteration 8 all the curves meet up again because that is the point when a new SPS begins, so all the same locations are trained at that time, and from that point on they all agree which point in the new SPS has the highest error indicator. When looking at all the curves where the load locations are chosen in the same order, then generally the curves related to high global energy tolerances have lower exact errors as expected, because those trainings keep more of the information from the FOMs. The reason there is not much difference for low inelastic tolerances (especially in Figure 4.12b) is because the number of local inelastic surrogate snapshots is smaller, and therefore there is not much difference in how much information is kept in the global basis for varying tolerances. As more inelastic surrogate snapshots are constructed, the curves vary more because the global SVD does not always have access to as many local surrogate snapshots as it can add to the global surrogate snapshot matrix, as mentioned in the description.

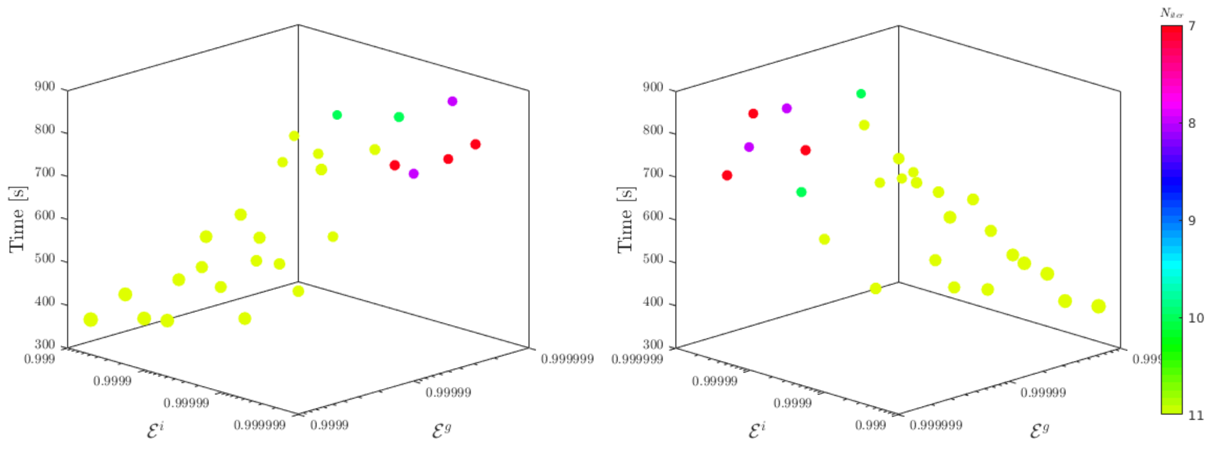
While it is not clearly visible in Figure 4.12c, the training phase with a global tolerance of 99.9999% converged at $N_{it} = 10$, and as the inelastic tolerance keeps increasing, we see more cases where the training phases converge earlier than $N_{it} = 11$. This is of course because enough information is kept from every iteration, so fewer load cases are required. However, fewer iterations does not automatically translate to quicker training time, because more information means more basis vectors. The more basis vectors, the more time the reduced model takes, and the ROM is run many times in the greedy algorithm. The time taken for the different processes in the training phase is discussed in 4.3.2 Expense of MOR Processes.



(a) Lengthy convergence (b) Early convergence

Figure 4.14: Error mapping using GPRC for lengthy and early training phases

4.3.5 SPS Method - Input Parameters and Efficiency



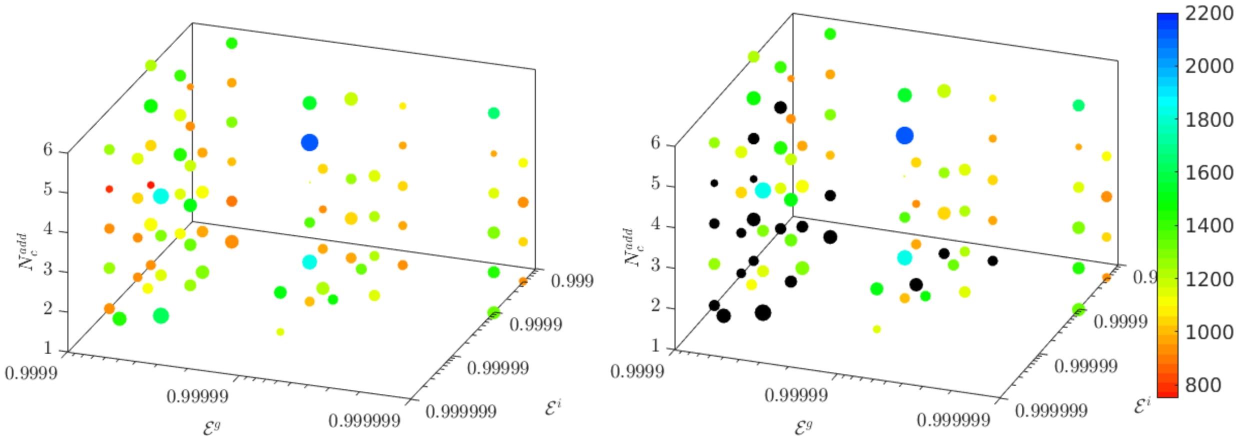
(a) View 1 (b) View 2

Figure 4.15: Total time and number of iterations depending on input parameters for the SPS method

While Figure 4.12 compares how the different tolerances affect the exact error in each iteration, Figure 4.15 gives a comparison of the performance in terms of time taken by the SPS method for various values of the input parameters. The two horizontal axes are the inelastic (ε^i) and global (ε^g) energy tolerances. The vertical axis indicate the total time taken for the offline training phase to reach convergence for a given set of input parameters, and the distinct colors show the amount of training iterations required.

It is clear from the figure that as both tolerances decrease, the time taken for the offline training phase to converge becomes shorter. In general, if one tolerance is kept constant, and the other is increased, the time increases. When both tolerances are very high, however, the number of iterations decreases, which reduces the time, but since the reduced basis is significantly larger in those cases, each iteration takes longer, so it is still more efficient to use low tolerances. The figure shows that the combination of the lowest tolerances gives the best performance with a time of 383 seconds, so it might be possible that even lower tolerances for either the inelastic SVD or the global SVD could yield better results. This is true when decreasing the global tolerance as it requires the same amount of iterations, but the basis is smaller in every iteration, which brings the time down to 308 seconds, though the maximum exact error in the beam is 0.475% close to the middle, so we cannot decrease the tolerance further. On the other hand, when decreasing the inelastic energy tolerance the time almost doubles to 743 seconds. In this case it requires significantly more iterations as the basis is small and the uncertainty in the exact error along the beam increases as described by the GPRC. Nonetheless, the chosen range of tolerances is due to the instability of the GPR method and is explained in the description in 4.3.4 Energy Tolerances and Accuracy, and the range is determined to be the same for both methods.

4.3.6 GPR Method - Input Parameters and Efficiency



(a) Total time and uncertainty

(b) Total time and uncertainty with cases where error tolerance is exceeded blacked out

Figure 4.16: Total time and uncertainty depending on input parameters for the GPR method

The purpose of Figure 4.16 is to show how the three input parameters (ε^i, ε^g, N_c^{add}) affect the performance of the GPR method in terms of time taken to converge, and also the uncertainty in this time. Unlike Figure 4.15, all axes are now the input parameters. The two horizontal axes are the inelastic (ε^i) and global (ε^g) energy tolerances, and the vertical axis is the least number of additional improvements of the GPRP N_c^{add}. For all sets of parameters, the offline training phase is run several times in order to assess the uncertainty in the time-wise performance. Therefore, each dot represents the *maximum* time taken for the offline training phase to converge for a given set of inputs. The color indicates the time, and the radius indicates the uncertainty in the time for a specific input set. Both scatter plots show the same results, but in Figure 4.16b, the colored dots are replaced with black dots where at least one reduced model exceeded the exact error tolerance (see 4.3.3 Online Accuracy).

Compared to Figure 4.15, it is not as clear how the time depends on the input parameters, especially not when considering the cases when the convergence criterion has failed. The reason it is less clear is first of all because of the inclusion of a third input parameter, but it is mostly due to the randomness of the first load location and the initial observations. This causes significant uncertainty for some sets of input parameters. The relationship could become clearer if each and every case is run many times, but this would take a significant amount of time to achieve. However, the figure does show that the uncertainty is low for those cases that are efficient (orange dots are often smaller than the green and blue), and those cases have also been run more times than those that take longer time to establish how well they actually perform. To reduce the uncertainty, the intial load of the training phase and the initial observations in each GPRP could have been chosen better. For example the initial load could always be placed in the middle, and the initial observations could be done in a uniformly spaced grid along the parameter space or in areas where there have been fewer trainings. However, we have not chosen to do this due to the fact that such choices of initial load and observations cannot be made for many other high-fidelity models, so in order to keep the implementation more general, they were kept random.

Figure 4.16a does show that the GPR method converges quickly for cases when the global energy tolerance is low, and especially when the least number of improvements is low. However, it is also these most efficient cases where the convergence criterion fails most often. There are also many cases where $N_c^{add} = 1$ or when the inelastic tolerance is low, for which the GPR method converges quickly, but for some of those the convergence criterion also fails. With the current results, Figure 4.16b shows that the convergence criterion fails often when $\varepsilon^g < 0.99999$ and/or when $N_c^{add} = 1$. As mentioned in 4.2 Convergence Progression, this issue of the failing convergence criterion could be circumvented by changing the criterion or making it more conservative.

When ignoring the unstable zones mentioned in the previous paragraph, Figure 4.16b does show that for large N_c^{add}, the convergence takes long, especially when combined with high tolerances. It also shows that the GPR method performs well when using low inelastic tolerances. Beyond that, it is not clear which values of the parameters yield the best performing reduced model.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The main objective of this thesis was to investigate methods to reduce the computation time of finite element analyses that involve high-fidelity mechanical models by implementing an automatic and efficient procedure to sample the parameter space for load cases. This thesis built upon some of the acceleration techniques implemented by [26] to minimize the computational cost of a multiscale/multiphysics model, and the existing MOR framework from this work was expanded to facilitate the iterative greedy sampling of the parameter space of a beam model. Following the computational implementation, the behaviour and performance of both methods were assessed in a parametric study that varied the three input parameters ε^i, ε^g and N_c^{add}. The results have been used to obtain a better understanding of the behaviour and performance of the greedy sampling process of a mechanical model, and also a better understanding of issues and pitfalls that have to be tackled when attempting to construct a useful reduced basis with many load cases and automatize the sampling. The findings of the research are summarized below as answers to the research questions.

1. *What is the best way to sample the parameter space so as to minimize the number of training cases?*
   Two methods were developed to sample the parameter space. The first was the SPS method that performs greedy sampling of incrementally finer grids of points along the beam. The second method was the GPR method that also performs greedy sampling, but the points are selected using Gaussian Process Regression to predict which location along the beam would yield the highest error. As the research progressed, it turned out that it was not about the number of training cases, but about how the training cases were selected, and how much information was stored from each case. The amount of information kept in each case also increases the time taken by the greedy sampling, and it was clearly shown for the SPS method that it is significantly more efficient to store less information in each training iteration, and instead run more training cases to collect smaller amounts of information from more locations.

   For the beam model used in this thesis, the SPS method is more efficient with the quickest training taking only 383 seconds. For the GPR method when only considering cases where the convergence criterion did not fail, the lowest maximum time was 920 seconds, and the lowest minimum time was 436 seconds. Even for the input parameters of the GPR where the exact error is always below the tolerance in the whole beam, the minimum time is still higher than the most efficient SPS. However, it was shown that the GPRP recognized areas in the beam that required fewer load cases to approximate the behaviour, which the SPS method did not take into account. Despite this, greedy sampling using GPRP is not efficient for this specific problem because of the significant number of ROMs that have be run in order to compute an accurate prediction of the location with the highest error. For the beam model, the SPS works extremely well because the parameter space can easily be split up into grids, despite the points in the grid being suboptimal.

   For other high-fidelity models such as those discussed in 2.1 High-Fidelity Models, the parameter space cannot easily be subdivided into incrementally finer grids that we can sample. This is because those parameter spaces are often potentially infinitely dimensional, so any sort of grid in this space would also have to be of high order. Therefore, the exhaustive sampling of such a parameter space would be prohibitively expensive using a method similar to the SPS method explored in this thesis. On the other hand, we have seen that the GPRP can accurately predict the location in the parameter space where the error is the highest and would thus add the most new information to the system. Of course, the computation cost would increase significantly with the additional dimensions because many more observations would be required to compute a reliable regression. Additionally, the computation of the FOM solutions themselves from which the snapshots are collected would also in general be considerably more expensive. Using a uniform sampling method, we would have to compute many expensive FOM solutions, several of which will be relatively obsolete, but using GPR we would ideally only compute the necessary FOMs to make the ROM as accurate as desired. Additionally, it was discussed previously that for the SPS method it was more efficient to run more training cases and store less information in each iteration, but this does not necessarily hold if the cost of computing the FOM solution is significantly more expensive. It was found that the bottleneck for efficiency for the beam model was the amount of ROM solutions that had to be computed for the greedy sampling, but for other high-fidelity models an expensive FOM could be the bottleneck instead.

To summarize, the SPS method is efficient when the FOM solution is cheap and the parameter space can be subdivided into low-order grids that can be exhaustively sampled, but the GPR method shows great potential when it comes to sampling a high-order parameter space that cannot be discretized easily.

2. *Can training be performed in an optimized way by a heuristic algorithm that takes the maximum tolerable loss of accuracy as input?*
In order to investigate how the offline training phase can be performed in the most optimal way, a parametric study was performed where three input parameters were varied. For each set of parameters, the training was run until the convergence criterion judged that the ROM was accurate enough to yield an exact error lower than $E^{max} = 0.5\%$ in the entire beam. This tolerance is considered to be highly accurate and the exact error has been shown to reach a plateau at around $0.2-0.3\%$.

If a user desires a certain accuracy defined by the exact error in the entire beam, all that has to be done is to adjust the convergence criterion accordingly, and choose the set of input parameters that would result in the most efficient training phase. Ideally, these input parameters would be selected by the program, but that is not implemented in this thesis. For the SPS method there is no randomness involved, and the most efficient parameters can easily be chosen. For the GPR method, on the other hand, there is not a clear relationship between the input parameters and the efficiency, but the parametric study did show that a low ε^i is efficient and generally robust in terms of convergence, but ε^g and N_c^{add} must be chosen with more care. If the two latter values are too low, then the chosen convergence criterion can fail to grant the desired accuracy. However, in general, the input parameters should be kept as low as possible while still ensuring that the accuracy is achieved. Therefore, the current work employs a heuristic algorithm that takes not only the desired accuracy as input, but also the other input parameters mentioned.

When comparing the two methods' ability to reach a certain accuracy efficiently, the SPS would likely work best for $E^{max} \geq 0.5\%$. If E^{max} is chosen to be lower than this, the SPS method might risk having to move to the next SPS, which would occur if 13 iterations are needed, but for a 0.5% tolerance, only 11 are needed with the most efficient input parameters. Moving to an even finer SPS means that the ROM has to be computed in a significant number of points to perform the greedy sampling. Therefore, if very high accuracy is desired, the GPR method could be more efficient as the number of ROM computations required does not really change with more iterations.

The specific input parameters required and their viable range is highly problem-dependent. For other high-fidelity models, the energy tolerances ε^i and ε^g will likely be around the same range, but an elastic energy tolerance ε^e might have to be introduced. The least number of additional improvements to the GPRP N_c^{add} will be considerably higher for parameter spaces of higher dimensions than the one-dimensional span of a simply supported beam. Additionally, other high-fidelity models might include more or other steps in the construction of the ROM, which again highlights how problem-dependent MOR is. Therefore, for other high-fidelity models, a parametric study of the performance with varying input parameters has to be performed to determine how most efficiently achieve a desired accuracy.

5.2 Recommendations

This section presents recommendations for future work to be done on the topic of model-order reduction with automatic and efficient parameter sampling, as well as improvements that can be made to the current work.

Alternative High-Fidelity Models
The purpose of investigating methods to efficiently and accurately construct a reduced order model for the beam problem in this thesis, is to use the knowledge and experience as a stepping stone to implementing similar sampling methods for other high-fidelity models. For the beam problem, it is highly inefficient to construct a reduced-order model because of how low order the full-order model actually is, but for the RVE problem for instance, efficient and automatic sampling methods would be extremely useful for constructing a ROM. Therefore, the GPR method should be adjusted and implemented for e.g. the RVE problem in Rocha [26] to assess the suitability of the GPR to greedily sample a high-dimensional parameter space.

Hyper+reduction
For the beam problem presented in this thesis, the bottleneck of both the SPS and the GPR methods was the number of ROMs that had to be run during the greedy sampling process. For the SPS, this number was predetermined based on the chosen grids, but for the GPR it fluctuated, which caused the efficiency to be oscillatory and unpredictable. Hyper+reduction methods such as ECM should be implemented in order to alleviate this issue. This would drastically reduce the time required to compute the ROM solutions that are used for both GPRP and GPRC. However, ECM changes the way the internal force in the beam is computed, meaning that the error indicator will likely have to be computed in a different way that takes this into account.

Error Indicator and Exact Error
In this work, it has been repeated that the error indicator $\mathcal{J}$ gives an indication of what the exact error E of the ROM is. This implies that the two should be computed in a consistent manner, but that is unfortunately not the case in this thesis. The goal of the training phase is to construct a ROM that approximates the FOM in the whole domain of the mode, which is why the error indicator and exact errors are computed from the residual and the displacement field in the whole beam, respectively, and both take into account the whole time-history. However, there are some inconsistencies in how $\mathcal{J}$ and E are computed as shown in Eq. (2.20) and Eq. (2.30), respectively. In the former, a square root is taken of the entire term, which is not done in the latter. Additionally, in the computation of E, the exact error is normalized by the stiffness-projected dot product of the FOM displacements in the denominator, but no such normalization is done in the computation of $\mathcal{J}$. Several other versions of both formulas were tried that were most consistent with each other, but it was decided to leave this inconsistency because the GPR method performed better in terms of accuracy and efficiency using the formulas presented. It is recommended to look into alternative ways of computing $\mathcal{J}$ and/or E that are consistent with each other.

GPRP Improvement Stopping Criterion
Currently, the GPRP improvement is stopped when the following condition is satisfied:

$$\left(\tilde{\mathcal{J}}^{max} - \tilde{\mathcal{J}}^{min}\right)\beta > \max\left(\tilde{\mathcal{J}}(\mu^\star) + \alpha_{\mathcal{J}}\tilde{s}_{\mathcal{J}}(\mu^\star)\right) \tag{5.1}$$

This is not a very clean and robust method of determining when the GPRP improvement should be stopped, and is based simply on observations made by the author of the behaviour of the GPRP. Instead, it would be significantly cleaner and more general to stop the improvement when the probability of improving the current maximum of the GPRP by the target factors in $\boldsymbol{\tau}$ is below a certain value. This value should be chosen with care because it could cause the improvement to stop too early before an accurate maximum is predicted. Potentially, this could be an input parameter that should be included in the parametric study.

Sampling Convergence Criterion
It has been shown in this thesis that the convergence criterion sometimes failed to guarantee the desired accuracy. Therefore, the criterion should be adjusted accordingly to guarantee this. It could e.g. be adjusted simply by changing the number of consecutive iterations with a pessimistic errors $\hat{E}^{+}$ below the tolerance, or by ignoring the *consecutive* part, and allowing it to convergence simply when a certain number of pessimistic errors are below the tolerance. Another method could be to simply abandon the GPRC, and determine convergence based on the exact errors E^{p}. In any case, this is an important issue that must be investigated further.

Comprehensive Parametric Study
The parametric study in this thesis was not as comprehensive as it should have been in order to obtain a better understanding of how the input parameters affect the efficiency and accuracy of the ROM. However, the results of such a parametric study is highly problem-dependent, so for a toy model like the simply supported beam, the knowledge that can be gained from such a study is limited. In any case, for a parametric study performed for any high-fidelity mechanical model, *all* combinations of parameters should be used, even those that are likely to be inefficient because even those cases can help the understanding. The study must also check how the relationships between the input parameters and efficiency are affected when adjusting the error tolerance. It could be that e.g. a lower ε^{i} is favoured over a lower ε^{g} when a lower accuracy is desired, or the other way. Lastly, it is important to run training phases many times for each set of parameters to assess the uncertainty in efficiency. It could be that one set of parameters yields the most efficient training, but that would be relatively useless if there is significant variability. Furthermore, a better understanding of the uncertainties could help to adjust steps in the greedy sampling procedure to lower these uncertainties.

Automatic and Adaptive Input Parameter Selection
When a comprehensive parametric study has been performed for a high-fidelity model, there should be a good understanding of how the various input parameters affect the efficiency and accuracy of the MOR. Based on this understanding, the training algorithm could be expanded to adaptively choose the most efficient input parameters based on the desired accuracy defined by the user. E.g. as the global surrogate matrix becomes larger, it could be that the energy tolerance should be adapted in order to not lose too much information in the SVD. Another example could be that N_{c}^{add} should be increased as more training cases are added because the distribution of the error indicator becomes more irregular, making it more difficult for the GPRP to accurately predict the maximum.

Alternative prediction methods
In the world of machine learning, there are many ways to perform predictions based on observations of various types of data. It has been shown that the GPR makes accurate predictions of the maximum error indicator for the beam model, given that enough observations are provided. However, it could be that alternative prediction methods such as Artificial Neural Networks (ANN) or Support Vector Machines (SVM) perform better predictions with less data. Additionally, for other high-fidelity models with parameter spaces that yield observations in higher dimensions, the nature of the data is vastly different, and the GPR method might not perform well anymore, so it would be worthwhile to look into alternative prediction methods and assess their suitability to the problem.

BIBLIOGRAPHY

[1] M.A. Bessa and S. Pellegrino. Design of ultra-thin shell structures in the stochastic post-buckling range using bayesian machine learning and optimization. *International journal of Solids and Structures*, 139-140:174–188, 2018.

[2] A. Bhattacharya, D. Pati, and D. Dunson. Anisotropic function estimation using multi-bandwidth gaussian processes. *The Annals of Statistics*, 42:352–381, 2014.

[3] H.J. Bijl. *LQG and gaussian process techniques - for fixed-structure wind turbine control*. PhD thesis, Technische Universiteit Delft, October 2018.

[4] S. Chaturantabut and D.C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32:2737–2764, 2010.

[5] M. Drohmann and K. Carlberg. The romes method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification*, 3:116–145, 2015.

[6] D.K. Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, June 2014.

[7] M. Ebden. Gaussian processes for regression: A quick introduction, 2008. Accessed at: arxiv.org/abs/1505.02965.

[8] M. Frangos, Y. Marzouk, K. Willcox, and B. van Bloemen Waanders. Surrogate and reduced-order modeling: a comparison of approaches for large-scale statistical inverse problems. In *Large-scale inverse problem and quantification of uncertainty*. John Wiley Sons, Ltd., 2010.

[9] F. Ghavamian, P. Tiso, and A. Simone. Pod-deim model order reduction for strain-softening viscoplasticity. *Computer Methods in Applied Mechanics and Engineering*, 317:458–479, 2017.

[10] O. Goury, D. Amsallem, S.P.A. Bordas, W.K. Liu, and P. Kerfriden. Automatised selection of load paths to construct reduced-order models in computational damage micromechanics: from dissipation-driven random selection to bayesian optimization. *Computational Mechanics*, 58:213–234, 2016.

[11] J.A. Hernández, J. Oliver, A.E. Huespe, M.A. Caicedo, and J.C. Cante. High-performance model reduction techniques in computational multiscale homogenization. *Computer Methods in Applied Mechanics and Engineering*, 276:149–189, 2014.

[12] J.A. Hernández, M.A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer Methods in Applied Mechanics and Engineering*, 313:687–722, 2016.

[13] M.I. Heywood, J. McDermott, M. Castelli, E. Costa, and K. Sim. *Genetic programming: 19th European conference, EuroGP 2016 Porto, Portugal, march 30 – April 1, 2016 proceedings*, page 155. Springer-Verlag, April 2016.

[14] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[15] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:342–383, 2001.

[16] I. Kalashnikova, S. Arunajatesan, M.F. Barone, B.G. van Bloeme Waanders, and J.A. Fike. Reduced order modeling for prediction and control of large-scale systems, 2014.

[17] D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR 2015*, 2015.

[18] G.P. Nikishkov. Lecture notes: Introduction to the finite element method, 2004.

[19] J. Oliver, M. Caicedo, A.E. Huespe, J.A. Hernández, and E. Roubin. Reduced order modeling strategies for computational multiscale fracture. *Computer Methods in Applied Mechanics and Engineering*, 313:560–595, 2017.

[20] G.M. Oxberry, T. Kostova-Vassilevska, W. Arrighi, and K. Chand. Limited-memory adaptive snapshot selection for proper orthogonal decomposition. *International journal for Numerical Methods in Engineering*, 109:198–217, 2017.

[21] A. Paul-Dubois-Taine and D. Amsallem. An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *International journal for Numerical Methods in Engineering*, 102:1262–1292, 2015.

[22] K. Pearson. L3. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and journal of Science*, 2:559–572, 1901.

[23] R.W. Preisendorfer, C.D. Mobley, and T.P. Barnett. The principal discriminant method of prediction: Theory and evaluation. *Journal of Geophysical research*, 93:10815–10830, 1988.

[24] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[25] S.J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *Proceedings of ICLR 2018*, 2018.

[26] I.B.C.M Rocha. *Numerical and Experimental Investigation of Hygrothermal Aging in Laminated Composites*. PhD thesis, Technische Universiteit Delft, January 2019.

[27] I.B.C.M. Rocha, F.P. van der Meer, R.P.L. Nijssen, and L.J. Sluys. A combined experimental/numerical investigation on hygrothermal ageing of fiber-reinforced composites. *International journal for Numerical Methods in Engineering*, 73:407–419, 2019.

[28] I.B.C.M. Rocha, F.P. van der Meer, and L.J. Sluys. Efficient micromechanical analysis of fiber-reinforced composites subjected to cyclic loading through time homogenization and reduced-order modeling. *Computer Methods in Applied Mechanics and Engineering*, 345:644–670, 2019.

[29] S.S. Sapatnekar. Overcoming variations in nanometer-scale technologies. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1:5–18, 2011.

[30] W.H.A. Schilders, H.A. van der Vorst, and J. Rommes. *Model order reduction: theory, research aspects and applications*. Springer, 2008.

[31] L. Sirovich. Turbulence and the dynamics of coherent structures. part i: coherent structures. *Quarterly of Applied Mathematics*, 45:561–571, 1987.

[32] E.L. Snelson. *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, University College London, 2007.

[33] P.A. Sparks. *Reduced order homogenization models for failure of heterogeneous materials*. PhD thesis, Vanderbilt University, May 2015.

[34] R.A. van Tuijl, J.J.C. Remmers, and M.G.D. Geers. Integration efficiency for model reduction in micro-mechanical analyses. *Computational Mechanics*, 62:151–169, 2018.