Accuracy Control and Nonintrusive Implementation of Adaptive Reduced Order Modelling Based on Greedy Sampling for Elasto-dynamics

Xiaohan Du

Supervisors: Dr. Pierre Kerfriden
Dr. Abhishek Kundu

A thesis submitted to the graduate school
in fulfilment of the requirements for the degree of
Doctor of Philosophy

Advanced Materials and Computational Mechanics Group
Cardiff School of Engineering

Cardiff, Wales, United Kingdom

December 11, 2018

Summary

Parametric problems have been widely studied and many researches have been provided to reduce the cost of computations. Reduced order modelling (ROM) achieves this goal by performing and storing a sequence of pre-computations in an expensive "*offline*" stage, and utilises the stored data to make predictions of solutions for parametric problems in an "*online*" stage with low cost. The (POD -) Greedy sampling algorithm is a powerful tool to obtain those pre-computations in an optimal sense.

Problems arise for conventional reduced order modelling when the system undergoes dynamic changes: first of all, a robust error estimate is needed for dynamic problems; moreover, a cost-effective procedure is required in the "*offline*" stage to generate the optimum set of sample points, such that the most representative reduced basis may be obtained, which would also keep the "*offline*" cost under control.

In this thesis, a new POD-Greedy sampling algorithm which utilises a new error indicator will be presented. This error indicator aims to predict paths of the optimum maximum error convergence. The standard POD-Greedy approach requires exact solutions over the entire parameter domain when *a-posteriori* error estimate is not available, thus is not practical. Instead, the proposed POD-Greedy algorithm avoids computations of the massive number of exact computations by applying interpolation, so that the numerical efficiency can be improved. Another contribution is an "*error in the error*" indicator which drives the local adaptivity of interpolation sample grids. This indicator compares low and high order interpolation scheme to obtain the correct sequence of local $h-$ refinement and Greedy iterations. Finally a nonintrusive Abaqus/Matlab code coupling technique will be presented in appendix to enable seamless integration of commercial software and Matlab source code in computations of exact solutions. The accuracy and feasibility of the proposed method will be experimented on varieties of

cases.

Declaration

DECLARATION

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed..... (candidate) Date:.....

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed..... (candidate) Date:.....

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated, and the thesis has not been edited by a third party beyond what is permitted by Cardiff University's Policy on the Use of Third Party Editors by Research Degree Students. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed..... (candidate) Date:.....

DECLARATION

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed..... (candidate) Date:.....

Acknowledgements

I would like to acknowledge my supervisors Dr. Pierre Kerfriden and Dr. Abhishek Kundu for their support, knowledge and guidance during my PhD. With their help I learned how to do research, established skills of programming, and most importantly, how to study and understand new knowledge.

I would like to thank all my friends in the research offices of Cardiff University. The spiritual support and the time we spent together helped me to relax and kept me motivated.

I would also like to acknowledge all my friends in Dragon CrossFit. CrossFit is an important part of my life, which keeps me physically and mentally strong.

Finally I would like to express my gratitude to my family for their love, trust and support: PhD is a long and difficult exploration, without them I would not be able to march through the darkness and pursue my dreams. I dedicate this work specially to them.

Contents

Nomenclature

Chapter 1

- μ: parameter value.
- ρ: density.
- E: Young's modulus.
- $\mathcal{N}$: dimension of the finite element space.
- N: dimension of the reduced space.
- $\mathcal{P}$: parameter domain.
- Ω: physical domain.
- X: space of exact solutions, Hilbert space.
- M: mass matrix.
- C: damping matrix.
- K: stiffness matrix.
- F: force matrix.
- $\ddot{\boldsymbol{U}}(t;\mu)$, $\dot{\boldsymbol{U}}(t;\mu)$, $\boldsymbol{U}(t;\mu) \in X$: finite element responses, the 'exact' solution.

- $\boldsymbol{u}(\mu_N)$: the 'snapshot' at parametric value μ_N.
- $\boldsymbol{e}(\mu)$: the RB error.

Chapter 2

- N_μ: number of discretised parametric values to be evaluated.
- a and b: damping coefficients.
- $\ddot{\boldsymbol{U}}^{\mathrm{r}}(\mu)$, $\dot{\boldsymbol{U}}^{\mathrm{r}}(\mu)$, $\boldsymbol{U}^{\mathrm{r}}(\mu)$: solutions from a reduced static problem.
- $\underline{\phi}_i$: i^{th} reduced basis vector.
- $\boldsymbol{\Phi}$: the reduced basis.
- δ_{ij}: the *Kronecker delta* symbol.
- $\mathcal{P}_N = \{\mu_1, \ldots, \mu_N\} \subset \mathcal{P}$: the snapshot parameter set.
- X_N: space of reduced solutions.
- $\ddot{\boldsymbol{u}}_N(\mu)$, $\dot{\boldsymbol{u}}_N(\mu)$, $\boldsymbol{u}_N(\mu) \in X_N$: reduced solutions lie in the reduced space.
- $\mathbf{K}_N$, $\mathbf{f}_N$, $\mathbf{l}_N$: system matrices projected on reduced space.
- $\boldsymbol{e}$: the exact (or RB) error.
- $\boldsymbol{R}(\mu)$: residual for a static problem.
- $\boldsymbol{R}(t;\mu)$: residual for a dynamic problem.
- Z: snapshot matrix, the collection of static solutions.
- $\mathcal{P}^{\mathrm{train}}$: the training set, $\mathcal{P}^{\mathrm{train}} \subset \mathcal{P}$.
- N_{train}: number of discretised point in training set.
- $\boldsymbol{e}_{\mathrm{ind}}$: error indicator.

- $\boldsymbol{e}_{\mathrm{proj}}$: projection error.
- $\mathcal{P}^{M}$: the magic point set.
- N^{add}: number of newly added reduced basis vectors.
- $e_{\mathrm{r}}^{\mathrm{to}}$: error reduction threshold for the obtained magic point.

Chapter 3

- $\ddot{\boldsymbol{U}}^{\mathrm{h}}(t)$, $\dot{\boldsymbol{U}}^{\mathrm{h}}(t)$, $\boldsymbol{U}^{\mathrm{h}}(t)$: space-time FE response.
- $\underline{\ddot{\boldsymbol{u}}}_n^{\mathrm{h}}$, $\underline{\dot{\boldsymbol{u}}}_n^{\mathrm{h}}$, $\underline{\boldsymbol{u}}_n^{\mathrm{h}}$: FE response vector at time step n.
- $\boldsymbol{F}^{\mathrm{h}}(t)$: FE force matrix.
- $\underline{\boldsymbol{f}}_n^{\mathrm{h}}$: FE force vector at time step n.
- $\underline{\boldsymbol{x}}_n^{\mathrm{h}}$: space and time FE response vector at time step n.
- $\underline{\boldsymbol{X}}^{\mathrm{h}}$: the *full response vector*.
- $a_0 - a_5$: Newmark coefficients.
- $\mathbf{I}$: identity matrix.
- $\boldsymbol{H}^s, \widetilde{\boldsymbol{H}}^s$: diagonal entries of $\boldsymbol{A}$.
- $\boldsymbol{H}^f$: off-diagonal entries of $\boldsymbol{A}$.
- $\underline{\boldsymbol{g}}_n^{\mathrm{h}}$: force vector at time step n.
- $\underline{\boldsymbol{G}}^{\mathrm{h}}$: the *full force vector*.
- $\boldsymbol{A}$: the *dynamic operator*.
- $\underline{\boldsymbol{g}}_n^{\mathrm{r}}$: approximated force vector at time step n.
- $\boldsymbol{\psi}$: diagonal entries of $\boldsymbol{\Psi}$.

- $\underline{\boldsymbol{\pi}}_n$: reduced variable vector at time step n.
- $\boldsymbol{\Psi}$: the *assembled reduced basis matrix*.
- $\underline{\boldsymbol{\Pi}}$: the *full reduced variable vector*.
- $\underline{\boldsymbol{G}}^{\mathrm{r}}$: the *approximated full force vector*.

Chapter 4

- $\|\cdot\|_{\mathrm{F}}$: Frobenius norm of $\cdot$.
- $\underline{\boldsymbol{R}}(\mu)$: the *full residual vector*.
- $\underline{\boldsymbol{E}}(\mu)$: the *exact full error vector*.
- $\boldsymbol{A}^{-1}$: the *dynamic operator* inverse.
- $\widehat{\boldsymbol{A}^{-1}}(\mu)$: the approximated *dynamic operator* inverse.
- $l_i(\mu)$: the interpolation polynomial.
- $\mathcal{P}^i$: the interpolation sample domain, $\mathcal{P}^i \subset \mathcal{P}^{\text{train}} \subset \mathcal{P}$.
- $\boldsymbol{A}_i^{-1}$: the *dynamic operator* inverse at interpolation sample μ_i.
- $\mathsf{Term}_i^{\mathrm{h}}$ and $\mathsf{Term}_{ij\mathfrak{r}}^{\mathrm{r}}$: terms being used when solving approximated full error vector.
- $\widehat{\underline{\boldsymbol{E}}}(\mu)$: the approximated *exact full error vector*.
- $\boldsymbol{\Psi}_{\mathrm{r}}$: the *assembled reduced basis matrix* contains the $\mathfrak{r}^{\text{th}}$ basis vectors.
- $\underline{\boldsymbol{\Pi}}_{j\mathfrak{r}}(\mu)$: the *full reduced variable vector* associated with the $\mathfrak{r}^{\text{th}}$ basis vector and j^{th} affined term.
- $\boldsymbol{G}_{j\mathfrak{r}}^{imp}$: the *full impulse matrix*.
- $\boldsymbol{F}_{j\mathfrak{r}}^{imp,m}$, $\boldsymbol{F}_{j\mathfrak{r}}^{imp,c}$, $\boldsymbol{F}_{j\mathfrak{r}}^{imp,k}$: the impulses associated with mass, damping, stiffness matrices, respectively.

- $\underline{\boldsymbol{X}}_i^{\mathrm{h}}$: the *full response vector* obtained by applying $\underline{\boldsymbol{G}}^{\mathrm{h}}$.
- $\underline{\boldsymbol{X}}_{ij\mathfrak{r}}^{\mathrm{r}}$: the *full response vector* associated with the i^{th} interpolation sample, $\mathfrak{r}^{\mathrm{th}}$ basis vector and j^{th} affined term.
- $\boldsymbol{U}_{ij\mathfrak{r}}^{imp,m}$, $\boldsymbol{U}_{ij\mathfrak{r}}^{imp,c}$, $\boldsymbol{U}_{ij\mathfrak{r}}^{imp,k}$: impulse responses correspond to mass, damping, stiffness matrices, respectively. Displacement components of $\underline{\boldsymbol{X}}_{ij\mathfrak{r}}^{\mathrm{r}}$.
- $\boldsymbol{U}_i^{\mathrm{h}}$: displacement component of $\underline{\boldsymbol{X}}_i^{\mathrm{h}}$.
- $\mathbf{a}_{j\mathfrak{r}}$: the general form of reduced variables associated with acceleration, velocity and displacement.
- δ_{tn}: the *Kronecker delta* symbol.
- $\mathrm{tr}(\cdot)$: trace of $\cdot$.
- M_i: the matrix which contains vectors $\underline{\boldsymbol{U}}_i^{\mathrm{h}}$ and $\underline{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp}$.
- $\mathsf{M}_i^{\mathbf{trans}}$: the *displacement vector product matrix*.
- $\underline{\mathbf{a}}$: vector of affine coefficients and reduced variables.
- V_L, V_R, Σ: left singular vectors, right singular vectors and singular value matrix of impulse responses.
- $\mathcal{W}_{\boldsymbol{\alpha}}$: the *reduced variable vector matrix*, which is a collection of reduced variable vectors.
- $\mathsf{M}^{\mathbf{trans},\mathrm{r}}$: subspace projection of $\mathsf{M}_i^{\mathbf{trans}}$.

Chapter 5

- $\hat{\mathcal{P}}^i$, $\hat{\hat{\mathcal{P}}}^i$: the 'slave' and 'master' interpolation sample domain.
- $\hat{N}_b$, $\hat{\hat{N}}_b$: number of blocks partitioned by 'slave' and 'master' vertices.
- $\boldsymbol{e}^e$: the "*error in the error*" *indicator*.

Chapter 1

Introduction

1.1 Motivation

Modern engineering problems often require the simulation of structural behaviours which may compose multiple parts and various materials, and/or encounters different conditions. For example, pressure and temperature under different loading conditions (amplitude, frequency, location, etc); surgical simulations and computer games which requires real-time outputs; optimal control problems, etc. Analytical solutions are usually not available in these cases, therefore instead of executing complex and expensive physical experiments, computational simulations may be used to reduce the costs. The aforementioned types of problems are similar in the aspect that the simulations may need to be performed for many times, which can be time-consuming and expensive. Simulation methods can be varied, such as Finite Element (FE) Methods, Finite Volume (FV) Methods, Finite Difference Method, Finite Strip Method, etc. The complexity of the simulations depends on a large number of factors. While physical experiments are necessary for calibrating models, computer simulations show strong advantages over physical experiments in terms of cost and accuracy under certain conditions. For example, design of modern high-performance aircraft often faces challenges in the transonic

regime. Replacing the manufacture of wind tunnel flutter model by computational fluid dynamics (CFD) simulations is an efficient approach. The construction and analysis of a wind tunnel flutter model could cost over a year's time, and the nonlinear CFD model with 168799 degree of freedoms (dofs) requires less than 2 weeks on a 6-processor system, or less than a day on a 128-processor system [36]. The simulation process can be further accelerated, for example using the adaptive Finite Element method, adaptive time-integration method, etc. These methods produce relatively satisfying results under certain circumstances.

Moreover, many simulations require the solutions to be obtained in a very rapid manner, or the results of simulations and optimisations are dependent on the settings of various parameters, such as Young's modulus, Poisson's ratio, density in structural engineering practices. The particular setting decides that the solution is obtained in a repeated way which could result in thousands of calculations and/or subject to parameter variations. The applications of conventional computational methods are restricted for this class of problems. For example:

- High-dimensional problems: some models are defined in high-dimensional spaces, which suffer the well-known so-called curse of dimensionality. A model defined in dimension N with M nodes in each space results in M^N nodes in total. Applications of standard mesh-based methods are restricted in these cases.
- Time-dependent problems: dynamic problems are time-dependent but not necessarily high-dimensional. Such problems may contain very small time steps in order to satisfy the stability requirements. The system needs to be solved for each time step. If time interval is large, obtaining solutions simply becomes impossible with conventional time-integration methods due to the large number of calculations required.
- Uncertainty problems: Helmholtz-type PDEs govern the propagation model of harmonic waves in unbounded domains, which are solved with a large number of input

data. Uncertainty quantification (UQ) characterizes and predicts certain outcomes in a statistical sense (forward UQ usually requires 1000+ realisations). Small disturbances could have a large impact on the results for these problems. Due to the high computational cost, the number of tests is limited for different parameters, and the accuracy is therefore compromised [56, 82].

- Parametric problems: in problems of electromagnetics, in order to obtain outputs such as the Radar Cross Section (RCS), the electric field integral equation (EFIE) needs to be solved under a variation of a set of parameters (wavenumber, incident plane angle, etc) [35]. Another case is aerodynamic shape optimisation which aims to improve the aerodynamic performance of the design by a set of given parameters. Cost functions need to be minimised to achieve this improvement. Each function evaluation requires a CFD simulation on a different design, therefore it is not practical to re-mesh the model for each individual simulation.

The above problems are required to be solved in a rapid or iterative manner, therefore conventional methods are no longer suitable, and efficient techniques are needed to tackle the complexity of these problems, as well as to seek the balance between numerical accuracy and computational speed. Many methods have been proposed, the most well-known are: Multiscale methods, Modal Truncation methods, Reduced Basis (RB) method [6, 12, 21, 95, 98], Proper Generalised Decomposition (PGD) [18, 88], Proper Orthogonal Decomposition (POD) [8, 58, 67, 107, 112], Data-driven reduced order models [93, 115].

This thesis focuses on obtaining solutions for parametric problems utilising projection-based Reduced Order Modelling (ROM) techniques. The latter are widely applied to reduce the computational cost of predicting the time dependent behaviour of structures subjected to parameter variations, as well as providing efficient and reliable solutions to these PDE outputs. Reduced order model functions are computed in low-dimensional subspaces, therefore the repeated calculations become much more feasible. The reduction

is feasible due to the fact that the parameter-dependent solution can be represented by a linear combination of (i) parameter-independent, pre-computed modes, (ii) parameter-dependent unknowns which are inexpensive to obtain. The key is to construct the modes which is able to well recast the solution. These modes are exact solutions computed from a set of sample points which best represent the parameter domain. The reduction process is divided into an "*Offline/Online*" decomposition: the "*Offline*" stage consists of a sequence of expensive pre-computations during which the solutions are learned and stored; the "*Online*" stage makes use of this accumulated data to predict the solution related to any particular parameter of interest at extremely low costs. In the "*Offline*" stage, a good sampling technique is crucial to the selection of representative 'snapshots'. This thesis focuses on projection based reduction methods (i.e. POD type approaches), coupled with effective sampling techniques.

1.2 Aims and outline

POD-Greedy algorithm further divide the POD "*Offline*" stage into Greedy *basis processing* and *parameter sweep* stages in order to 'greedily' find the most optimised snapshots. The Greedy *basis processing* stage compresses the representative solutions into compact reduced basis; the Greedy *parameter sweep* stage evaluates the error over a carefully chosen training set, once this is finished, the sample snapshot parameter is chosen as the value associated with the maximum error. A main obstacle of applying the standard method is to balance the cost and performance: large training set are more costly but may result in better exploratory of the parameter domain; small training set are less expensive to evaluate but may lead to insufficient search of the information. The aim is to provide new computational strategies based on standard POD-Greedy sampling algorithm to deal with the aforementioned challenges. More specifically, we aim to develop:

- a proposed POD-Greedy sampling algorithm which utilises a new error indicator. This new indicator should enable fast evaluation of large training sets thus achieves an acceleration of the Greedy *parameter sweep* stage. The new POD-Greedy algorithm should achieve accurate prediction of maximum relative error convergence, i.e. approaching the error convergence of standard POD-Greedy algorithm.

- a new error estimate which decides the sequence of local $h-$refinement and Greedy iterations, such that the accuracy is guaranteed.

- a nonintrusive Abaqus/Matlab code coupling technique which allows users to compute solutions of parametric problems using Abaqus as a 'black box'. The nonintrusive technique should be seamlessly integrated into the computation of exact solutions, and works as a bridge between commercial software and ROM source code.

The thesis is structured into the following: in chapter 1, definitions of parametric linear elasto-dynamic problem and the Newmark method are given. Chapter 2 introduces popular reduced order modelling methods, which includes a brief introduction of a priori and a posteriori ROM methods for non-parametric problems, detailed review of a posteriori ROM methods for parametric problems, and other reduced order modelling methods. Various numerical experiments are investigated to show the pros and cons. Chapter 3 introduces the full space-time representation of the Newmark method, which shows that a linear dynamic problem can be solved in a 'static' way. This is the theoretical foundation of the proposed POD-Greedy algorithm. Chapter 4 explains the implementation of a new error indicator developed by applying the full space-time Newmark representation. A new "*error in the error*" *indicator* is presented in chapter 5 to determine the sequence of Greedy iterations and local refinement. A complete computational procedure is proposed in chapter 6, including the pseudo code and the *basis processing* and *parameter sweep* process for the proposed method. Finally the proposed method is validated in chapter 7 with a varieties of numerical examples.

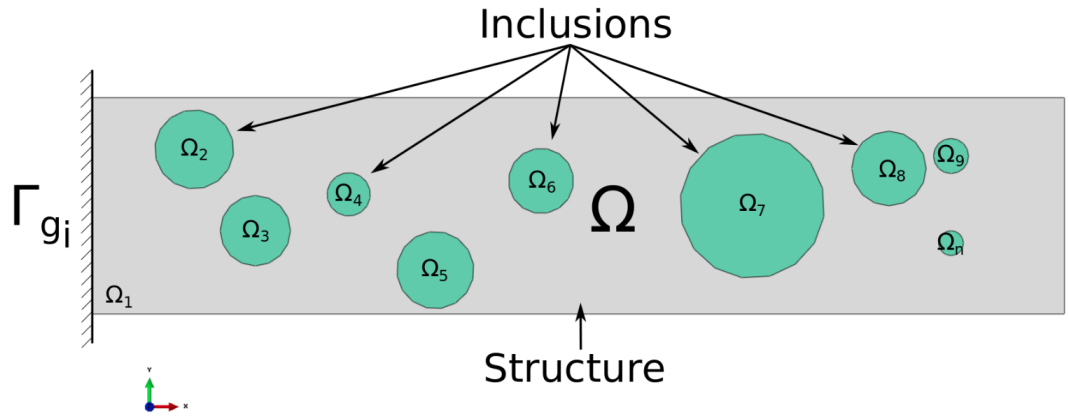


Figure 1.1: Parametric beam model: multiple inclusions.

1.3 Parametric linear elasto-dynamic problem

In this thesis, μ is used to denote the parameter which characterizes the dynamic problem, and $\mathcal{P}$ to denote the parameter domain, such that $\mu \in \mathcal{P}$. μ consists of a finite set of N_μ scalar values $\{\mu_i\}_{i=1}^{N_\mu}$. In structural dynamics, functions of parameter μ consists of density ρ, Young's modulus E, Poisson's ratio ν, etc. Structural responses include acceleration, velocity and displacement, which are also parameter-dependent functions.

The linear elastic structure occupies a bounded domain Ω in the physical space, $\Omega \in \mathbb{R}^d, d = \{1, 2, 3\}$. Domain Ω possesses multiple sub-domains $\{\Omega_1, \ldots, \Omega_n\}$ which denote the inclusions in a structure, for example, a 3D dental implant model is proposed in [51], which consists of 5 physical regions, each is homogeneous and isotropic. Any region which adapts parameter changes can be viewed as an inclusion. Modification of properties in these inclusions affects the behaviour of the structures. Dirichlet boundary conditions are satisfied on boundary Γ_{g_i}. The Hilbert space X is then introduced with inner product $\langle \cdot, \cdot \rangle$ and norm $\|\cdot\|$. The norm is given by $\|x\| = \sqrt{\langle x, x \rangle}$. The solution $\boldsymbol{u}(t; \mu)$ is discretised in space and time, lies in X.

For a general elasto-dynamic problem, the initial conditions specify both displacements and velocity:

$$\begin{aligned} u_{0i} &: \Omega \to \mathbb{R} \\ \dot{u}_{0i} &: \Omega \to \mathbb{R} \end{aligned} \tag{1.1}$$

these initial conditions are given functions for each i, $1 \leq i \leq n_{sd}$, where n_{sd} denotes number of space dimensions, Ω denotes an open set without boundary, $\mathbb{R}$ denotes real numbers. Define remaining prescribed data:

$$\begin{aligned} l_i &: \Omega \times]0,\ T[\to \mathbb{R} \\ g_i &: \Gamma_{g_i} \times]0,\ T[\to \mathbb{R} \\ h_i &: \Gamma_{h_i} \times]0,\ T[\to \mathbb{R} \end{aligned} \tag{1.2}$$

where l_i denotes prescribed body force per unit volume, g_i denotes prescribed boundary displacements, h_i denotes prescribed boundary tractions. For convenience, the parametric setting is omitted in these terms. Density $\rho : \Omega \to \mathbb{R}$ is positive. The input of a parametric problem is a set of N_μ parameter scalars. The strong form of the parametric elasto-dynamic problem is:

(S): Given $l_i, g_i, h_i, u_{0i}, \dot{u}_{0i}$, $\forall \mu \in \mathcal{P}$, find $\boldsymbol{u}_i(t;\mu) : \overline{\Omega} \times [0,\ T] \to \mathbb{R}$, such that

$$\begin{aligned} \rho u_{i,tt}(t;\mu) &= \sigma_{ij,j} + l_i, && \text{on } \Omega \times [0,T] \\ u_i(t;\mu) &= g_i, && \text{on } \Gamma_{g_i} \times [0,T] \\ \sigma_{ij} n_j &= h_i, && \text{on } \Gamma_{h_i} \times [0,T] \\ u_i(x,0;\mu) &= u_{0i}(x;\mu), && x \in \Omega \\ u_{i,t}(x,0;\mu) &= \dot{u}_{0i}(x;\mu), && x \in \Omega \end{aligned}$$

where σ_{ij} denotes Cauchy stress tensor, c_{ijkl} denotes elastic coefficients, u_i denotes the displacement vector. $\sigma_{ij} = c_{ijkl} u_{(k,l)}$ and $c_{ijkl} = c_{ijkl}(x)$. Correspondingly, the weak form reads:

(W): Given $\boldsymbol{l}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{u}_0, \dot{\boldsymbol{u}}_0$, $\forall \mu \in \mathcal{P}$, find $\boldsymbol{u}(t;\mu), t \in [0,T]$, such that $\forall \boldsymbol{w} \in \boldsymbol{V}$,

$$
\begin{aligned}
(\boldsymbol{w}, \rho \ddot{\boldsymbol{u}}(t;\mu);\mu) + a(\boldsymbol{w}, \boldsymbol{u}(t;\mu);\mu) &= (\boldsymbol{w}, \boldsymbol{l};\mu) + (\boldsymbol{w}, \boldsymbol{h};\mu)_\Gamma \\
(\boldsymbol{w}, \rho \boldsymbol{u}(0;\mu);\mu) &= (\boldsymbol{w}, \rho \boldsymbol{u}_0(\mu);\mu) \\
(\boldsymbol{w}, \rho \dot{\boldsymbol{u}}(0;\mu);\mu) &= (\boldsymbol{w}, \rho \dot{\boldsymbol{u}}_0(\mu);\mu)
\end{aligned}
$$

where $(\boldsymbol{w}, \rho \ddot{\boldsymbol{u}}(t;\mu);\mu)$ and $a(\boldsymbol{w}, \boldsymbol{u}(t;\mu);\mu)$ denotes time-dependent parametric bilinear forms. The above equation leads to the following parametric matrix problem in discretised spatial domain:

(M): Given $F : [0,T] \to \mathbb{R}^{n_{eq}}$, $\forall \mu \in \mathcal{P}$, find $\boldsymbol{U}(t;\mu) :]0,T[\to \mathbb{R}^{n_{eq}}$, such that

$$
\begin{aligned}
\boldsymbol{M}(\mu)\ddot{\boldsymbol{U}}(t;\mu) + \boldsymbol{K}(\mu)\boldsymbol{U}(t;\mu) &= \boldsymbol{F}(\mu), \qquad t \in [0,T] \\
\boldsymbol{U}(0;\mu) &= \boldsymbol{u}_0(\mu) \\
\dot{\boldsymbol{U}}(0;\mu) &= \dot{\boldsymbol{u}}_0(\mu)
\end{aligned}
$$

where n_{eq} is the number of global equations. $\boldsymbol{F}$ denotes the force matrix which can be parametric or deterministic depending on the specific problem. See [54] for the detailed derivation from (W) to (M). Introducing the parametric viscous Rayleigh damping matrix $\boldsymbol{C}$, which is mass and stiffness proportional by applying the following form:

$$
\boldsymbol{C}(\mu) = a\boldsymbol{M}(\mu) + b\boldsymbol{K}(\mu) \tag{1.3}
$$

where a and b are damping parameters. Now the parametric system becomes:

$$
\boldsymbol{M}(\mu)\ddot{\boldsymbol{U}}(\mu) + \boldsymbol{C}(\mu)\dot{\boldsymbol{U}}(\mu) + \boldsymbol{K}(\mu)\boldsymbol{U}(\mu) = \boldsymbol{F}(\mu) \tag{1.4}
$$

for $\boldsymbol{M}(\mu), \boldsymbol{C}(\mu), \boldsymbol{K}(\mu) \in \mathbb{R}^{H \times H}$, and $\boldsymbol{F}(\mu) \in \mathbb{R}^{H}$. Physical inclusions in this equation is introduced by the affine parameter expansion, which will be addressed in section 2.2.1.

Consider a parametric quantity of interest (QoI) $\boldsymbol{s}(\mu) = \boldsymbol{s}(\boldsymbol{U}(\mu))$, where $\boldsymbol{s}$ denotes a matrix that consists of 0s and 1s such that certain values in the outputs can be extracted as the interested quantity. The goal is to obtain the quantity of interest that depends on the solution, rather than the finite element solution itself. Engineers might know what part of structure they are interested in, therefore only focus on the output of this part. For instance, vertical displacement of the bridge deck edge [119], average temperature on the boundary of the thermal block [45], average displacement on the head of dental implant screw [52], etc.

1.4 Time discretisation: the Newmark method

In this thesis, Newmark method [20, 39, 40, 87] is applied as the time discretisation scheme. Consider time domain $[0, T]$, it is divided into N subintervals with equal lengths $\Delta t = \frac{T}{N}$, therefore the number of time steps $N_t = N + 1$. The aim is to find a solution $\boldsymbol{U}(t)$ satisfying eq. (1.4). Given initial data:

$$\begin{aligned} \boldsymbol{U}(0) &= \underline{\boldsymbol{u}}_0 \\ \dot{\boldsymbol{U}}(0) &= \underline{\dot{\boldsymbol{u}}}_0 \end{aligned} \tag{1.5}$$

For time step n, the Newton's law of motion reads:

$$\boldsymbol{M}\underline{\ddot{\boldsymbol{u}}}_n + \boldsymbol{C}\underline{\dot{\boldsymbol{u}}}_n + \boldsymbol{K}\underline{\boldsymbol{u}}_n = \underline{\boldsymbol{f}}_n \tag{1.6}$$

Newmark family of method is widely used to solve dynamic problems. The Newmark

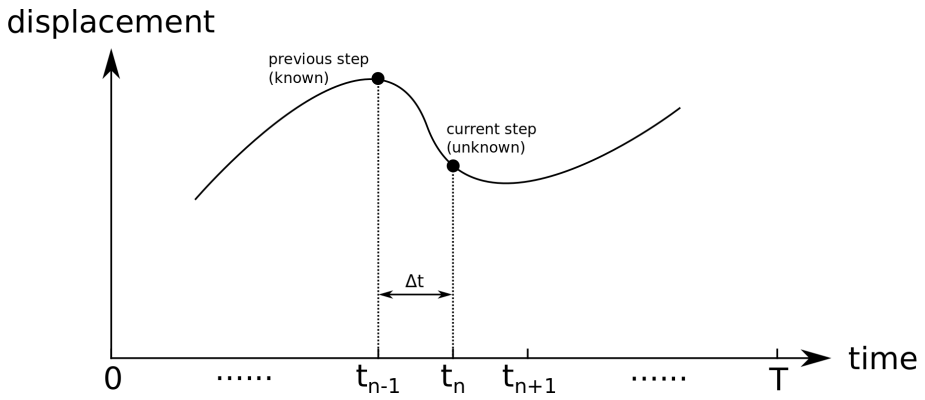


Figure 1.2: Time discretisation

method consists of the following equations:

$$\begin{cases} \underline{\dot{\boldsymbol{u}}}_n = \frac{\gamma}{\beta\Delta t}\underline{\boldsymbol{u}}_n - \frac{\gamma}{\beta\Delta t}\underline{\boldsymbol{u}}_{n-1} - \left(\frac{\gamma}{\beta} - 1\right)\underline{\dot{\boldsymbol{u}}}_{n-1} - \frac{\Delta t}{2}\left(\frac{\gamma}{\beta} - 2\right)\underline{\ddot{\boldsymbol{u}}}_{n-1}, & 0 \le n \le N_t \\ \underline{\ddot{\boldsymbol{u}}}_n = \frac{1}{\beta\Delta t^2}\underline{\boldsymbol{u}}_n - \frac{1}{\beta\Delta t^2}\underline{\boldsymbol{u}}_{n-1} - \frac{1}{\beta\Delta t}\underline{\dot{\boldsymbol{u}}}_{n-1} - \left(\frac{1}{2\beta} - 1\right)\underline{\ddot{\boldsymbol{u}}}_{n-1}, & 0 \le n \le N_t \end{cases} \tag{1.7}$$

if coupling eq. (1.7) with eq. (1.6), and utilize initial condition eq. (1.5), $\underline{\ddot{\boldsymbol{u}}}_n, \underline{\dot{\boldsymbol{u}}}_n, \underline{\boldsymbol{u}}_n$ can be calculated with information from previous time steps. Scalar coefficients γ and β determine the stability and accuracy of the algorithm. In this thesis, coefficients γ and θ are chosen to be $\frac{1}{2}$ and $\frac{1}{4}$, respectively, such that the results are unconditionally stable and 2nd order accuracy is reached, that is to say the numerical error of Newmark outputs is proportional to the time step length to the 2nd power $\left(\boldsymbol{e}(\delta t) = c(\delta t)^2\right)$.

The following clarifications are made for this thesis: the solutions obtained by applying the standard Galerkin finite element method and Newmark's method are defined as the 'truth' or 'exact' solutions. In the reduced order modelling, the RB outputs are approximations of the exact solutions.

Chapter 2

Reduced Order Modelling (ROM) Methods

In this chapter, various techniques applied in reduced order modelling (ROM) community are reviewed. Distinctions are made between 2 main categories: (i) a priori and a posteriori ROM methods for non-parametric problems, (ii) a posteriori ROM methods for parametric problems. The former introduces modal analysis and Proper Orthogonal Decomposition (POD) for time-dependent problems. Then for parametric problems, the introduction starts with snapshot-POD, then extend to global-POD. The Greedy procedure is introduced next. Then these two popular methods are combined to form the POD-greedy algorithm, which is the theoretical foundation of this thesis. At the end of this chapter, other popularly applied methods in ROM community are reviewed. In this chapter, applications of Greedy procedure and POD-Greedy algorithm are demonstrated with numerical examples to assist readers to better understand this thesis.

2.1 A priori and a posteriori ROM methods for non-parametric problems

2.1.1 Modal analysis

Structure under vibrational excitation is studied in the frequency domain using modal analysis [20, 40]. The computational cost can be effectively reduced by using a few dominant modes. The eigenfrequency and eigenvectors are extracted by solving the following eigenvalue problem:

$$\boldsymbol{K}\underline{\boldsymbol{\phi}}_i = \boldsymbol{M}\underline{\boldsymbol{\phi}}_i\lambda_i^2 \tag{2.1}$$

where λ_i denotes the i^{th} eigenvalue, or eigenfrequency or natural frequency in structural dynamics, and $\underline{\boldsymbol{\phi}}_i$ denotes the i^{th} eigenvector, or eigenshapes or mode shape in structural dynamics. The collection of λ_i and $\underline{\boldsymbol{\phi}}_i$ together are called a dynamic mode. The mode shapes are orthogonal to the dynamic system matrices, i.e. $\underline{\boldsymbol{\phi}}_i^T\boldsymbol{M}\underline{\boldsymbol{\phi}}_j = 0$ and $\underline{\boldsymbol{\phi}}_i^T\boldsymbol{K}\underline{\boldsymbol{\phi}}_j = 0$, if $i \neq j$. This property guarantees to recast the vibrations of a multi-dof problem into a sequence of single-dof vibration problems, namely modal superposition. The recast is a linear combination of eigenvectors as follows:

$$\boldsymbol{U}(t) = \sum_{i=1}^{n}\underline{\boldsymbol{\phi}}_i\alpha_i(t) \tag{2.2}$$

where α_i denotes the unknown modal coordinates for mode i. Now the dynamic system can be projected to the subspace spanned by mode i:

$$\underline{\boldsymbol{\phi}}_j^T\left(\boldsymbol{M}\sum_{i=1}^{n}\underline{\boldsymbol{\phi}}_i\ddot{\alpha}_i + \boldsymbol{K}\sum_{i=1}^{n}\underline{\boldsymbol{\phi}}_i\alpha_i\right) = 0 \tag{2.3}$$

Orthogonality of the mode shapes dictates that the above equation is non-zeros only

if $j = i$. Thus a sequence of single-dof system can be obtained:

$$\boldsymbol{m}_i\ddot{\alpha}_i + \boldsymbol{k}_i\alpha_i = 0, \forall\ i \tag{2.4}$$

the space-time responses can then be recast by applying eq. (2.2). Here the natural frequency of mode i is obtained by solving $\lambda_i = \sqrt{\boldsymbol{k}_i/\boldsymbol{m}_i}$. Again the damping matrix is defined by a Rayleigh model: $\boldsymbol{C} = a\boldsymbol{M} + b\boldsymbol{K}$. The reason is Rayleigh damping model facilitates modal analysis and it's computationally convenient. Engineers often want to express $\boldsymbol{C}$ in terms of $\boldsymbol{M}$ and $\boldsymbol{K}$ in order to maintain the classical normal modes. The Rayleigh coefficient is defined as: $\zeta = a\frac{1}{2\lambda_i} + b\frac{\lambda_i}{2}$. [3] is referred as a detailed introduction. With the eigenfrequencies and Rayleigh coefficients in hand, the unknowns can be solved for the dynamic system subject to prescribed forces:

$$\ddot{\alpha}_i + 2\zeta_i\lambda_i\dot{\alpha}_i + \lambda_i^2 = \underline{\phi}_i^T\boldsymbol{F}(t) \tag{2.5}$$

the solution is then recast by applying eq. (2.2). The model is reduced by the fact that not all mode shapes are necessary for recasting the solutions. In fact, due to the large number of degrees of freedom, only the first N dominant eigenfrequencies and associated mode shapes are retained and the rests are discarded. Then the solution is then expanded by the linear combination of the dominant mode shapes and modal coordinates. This is *a priori*, i.e. no solution or *a priori* knowledge is required, the process of obtaining results is independent of experiences.

2.1.2 Proper Orthogonal Decomposition (POD)

For time-dependent problems, construction of a **good and small** basis becomes much harder in this case when time is involved as an additional parameter. The main difficulties are: (i) computation of each solution is more expensive, as the full trajectory of the time-dependent problem needs to be evaluated. (ii) maintaining detailed infor-

mation might result in a very large basis. Hence, discarding unimportant information becomes necessary. However, it might be difficult for the users to decide what is less important and to be discarded. These two obstacles can be tackled by the so-called **Proper Orthogonal Decomposition (POD)** [8, 15, 94, 104]. Different from modal analysis, POD is *a-posteriori* as it generates the basis from the output solution without knowing any knowledge of the system.

Consider the general parametric linear elasto-dynamic problem as described in section 1.3, POD is based on the fact that the time-dependent solution $\boldsymbol{U}(t)$ can be approximated by a finite sum of the linear expansion:

$$\boldsymbol{U}(t) \approx \boldsymbol{U}^{\mathrm{r}}(t) = \sum_{i=1}^{N} \underline{\phi}_i \underline{\alpha}_i(t) = \boldsymbol{\Phi}\boldsymbol{\alpha}(t) \tag{2.6}$$

where $\boldsymbol{\Phi}$ denotes the reduced basis matrix which consists of orthogonal basis vectors:

$$\begin{cases} \boldsymbol{\Phi} = (\underline{\phi}_1, \underline{\phi}_2, \ldots, \underline{\phi}_N) \\ \langle \underline{\phi}_i, \underline{\phi}_j \rangle = \delta_{ij} \end{cases} \tag{2.7}$$

where δ_{ij} is the *Kronecker delta* symbol:

$$\delta_{ij} = \begin{cases} 0, i \neq j \\ 1, i = j \end{cases} \tag{2.8}$$

$\boldsymbol{\alpha}$ denotes the unknown time-dependent reduced variables. The approximation $\boldsymbol{U}^{\mathrm{r}}$ is separated into 2 parts: space-related reduced basis $\boldsymbol{\Phi}$ and time-related reduced variables $\boldsymbol{\alpha}$. POD looks for the reduced basis which gives the best approximation by solving the following minimisation problem:

$$\boldsymbol{\Phi} = \underset{t \in \boldsymbol{T}}{\arg\min} \int_t \left\| \boldsymbol{U}(t) - \boldsymbol{\Phi}\boldsymbol{\Phi}^T \boldsymbol{U}(t) \right\|_2^2 dt \tag{2.9}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product and $\|\cdot\| = \sqrt{\langle \cdot, \cdot \rangle}$ denotes the norm. These definitions depend on the specific settings of the problems. In order to obtain a compact, representative reduced basis, one possible solution is to search for the optimal snapshot location, see [68]. Another family of methods is the compression of the Lagrangian basis. This thesis focuses on the latter approach, which can be achieved by applying the **Singular Value Decomposition (SVD)**.

Singular Value Decomposition

POD is closely related with SVD as the compression in time is achieved by application of SVD. SVD compresses the exact solution into a compact basis without losing the key information. Imagine a solution $\boldsymbol{U}(t)$ is obtained, and let $Z = \boldsymbol{U} = [\underline{z}_1, \ldots, \underline{z}_n] \in \mathbb{R}^{m \times n}$, where $\underline{z}_n$ denotes the column vector for the n^{th} time step, and Z possesses rank c, $c \leq \min\{m, n\}$. Z cannot be directly used as the reduced basis due to the large size. The solution is to apply the Singular Value Decomposition (SVD):

$$Z = U \Sigma V^T \tag{2.10}$$

where $\Sigma \in \mathbb{R}^{m \times n}$ denotes the singular value matrix which contains non-zero singular values σ_i, $(i = 1, \ldots, c)$ on the diagonal entries in a descending order. These singular values are square roots of eigenvalues of both $Z^T Z$ and ZZ^T. $U = [\underline{u}_1, \ldots, \underline{u}_m] \in \mathbb{R}^{m \times m}$ denotes orthonormal left singular vectors, $V = [\underline{v}_1, \ldots, \underline{v}_n] \in \mathbb{R}^{n \times n}$ contains orthonormal right singular vectors. The left singular vector matrix U are a set of eigenvectors of ZZ^T, the right singular vectors V is a set of eigenvectors of $Z^T Z$. More specifically, $\{\underline{u}_i\}_{i=1}^{c}$ is a set of orthonormal vectors which satisfies the eigenvalue problem: $ZZ^T U = \lambda U$, and $\lambda_i = (\sigma_i)^2$. The remaining singular vectors $\{\underline{u}_i\}_{i=c+1}^{m}$ are eigenvectors of ZZ^T with zero eigenvalues.

The following clarifications are made for dimensions of non-zero portion of Σ: if $m < n$, diagonal entries of the $m \times m$ dimensional left block of Σ contains non-zero singular values

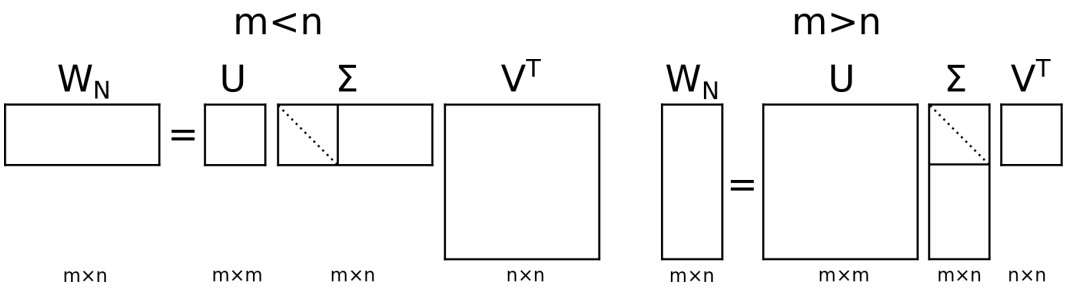


Figure 2.1: Dimensional differences between SVD cases.

and the $m \times (n-m)$ dimensional right block contains zeros; in the contrary, if $m > n$, diagonal entries of the $n \times n$ dimensional upper block of Σ contains non-zero singular values and the $(m-n) \times n$ dimensional lower block contains zeros. See fig. 2.1 for a schematic illustration.

The goal of POD is finding k vectors which best represent solution Z. For time-dependent problems, dimension m and n might be very large, therefore it is not economic to directly utilize all c ($c \leq \min\{m,n\}$) vectors of U as the reduced basis. One can prove that using the first k ($k \leq c$) left singular vectors $\tilde{U} = \{\underline{u}_i\}_{i=1}^{k}$ as an approximation of Z is optimal on the mean along all rank k approximations to the columns of Z [107]. Hence $\tilde{U}$ is used as the **POD basis**. Denote the POD progress as: $\mathbf{\Phi}_k := POD_k(Z) := \{\underline{\phi}_i\}_{i=1}^{k}$, which means POD is applied to dataset Z and k basis vectors are chosen as the reduced basis. The best approximation property of the left singular vectors with respect to the mean square projection error is: chosen $\mathbf{\Phi} = \tilde{U}$ results in the following:

$$\left\| Z - \tilde{U}\tilde{\Sigma}\tilde{V}^T \right\|_{\mathrm{F}} = \left\| Z - \mathbf{\Phi}\mathbf{\Phi}^T Z \right\|_{\mathrm{F}} = \sqrt{\sum_{i=k+1}^{c} (\sigma_i)^2} \tag{2.11}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. The POD basis is hierarchical, i.e. $\mathbf{\Phi}_j \subset \mathbf{\Phi}_k$ for $j \leq k$.

Truncated-SVD

Using the first k ingredients of SVD vectors leads to the concept of truncated-SVD (t-SVD):

$$Z \approx \tilde{Z} = \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{2.12}$$

truncated-SVD selects the first k ($k \leq c$) column vectors of U and V corresponding to the first k largest singular values $(\sigma_i)_{i=1}^k$, such that $\tilde{Z}$ is the best approximation of Z in terms of order k. T-SVD results in the following truncation error which can be used to determine the dimension of the reduced basis:

$$\boldsymbol{e}(\boldsymbol{\Phi}) = \sqrt{\frac{tr(\tilde{\Sigma}^T\tilde{\Sigma})}{tr(\Sigma^T\Sigma)}} = \sqrt{\frac{\sum_{i=k+1}^{c}(\sigma_i)^2}{\sum_{i=1}^{c}(\sigma_i)^2}} \tag{2.13}$$

the numerical distance between Z and $\tilde{Z}$ in the l^2-norm and Frobenius norm are:

$$\begin{aligned} \left\|Z - \tilde{Z}\right\|_2 &= \sigma_{k+1} \\ \left\|Z - \tilde{Z}\right\|_F &= \sqrt{\sum_{i=k+1}^{c}(\sigma_i)^2} \end{aligned} \tag{2.14}$$

Truncation error may be used as a choice of determining the number of basis vectors to be used in an RB-model. However the truncation error is different from the RB-error (eq. (2.21)) or projection error (eq. (2.27)), thus needs to be treated carefully in applications.

2.2 A posteriori ROM methods for parametric problems

From now on the thesis focuses on parametric problems. The definition of parametric linear elasto-dynamic problem refers to section 1.3. Solutions of parametric PDEs

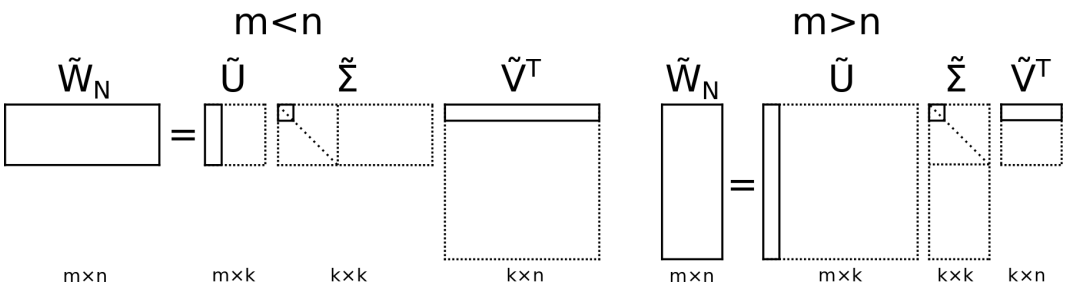


Figure 2.2: Dimensions of truncated SVD.

might be obtained in high-dimensional space, result in large number of simulations and expensive computational cost. Numerical algorithms are in high demand to improve the efficiency and reduce the cost. $\boldsymbol{u}(\mu)$ is used to denote the solution of parametric PDE, a quantity of interest $s(\mu)$ in both space and time is desired as an output. High dimensional parametric space remains as a challenge, as parametric problems are often required to be solved in a iterative manner. In other applications, the simulations sometimes are required to be real-time or at least in a fast manner. However high-dimensional solutions are expensive to be obtained, thus limits the applications of conventional methods.

The reduced Basis (RB) method is a family of methods which aims at tackling the above limitations. The goal of the reduced basis method is to compute low-dimensional approximations that allow rapid computation of the outputs. For the family of parametric PDEs, the crucial insight which allows the use of reduced basis method is that the solution manifold can often be well-approximated by a low-dimensional subspace $X_N \in X$. The so-called '*snapshots*' $(\boldsymbol{u}(\mu_i) \in X, i = 1, \ldots, N)$ are sought to construct the subspace. The snapshots are solutions of the full problem at parameter samples μ_i, originally defined in [103]. Correspondingly, the chosen parametric sample point μ_i are known as magic points [38, 76]. Construction of the reduced basis is a progress of seeking the magic point set $\mathcal{P}^M$, as well as the associated representative snapshots. It will be shown that the selection of snapshot parameter samples has direct impact on the performance of the RB

model.

Reduced basis methods usually works with the assistance of *Galerkin Projection*. Once the reduced model is established, one can quickly compute solution approximation $u_N(\mu) \subset X_N$ and output approximation $\boldsymbol{s}_N(\mu)$. *A-posteriori* error estimation also plays an important role to evaluate the accuracy of the approximations. When applying the reduced basis method, the following questions arise and need to be answered:

- How to construct the reduced space X_N?
- How to compute an accurate approximation $\boldsymbol{u}_N$?
- How to construct a good reduced basis?
- How to evaluate the performance of the reduced basis?

Definitions in terms of static problems are given first, then this is extended to dynamic problems.

2.2.1 Problem definitions

Full problem (P)

The full problem (P) is defined as follows:

(P): For $\mu \in \mathcal{P}$, find solution $\boldsymbol{u}(\mu) \in X$, and output of interest $\boldsymbol{s}(\mu) \in \mathbb{R}$, $t \in [0, T]$, such that $\forall \boldsymbol{v} \in X$,

$$\begin{aligned} a(\boldsymbol{u}(\mu), \boldsymbol{v}; \mu) &= f(\boldsymbol{v}; \mu), \\ \boldsymbol{s}(\mu) &= l(\boldsymbol{u}(\mu)) \end{aligned} \tag{2.15}$$

Lagrangian reduced basis

Given a set of linearly independent solutions, Lagrangian Reduced basis is a simple type of reduced basis defined as:

$$\begin{aligned} \boldsymbol{\Phi} &:= \{\boldsymbol{u}(\mu_1), \ldots, \boldsymbol{u}(\mu_N)\} \\ \text{where } \mathcal{P}_N &:= \{\mu_1, \ldots, \mu_N\} \subset \mathcal{P} \end{aligned} \tag{2.16}$$

RB-problem (P_N)

A low-dimensional space X_N can be constructed as follows:

$$X_N := span(\boldsymbol{\Phi}) = span\{\boldsymbol{u}(\mu_1), \ldots, \boldsymbol{u}(\mu_N)\} \subset X \tag{2.17}$$

with a basis $\boldsymbol{\Phi} = \{\underline{\phi}_1, \ldots, \underline{\phi}_N\}$, X_N denotes the reduced basis space. Here $\boldsymbol{u}(\mu_i)$ denotes the exact solutions being computed at parameter sample value μ_i. With a careful selection of $\mathcal{P}_N$, a Lagrangian reduced basis is able to provide a good approximation globally. Defining the reduced basis space allows us to define the RB-problem (P_N):

(P_N): For $\mu \in \mathcal{P}$, find solution $\boldsymbol{u}_N(\mu) \in X_N$, and output of interest $\boldsymbol{s}_N(\mu) \in \mathbb{R}$, such that $\forall \boldsymbol{v} \in X_N$,

$$\begin{aligned} a(\boldsymbol{u}_N(\mu), \boldsymbol{v}; \mu) &= f(\boldsymbol{v}; \mu), \\ \boldsymbol{s}_N(\mu) &= l(\boldsymbol{u}_N(\mu)) \end{aligned} \tag{2.18}$$

Affine parameter expansion An important assumption in this thesis is that the bilinear and linear forms naturally admit an affine parameter expansion:

$$a(\boldsymbol{u}, \boldsymbol{v}; \mu) = \sum_{c=1}^{C_a} a_c(\boldsymbol{u}, \boldsymbol{v}) \gamma_c^a(\mu) \tag{2.19}$$

for some small integer C_a (the complexity of the RB problem depends explicitly on the quantity of C_a). Here $\gamma_c^a(\mu) \in \mathbb{R}$ are parameter-dependent coefficient functions, and the bilinear forms $a_c(\cdot, \cdot)$ are parameter-independent. Affine parameter dependence is beneficial in reduced order modelling techniques as the parameter-independent terms can be separated and pre-computed in the "*offline*" stage once and for all. Whenever is needed, the system matrices may be obtained by a simple linear combination of the pre-computed operators and coefficients, such that the complex assembly is avoided. The "*offline/online*" computational strategy will be introduced in detail later. If this property does not hold, i.e. the system matrices are non-affine, then some types of interpolation methods are required to construct the affine approximations of the non-affine terms, such as Empirical Interpolation Method (EIM) [6, 32], Discrete Empirical Interpolation Method (DEIM), [16, 92], etc.

Discrete RB-problem (P_D)

Defining the reduced space allows us to write the discrete RB-problem (P_D):

(P_D): Given reduced basis $\boldsymbol{\Phi} = \{\underline{\phi}_1, \ldots, \underline{\phi}_N\}$, for $\mu \in \mathcal{P}$, find solution $\mathbf{u}_N(\mu) = (u_N, i)_{i=1}^N \in \mathbb{R}^N$ by solving the following linear system:

$$\mathbf{K}_N(\mu)\mathbf{u}_N(\mu) = \mathbf{f}_N(\mu), \text{ and}$$

$$\mathbf{K}_N(\mu) := (a(\underline{\phi}_j, \underline{\phi}_i; \mu))_{i,j=1}^N = \boldsymbol{\Phi}^T \mathbf{K} \boldsymbol{\Phi} \in \mathbb{R}^{N \times N},$$

$$\mathbf{f}_N(\mu) := (f(\underline{\phi}_i; t; \mu))_{i=1}^N = \boldsymbol{\Phi}^T \mathbf{f} \in \mathbb{R}^N,$$

$$\mathbf{l}_N(\mu) := (l(\underline{\phi}_i; \mu))_{i=1}^N = \boldsymbol{\Phi}^T \mathbf{l} \in \mathbb{R}^N$$

after solving P_D and obtaining solution $\mathbf{u}_N$, the solution of P_N can be written as follows:

$$\boldsymbol{u}_N(\mu) = \boldsymbol{\Phi}\mathbf{u}_N = \sum_{j=1}^{N} u_{N,j}\underline{\phi}_j \in \mathbb{R}^H, \ \ \boldsymbol{s}_N(\mu) = \mathbf{l}_N^T(\mu)\mathbf{u}_N(\mu) \tag{2.20}$$

the solution $\mathbf{u}_N$ may be compared with the reduced variables $\boldsymbol{\alpha}$ as aforementioned in POD, see section 2.1.2. Similarities: (i) they are both μ-dependent (for parametric problems); (ii) they are the solutions to the unknown RB-problems, which lies in a low-dimensional space, thus easy to be solved in a parameter sweep stage. Their difference lies in dimensions as $\mathbf{u}_N \in \mathbb{R}^N$ and $\boldsymbol{\alpha} \in \mathbb{R}^{N\times T}$. In other words, for time-dependent problems, RB-model only reduces the spatial complexity.

The solution of the RB-problem $\boldsymbol{u}_N(\mu)$ is high-fidelity, in contrast to some other numerical methods which also provide fast solutions but with low-fidelity. The reduced basis is usually orthogonalised by post-processing: for example, once the snapshots $\boldsymbol{u}(\mu_i)$ are obtained, a standard *Gram-Schmidt* process can be applied to the snapshots such that the orthogonality of basis $\boldsymbol{\Phi}_N$ is ensured. One may notice the differences between Reduced Basis method and classical Finite Element method:

- In the setting of Finite Element method, $\mathbf{K} \in \mathbb{R}^{H\times H}$, where H is a large number; in Reduced Basis method, $\mathbf{K}_N \in \mathbb{R}^{N\times N}$, where N is small. However, $\mathbf{K}_N$ is typically dense while $\mathbf{K}$ is sparse.
- The solution of P_N is produced by a linear combination of $u_{N,j}$ and $\underline{\phi}_j$ (eq. (2.20)). N is typically small, hence real-time computations are enabled and this process is much faster than solving the classical Finite Element problem.
- Orthogonality of the reduced basis guarantees stability, i.e. the condition number of $\mathbf{K}_N$ remains stable if N grows. In Finite Element method, the condition number of $\mathbf{K}$ grows polynomially in H.

The exact error or true RB-error for elliptic problems is:

$$\boldsymbol{e}(\mu) := \boldsymbol{u}(\mu) - \boldsymbol{u}_N(\mu) \in X \tag{2.21}$$

the error analysis is residual-based, therefore the residual is defined as:

$$r(\boldsymbol{v};\mu) := f(\boldsymbol{v};\mu) - a(\boldsymbol{u}_N(\mu), \boldsymbol{v};\mu), \boldsymbol{v} \in X \tag{2.22}$$

hence the error-residual relation is:

$$a(\boldsymbol{e}, \boldsymbol{v};\mu) = r(\boldsymbol{v};\mu) \tag{2.23}$$

the residual in static problems can be expressed in the matrix form as follows:

$$\boldsymbol{R}(\mu) = \mathbf{f} - \mathbf{K}(\mu)\boldsymbol{\Phi}\mathbf{u}_N(\mu) \tag{2.24}$$

similarly, the residual can be extended to dynamic problems as follows:

$$\boldsymbol{R}(t;\mu) = \mathbf{f} - \mathbf{M}(\mu)\boldsymbol{\Phi}\ddot{\mathbf{u}}_N(t;\mu) - \mathbf{C}(\mu)\boldsymbol{\Phi}\dot{\mathbf{u}}_N(t;\mu) - \mathbf{K}(\mu)\boldsymbol{\Phi}\mathbf{u}_N(t;\mu) \tag{2.25}$$

the residual satisfies a Petrov-Galerkin condition:

$$\boldsymbol{\Phi}^T\boldsymbol{R}(t;\mu) = 0 \tag{2.26}$$

"*Offline/Online*" decomposition

Standard ROM process typically includes the following ingredients: (i) computation of N snapshots, i.e. N exact solutions, (ii) projection of system matrices on a low-dimensional subspace, (iii) computation of the approximated solutions $\mathbf{u}_N$. Compared

with Finite Element method, RB-approach would only be worthy under a multi-query scenario, which particularly suits the need of parametric problems. An "*Offline/Online*" decomposition is thus developed to satisfy the above requirement: during the "*offline*" stage, a series of expensive solutions to the parametric problems are precomputed and stored, which is done once and for all; in the "*online*" stage, the accumulated data obtained from the *offline* stage are utilised to solve the low-dimensional system and predict the μ-dependent solution $\mathbf{u}_N(\mu) \in \mathbb{R}^N$ and output of interest $s_N(\mu)$ rapidly. The reduced basis method is efficient due to the fact that the numerical complexity in the online stage only depends on N, which is typically a small quantity. The offline stage can be a relatively expensive process, usually costs more than computing a few high-dimensional solutions. The detailed procedures are given below:

"***Offline***" **stage:**

- Select the training samples $\mu_i \in \mathcal{P}^{\text{train}}$ to be the snapshot parameters, solve the set of full problems at μ_i, obtain the exact solutions $\boldsymbol{u}(\mu_i)$, then compute the reduced basis $\boldsymbol{\Phi} = \{\underline{\phi}_1, \ldots, \underline{\phi}_N\}$, obtain the reduced affine system matrices and vectors by Galerkin projection:

$$\begin{aligned} \mathbf{K}_{N,c} &:= \boldsymbol{\Phi}^T \mathbf{K}_c \boldsymbol{\Phi} \in \mathbb{R}^{N\times N}, && c = 1, \ldots, C_k, \\ \mathbf{f}_{N,c} &:= \boldsymbol{\Phi}^T \mathbf{f}_c \in \mathbb{R}^N, && c = 1, \ldots, C_f, \\ \mathbf{l}_{N,c} &:= \boldsymbol{\Phi}^T \mathbf{l}_c \in \mathbb{R}^N, && c = 1, \ldots, C_l. \end{aligned}$$

"***Online***" **stage:**

- Apply affine expansion, multiply the affined system matrices and vectors with co-

efficient functions for all $\mu \in \mathcal{P}$:

$$\mathbf{K}_N(\mu) = \sum_{c=1}^{C_k} \gamma_c^k \mathbf{K}_{N,c}$$
$$\mathbf{f}_N(\mu) = \sum_{c=1}^{C_f} \gamma_c^f \mathbf{f}_{N,f}$$
$$\mathbf{l}_N(\mu) = \sum_{c=1}^{C_l} \gamma_c^l \mathbf{l}_{N,c}.$$

which results in the discrete system as aforementioned in section 2.2.1. Then for any desired parameter values μ, the approximation and output quantity of interest is obtained by applying eq. (2.20).

2.2.2 Snapshot-POD

For parametric problems, ROM seeks the many 'snapshots' to construct the low-dimensional subspace X_N. The snapshots are a dataset of N exact solutions $\boldsymbol{u}(\mu_i)$, being computed at parameter sample points $\mu_i, i = 1, \ldots, N$, known as the magic points. Computation of the reduced basis depends on the size of snapshot matrix $Z = [\boldsymbol{u}(\mu_1), \ldots, \boldsymbol{u}(\mu_N)]$: if the dataset is small, then dataset Z can be used as POD basis without compression, orthonormalization is still required in order to guarantee stability; if the dataset is large, the dataset can be compressed by SVD and the orthonormalized left singular vectors are used as POD basis. However for dynamic problems, the snapshot dataset is usually large, therefore SVD is essential to be applied to the dataset such that a compact basis can be built. Using snapshots to generate POD basis is named as snapshot-POD method [14, 58, 59, 60, 80, 99].

A crucial point to the application of snapshot-POD is the choices of magic points. In some applications they are uniformly distributed in the parameter domain [4, 13,

49], otherwise they might be determined by sampling methods such as pseudorandom sampling, quasi-random sampling [60, 102], Latin Hypercube sampling [9, 17, 78], etc. First briefly the above sampling methods are briefly introduced in the subsequent section.

Statistically based sampling approaches (choice of magic points)

Pseudorandom sampling The magic points might be a subset of $\mathcal{P}$ in which each point of the subset has equal probability of being selected. Matlab `Randi` function is used in this thesis to randomly select magic points in the logarithmically distributed parameter domain to compute snapshots. Due to the random property, each time the set of chosen magic points might be different, thus N experiments are conducted to ensure generality. Notice that the random numbers generated in Matlab are pseudorandom, i.e. they are not random in a strict, mathematical sense.

Quasi-random sampling Quasi-random sampling uses low-discrepancy sequences, in contrast to pseudorandom sequences which have high-discrepancy. Low-discrepancy property indicates that quasi-random sequences are more uniformly distributed. Quasi-random sampling is more advanced than pseudorandom sampling in Monte-Carlo simulations as it converges faster. Quasi-random sequences are not real-random sequences either, instead they are deterministic. In this thesis Matlab `haltonset` and `sobolset` functions are used to generate Halton and Sobol sequences, respectively.

Latin Hypercube sampling Latin Hypercube sampling (LHS) can also be used to construct sets of near random values. It divides the input samples into N Latin squares, each possesses equal probability. Only 1 sample is drawn from each row and column of the Latin square, such that uniform distribution is guaranteed. Matlab `lhsdesign` function is used in this thesis to generate Latin Hypercube samples.

Demonstration A 2-dimensional domain $\mathcal{P} = [0,1]\times[0,1]$ with sample number $N = 100$ are used to demonstrate the points generated by different sampling methods. Visually Sobol sequence has the lowest discrepancy.

2.2.3 Global-POD

In section 2.1.2, SVD is applied on the individual snapshot $\boldsymbol{U}(\mu_i)$, which results in a reduction in time. However, the application of POD is not limited in time, the reduction in parameter might also be achieved. The POD in both parameter and time is named as the Global-POD. Imagine N snapshots have been successfully collected, and Z is used to denote this collection, such that $Z = \{\boldsymbol{U}(\mu_i)\}_{i=1}^{N}$. The POD basis of rank k consists of the first k left singular vectors obtained by applying SVD on Z, i.e. $\boldsymbol{\Phi}_k := POD_k(Z) = \{u_i\}_{i=1}^{k} = \{\underline{\phi}_i\}_{i=1}^{k}$. Again, this POD basis is optimal in the mean along all rank k approximations to Z. Global-POD requires all snapshots available and compresses them all at once, thus an effective sampling method is needed for selecting the snapshots. The output global-POD basis is not a simple combination of POD basis.

2.2.4 The Greedy procedure for linear static problems (choice of magic points)

Driven by the goal of minimising the error indicator, the Greedy procedure [44, 45, 91, 106] plays a powerful role in reduced basis construction, i.e. the ROM "*offline*" stage. Greedy algorithm is widely applied in optimisation type of problems, which require the optimal solution iteratively such that the overall result is optimal. This iterative in the algorithm gives the name 'Greedy'. The key idea is to minimize the output error by iteratively enriching the snapshot space with solutions computed from a discrete path which trained in a Greedy manner. A training set of parameter values $\mathcal{P}^{\text{train}} \subset \mathcal{P}$ are selected to represent the parametric domain. Correspondingly, the number of the training samples in $\mathcal{P}^{\text{train}}$ is denoted by N_{train}. N_{train} is crucial to the performance of the Greedy

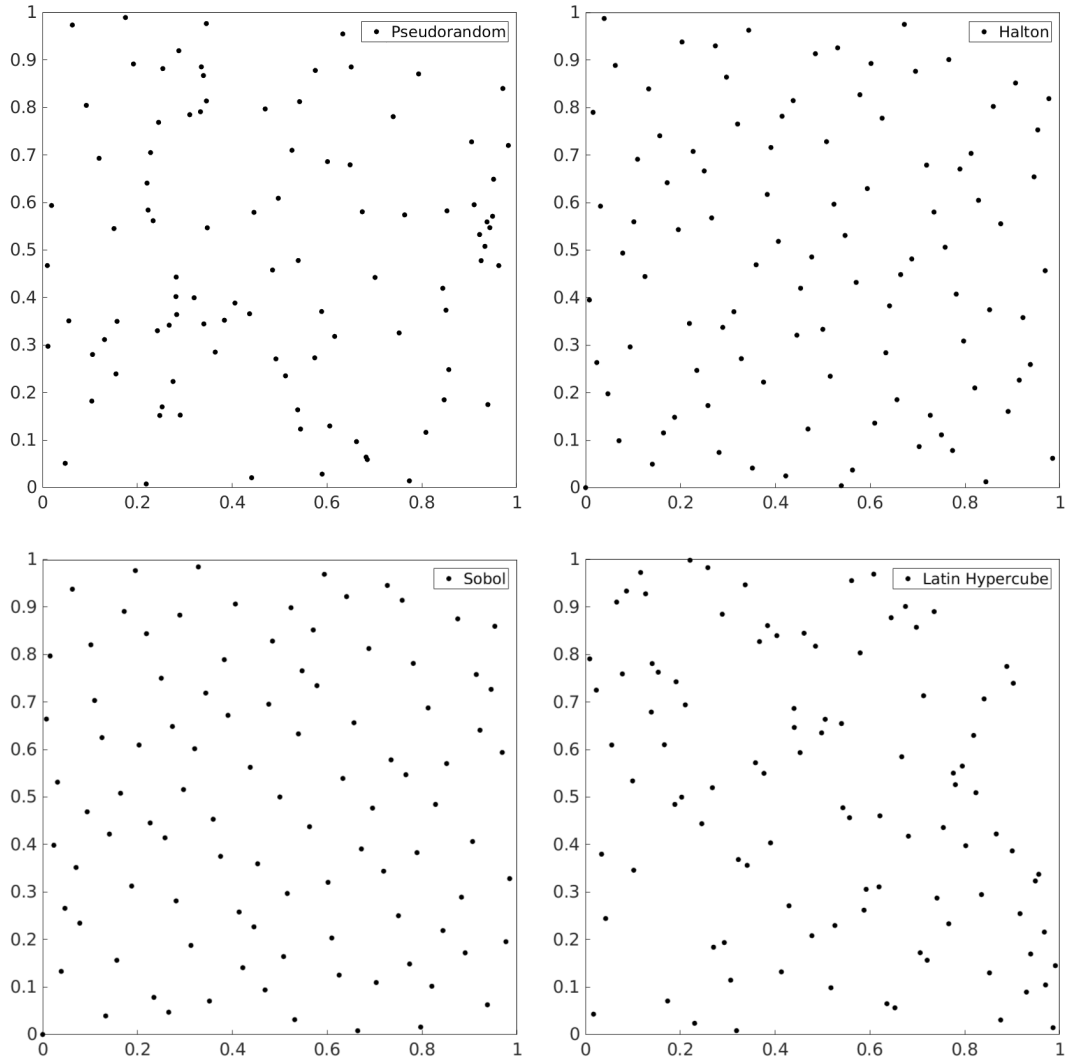


Figure 2.3: An example demonstration of different sampling methods, 100 sample points are drawn from each method. Discrepancy $D_{\text{Sobol}} < D_{\text{Halton}} < D_{\text{LHS}} < D_{\text{Pseudorandom}}$.

algorithm: if too small, $\mathcal{P}^{\text{train}}$ is not able to represent $\mathcal{P}$, and key information might be ignored; if too large, cost of the Greedy algorithm would be too expensive. The reduced basis is enriched in an accumulative manner: each Greedy iteration finds 1 basis vector and adds it to the previous basis to form a new basis (orthonormalization might needed). The Greedy procedure requires the users to initiate the following ingredients: an error indicator $\boldsymbol{e}_{\text{ind}}(\mu)$ and a tolerance tol^{err}.

Error indicator

An error indicator $\boldsymbol{e}_{\text{ind}}(\mu)$ is set to denote the approximation error. This indicator is essential as it directs the linear search of the reduced basis. There are different choices of $\boldsymbol{e}_{\text{ind}}(\mu)$, for example

- True RB-error or the exact error $\boldsymbol{e}(\mu)$ (eq. (2.21)): this error indicator requires the RB-model and all exact solutions for the discretised parameter values, hence it is expensive to evaluate. However, $\boldsymbol{e}(\mu)$ fits the goal of Greedy procedure and is accurate as it directly evaluates the numerical distance between exact solution and approximation, thus is advantageous.

- Projection error $\boldsymbol{e}_{\text{proj}}$, which is defined as:

$$\boldsymbol{e}_{\text{proj}}(\mu_n) := \left\| \boldsymbol{u}(\mu_n) - \text{proj}_{\boldsymbol{\Phi}} \boldsymbol{u}(\mu_n) \right\| = \left\| \boldsymbol{u}(\mu_n) - \boldsymbol{\Phi}\boldsymbol{\Phi}^T \boldsymbol{u}(\mu_n) \right\| \tag{2.27}$$

  where $\text{proj}_{\boldsymbol{\Phi}} \boldsymbol{u}(\mu_n)$ denotes the orthogonal projection of $\boldsymbol{u}$ at magic point μ_n onto the current reduced space spanned by $\boldsymbol{\Phi}$. $\boldsymbol{e}_{\text{proj}}$ has several advantages: (i) the information which cannot be approximated by $\boldsymbol{\Phi}$ is contained in $\boldsymbol{e}_{\text{proj}}$; (ii) once a reduced basis is obtained, $\boldsymbol{e}_{\text{proj}}$ can be utilised without solving the RB problem; (iii) no a-posteriori error estimators are required. The projection error is also expensive to use as exact solutions $\boldsymbol{u}(\mu_n)$ need to be computed.

- A-posteriori error estimator, which requires an RB-model and a-posteriori error estimator. If these are available, this approach is recommended as it does not

require computation of all snapshots. The training set can be set to be fairly large and more representative, results in a much better reduced basis.

- Goal-oriented error indicator. The basis can be quite small but $\boldsymbol{u}(\mu)$ is not guaranteed to be well-approximated.

This thesis focuses on elasto-dynamic problems, which lack of a verified, inexpensive a-posteriori error estimator (to the best of author's knowledge). Therefore the true RB-error is utilised when an error indicator is needed to evaluate the numerical distance between exact solution and approximation. The following clarification is made such that readers comprehend the associated terms when reading subsequent chapters: a standard Greedy procedure refers to the Greedy algorithm with any available error indicators; a reference Greedy procedure refers to Greedy algorithm with true RB-error as an error indicator.

The algorithm

The Greedy algorithm is demonstrated in algorithm 1. The Greedy procedure further divides the RB "*Offline*" stage into a Greedy *basis processing* stage and a *parameter sweep* stage (the terminology *parameter sweep* is adapted from [45]), where the training set $\mathcal{P}^{\text{train}}$ is exhaustively swept during the *parameter sweep* such that the sample which maximize the error can be drawn. Once the samples are chosen, the associated exact solutions are computed. With current reduced basis and the exact solution in hand, the projection error can be computed to be used as the newly added information. Size of $\mathcal{P}^{\text{train}}$ has a direct impact on the number of operations in the Greedy procedure. The numerical complexity is introduced in [45]: the computation of the snapshots consists of $\mathcal{O}(NH^2)$ operations which dominants the cost of the entire Greedy procedure. Hence the size of $\mathcal{P}^{\text{train}}$ needs to be carefully set to guarantee the efficiency. In order not to miss any important information, the size should be chosen as large as possible.

Algorithm 1 The Greedy algorithm

Input: tol^{err} (User defined error tolerance)

Output: N, X_N, $\mathcal{P}_N$

Choose $\mu_1 \in \mathcal{P}^{\text{train}}$, initialize $N = 1$, set $\mathcal{P}_1 = \{\mu_1\}$, $X_1 = \text{span}\{\boldsymbol{u}(\mu_1)\}$ ▷ Initialisation
Define $\boldsymbol{e}_{\text{gr}} := \max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$
while $\boldsymbol{e}_{\text{gr}} \geq \text{tol}^{\text{err}}$ **do**
    for all $\mu \in \mathcal{P}^{\text{train}}$ **do**
        compute $\boldsymbol{e}_{\text{ind}}(\mu)$ ▷ *parameter sweep*
    end for
    $\mu_{n+1} := \arg\max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$ ▷ New samples (magic point)
    $\phi_{n+1} := \boldsymbol{u}(\mu_{n+1})$
    $\boldsymbol{\Phi}_{n+1} := \boldsymbol{\Phi}_n \bigcup \phi_{n+1}$ ▷ Basis enrichment
    $\mathcal{P}_{N+1} := \mathcal{P}_N \bigcup \{\mu_{n+1}\}$
    $X_{N+1} := X_N \bigoplus \phi_{n+1}$
    Set $N := N + 1$
end while

Here a detailed definition of different stages of a ROM approximation with the Greedy algorithm is given as follows:

- ROM "*offline*" stage: basis construction
  - POD-Greedy *parameter sweep*: evaluate error for every single parameter value of $\mathcal{P}^{\text{train}}$, step 4-5.
  - POD-Greedy *basis processing*: find and compress snapshot to obtain $\boldsymbol{\Phi}$, step 6-11.
- ROM "*online*" stage: calculations of $\boldsymbol{U}^{\text{r}}(\mu)$ for single or multiple parameter values of interest μ.

Error response surface

An important component of the Greedy *parameter sweep* is construction of the error response surfaces. Response surface methodology [27, 61, 83, 85] is a family of mathematical and statistical methods which establish a functional relationship between the input control variables $x_1, \ldots, x_n$ and output of interest y. The *parameter sweep* process fits this definition as it establishes the relationship between the input of parametric values and the output error indicator values. For n control variables $\{x_i\}_{i=1}^{n}$, n experiments are carried out, and for each experiment the error $\boldsymbol{e}_{\text{ind}}$ is measured. Once this evaluation stage is finished, an error response surface can be constructed and the control variable which gives the maximum error is located.

Training set treatment

A training set $\mathcal{P}^{\text{train}}$ is generated and swept in order to evaluate the error in the parameter domain. This training set needs to be carefully selected: if it's too large, the computation would be unnecessarily cumbersome; if it's too small, the domain $\mathcal{P}$ would

not be represented well and key information might be dropped out. This training set problem has been treated in many researches, such as:

- Multistage Greedy: a subset of the training set is generated, and start the Greedy algorithm on a subset, generate a basis, repeat the procedure until the Greedy algorithm is running on the entire $\mathcal{P}^{\text{train}}$. The method generates a basis with degenerated quality and improved efficiency, see [101].

- Full scale realisation: the entire discretised parameter domain can sometimes be treated as $\mathcal{P}^{\text{train}}$ under certain circumstances, see [105].

- Randomisation: in each Greedy iteration, a random training set is drawn such that effectively the training set is enlarged, more samples are evaluated than using a fixed training set. See [50].

- Adaption: the Greedy procedure is started with a coarse grid, then the error estimator is evaluated for parameter subdomains, the subdomain corresponding to the largest error is refined such that $\mathcal{P}^{\text{train}}$ is enriched. See [46].

Parameter nondimensionalisation

The parameters in this thesis are set to be nondimensional, the nondimensionalisation is done by the follows: take aluminium as an example material, knowing Young's modulus of aluminium is 69 GPa, then the nondimensional material parameter μ is set to be a relative value which ranges in $\frac{69\text{GPa}}{[690\text{GPa},6.9\text{GPa}]} = [10^{-1}, 10^{1}]$, such that discretisation of the parameter domain can be log-uniform. In real applications, the parameter set might be stochastic, which may involve uncertainty quantification. However in this thesis a deterministic sample set is used, as parameter uncertainty is not the focus of this study.

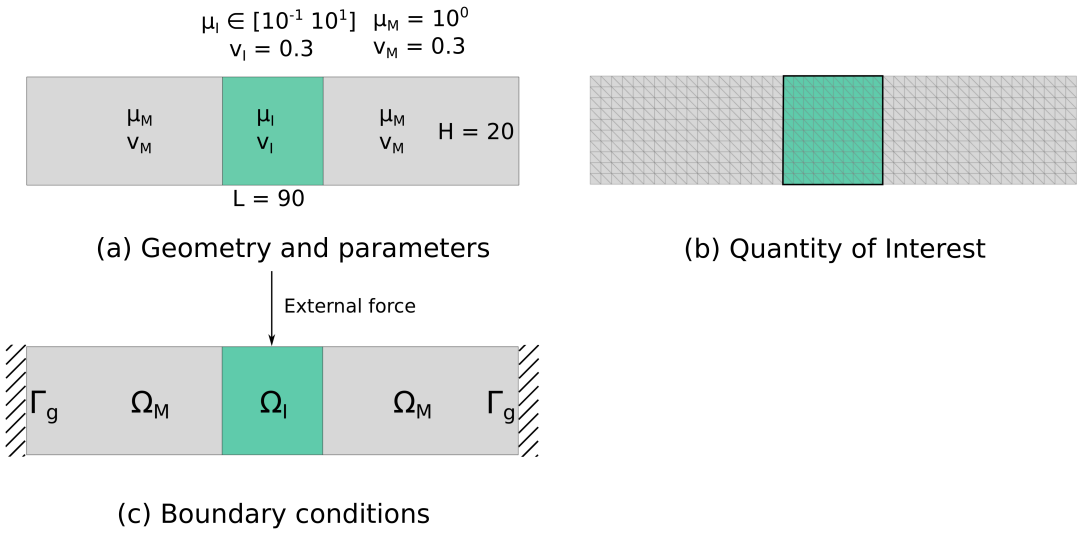


Figure 2.4: The 2D beam model and its FE reference mesh.

Greedy procedure: numerical example and results

The simple parametric beam model as shown in fig. 2.4 is considered to demonstrate the Greedy procedure with true RB-error as the error indicator. For simplicity, the parametric model is set to possess only 1 inclusion. Uniform Dirichlet boundary conditions is applied at both end.

Announcement These numerical examples in this section are based on standard Greedy procedure [45] thus the author does not claim any novelty (so does the numerical examples of POD-Greedy algorithm, see section 2.2.5). Instead, they work as benchmarks and assist readers to gain a better understanding of optimality of the (POD-) Greedy procedure by providing a schematic illustration of maximum error convergence (fig. 2.5) and the error reduction (fig. 2.8).

Example setting Figure 2.4 introduces the geometry, notation and material parameter of the beam model. The FE mesh consists of 527 nodes and 920 linear triangular elements. The quantity of interest is defined as the displacement of the inclusion. A point force is applied at the centre node of the beam upper edge.

A training set $\mathcal{P}^{\text{train}}$ is created and moderately discretised into $\{\mu_i\}_{i=1}^{129}, N_{\text{train}} = 129$ samples. μ in the central region ranges in $[10^{-1}\ 10^{1}]$ logarithmically. μ remains 10^0 in the matrix, thus μ_i in the central region is the sole parameter in this example. The first magic point is chosen randomly as $\mu_1 = 10^0$, which is the middle point of the 1d parametric domain. The *weak nonintrusive technique* is utilised in this model (see appendix A.2), the model is generated and meshed in Abaqus, all information are imported and processed in Matlab.

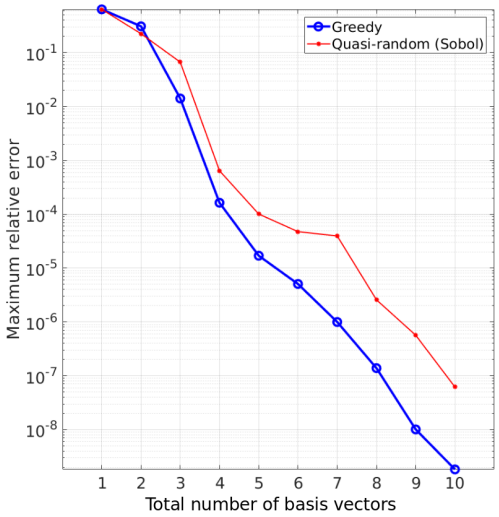
The beam model is selected to demonstrate the Greedy procedure due to its simple solution structure: for each training point, the exact solution can be obtained, such that the true RB-error (eq. (2.21)) can be computed and used as an error indicator. Equation (2.24) is used to obtain the error via the error-residual relation. The relationship between maximum relative true-RB error and total number of basis vectors are plotted as convergence, see fig. 2.5.

Comparison with statistically based sampling approaches The Greedy procedure are compared with the following statistically based sampling methods: pseudorandom sampling, uniform systematic sampling, Quasi-random (Sobol sequence) sampling, Latin Hypercube sampling. The desired output of Greedy procedure and other sampling approaches is the magic point set $\mathcal{P}^{M}$ and maximum relative error convergence. To ensure the tests are under equal conditions, the statistically based sampling approaches are performed in a Greedy manner: the n^{th} magic point μ_n is the output of the n^{th} iteration. This principle is applied to all comparison tests in this thesis (section 2.2.5, section 7.1, section 7.2). The difference among sampling approaches is: in other sampling approaches, the magic points may be generated prior to the search, while in the

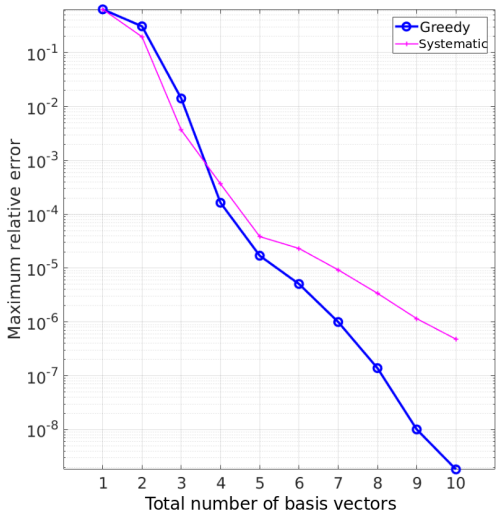
Greedy procedure the n^{th} magic point remains unknown until the n^{th} Greedy iteration is finished. For testing purpose, 10 iterations are performed for all tests regardless of the given error tolerance tol^{err}. In order to ensure generality, random type methods (pseudo-random, Latin Hypercube Sampling) are repeated for 5 times. Notice that in the Greedy procedure, an important nature is that no magic point will be selected more than once; however, for non-Greedy type methods, this principle might be violated (repeat selection of magic point), results in zero error reduction. In order to prevent this from happening, a safety check is set for statistically based methods: if a repeated magic point is detected, dump it and choose a new magic point until all magic points are unique.

Numerical results The error response surfaces, magic point locations and maximum error convergence are displayed in fig. 2.8. Some remarks can be made based on the results:

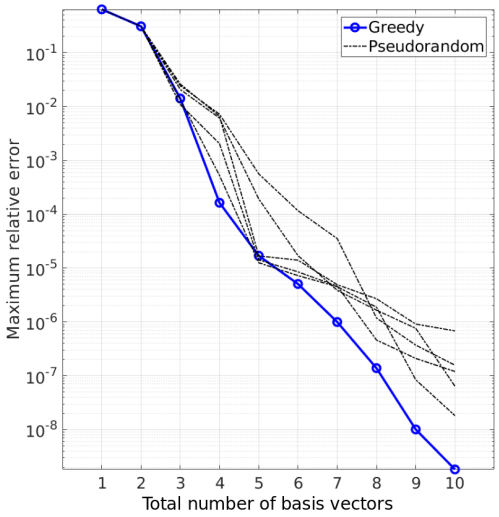
- For static RB problems, adding a single mode to the basis nicely reduces the error to zero.
- Due to the thorough error reduction, a magic point would not be selected more than once.
- The maximum error convergence of Greedy procedure is more rapid than other sampling approaches.
- The maximum error generally decreases with the basis enrichment for all cases. However, the convergence depends on the performance of the reduced basis. The enrichment of the reduced basis does not just reduce the error at magic point to zero, but also brings down the error at other parameter points. This effect weakens as the distance between the magic point and parameter point increases. If this distance is large, the overall reduction might be insufficient, which results in a larger maximum error than the previous one.



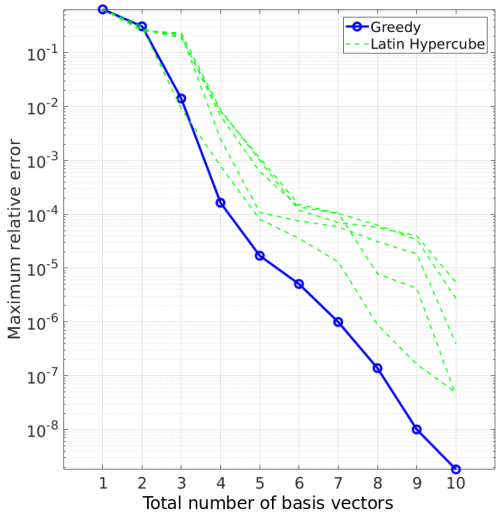
(a) Greedy vs Sobol sequence



(b) Greedy vs systematic sampling (uniform grid)



(c) Greedy vs pseudorandom sampling



(d) Greedy vs Latin Hypercube sampling

Figure 2.5: Compare Greedy procedure with other sampling approaches, Greedy procedure shows the most rapid convergence.

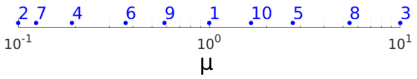
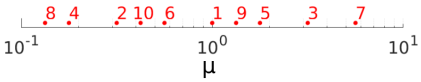
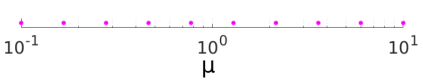


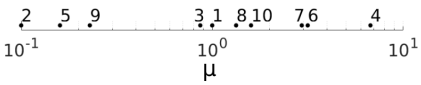
Figure 2.6: Greedy magic points, digits above the samples denote Greedy iterations.



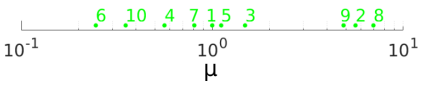
(a) Sobol samples



(b) Systematic samples (uniform grid)



(c) Pseudorandom samples



(d) Latin Hypercube samples

Figure 2.7: Samples obtained from statistically based approaches for section 2.2.4, digits on top of samples denote iteration numbers. For systematic sampling, showing the 10 samples from the last iteration due to application of global-POD.

2.2.5 POD-Greedy algorithm for linear dynamic problems

Greedy algorithm and POD are popularly applied methods for basis construction. They both produce reduced bases in an optimum sense, and the bases possess a hierarchical structure, i.e. $\mathbf{\Phi}_j \subset \mathbf{\Phi}_k$ for $j \leq k$. Both methods require to compute a sequence of exact solutions, i.e. snapshots, thus are not necessarily inexpensive in terms of computational costs. However, they are different in these aspects: (i) Greedy algorithm produces a Lagrangian space spanned by snapshots, while the reduced space generated by POD is not Lagrangian; (ii) POD aims at minimizing the squared projection error (in an average sense), while Greedy algorithm minimizes the maximum projection error. Greedy algorithm constructs the snapshot dataset, and POD compresses the snapshot into a representative, compact basis. Hence in linear dynamic problems, these two methods might be suitably combined, i.e. select magic points in a Greedy manner and compress the as-

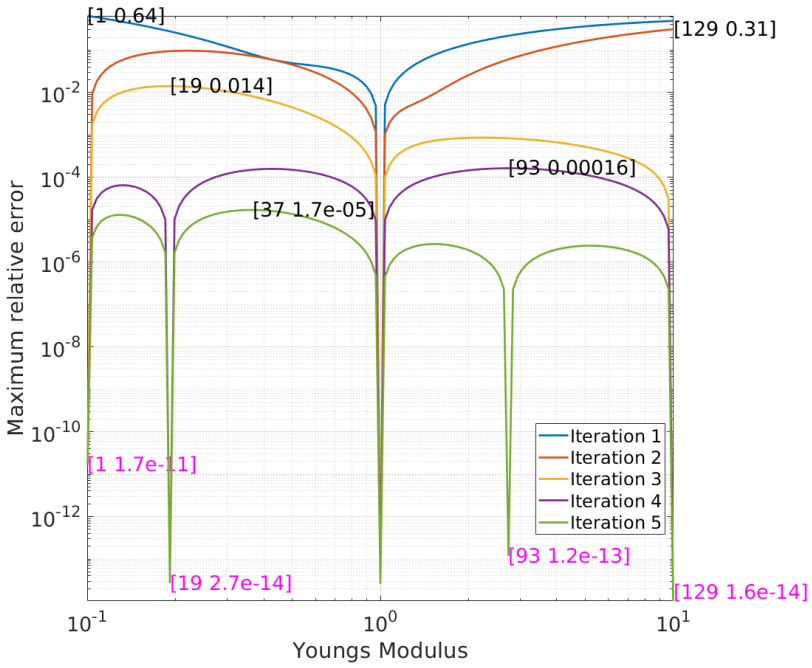


Figure 2.8: The first 5 error response surfaces and associated magic point locations from the Greedy procedure. The 2 digits in the square brackets are sample index and maximum error value, respectively. The digits in pink denotes the relative error at the magic point after the Greedy reduction. Visually it can be seen that no magic point is selected more than once due to the thorough reduction of error in static problems.

sociated snapshots with POD, such that advantages of both methods can be maximised. The combined method is called the **POD-Greedy algorithm** [30, 33, 42, 47, 51, 59], which acts as a standard approach in ROM for time-dependent problems.

POD-Greedy algorithm reduces the maximum projection error by adaptively searching for Greedy samples (same as the Greedy procedure). Once a full snapshot trajectory is revealed, POD is applied with respect to time to compress the projection error into a compact basis, and this new basis is added to the current one to achieve the enrichment.

Standard POD-Greedy Algorithm The standard POD-Greedy method is introduced in algorithm 2. The structure of POD-Greedy algorithm is similar to the Greedy procedure in the following aspects: (i) the training set $\mathcal{P}^{\text{train}} \subset \mathcal{P}$ is discretised into N_{train} samples; (ii) the projection error $\boldsymbol{e}_{\text{proj}}$ is being used as the data for basis enrichment; (iii) a suitable error indicator is required to direct the linear search of the reduced basis; (iv) the POD-Greedy algorithm is also divided into a Greedy *basis processing* stage and a Greedy *parameter sweep* stage. The main differences are (i) the snapshots are time-dependent; (ii) POD is an additional ingredient. See algorithm 2 for the pseudo-code.

Similar to section 2.2.4, detailed descriptions of different stages of a ROM approximation with standard POD-Greedy algorithm are given as "*offline*" stage:

- ROM "*offline*" stage: basis construction
  - POD-Greedy *parameter sweep*: evaluate error for every single parameter value of $\mathcal{P}^{\text{train}}$, step 4-5.
  - POD-Greedy *basis processing*: find and compress snapshot to obtain $\boldsymbol{\Phi}$, step 6-12.
- ROM "*online*" stage: calculations of $\boldsymbol{U}^{\text{r}}(\mu)$ for single or multiple parameter values of interest μ.

Algorithm 2 The standard POD-Greedy algorithm

Input: tol^{err} (User defined error tolerance)

Output: N, X_N, $\mathcal{P}_N$, $\boldsymbol{\Phi}$

1: Choose $\mu_1 \in \mathcal{P}^{\text{train}}$, initialize $N = 1$, set $\mathcal{P}_1 = \{\mu_1\}$, $X_1 = \text{span}\{\boldsymbol{U}(\mu_1)\}$ ▷ Initialisation

2: Define $\boldsymbol{e}_{\text{gr}} := \max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$

3: **while** $\boldsymbol{e}_{\text{gr}} \geq \text{tol}^{\text{err}}$ **do**

4: **for** $\mu \in \mathcal{P}^{\text{train}}$ **do**

compute $\boldsymbol{e}_{\text{ind}}(\mu)$ ▷ Greedy *parameter sweep*

5: **end for**

6: $\mu_{n+1} = \arg\max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$ ▷ New samples (magic point)

7: $\boldsymbol{e}_{\text{proj}}(\mu_{n+1}) := \boldsymbol{U}(\mu_{n+1}) - \boldsymbol{\Phi}_n \boldsymbol{\Phi}_n^T \boldsymbol{U}(\mu_{n+1})$

8: $\phi_{n+N^{\text{add}}} := POD_{N^{\text{add}}}\left(\boldsymbol{e}_{\text{proj}}(\mu_{n+1})\right)$ ▷ POD

9: $\boldsymbol{\Phi}_{n+N^{\text{add}}} := \boldsymbol{\Phi}_n \bigcup \phi_{n+N^{\text{add}}}$ ▷ Possible multiple basis vectors

10: $\mathcal{P}_{N+1} := \mathcal{P}_N \bigcup \{\mu_{n+1}\}$ ▷ Only one new magic point

11: $X_{N+N^{\text{add}}} := X_N \bigoplus \phi_{n+N^{\text{add}}}$

12: Set $N := N + N^{\text{add}}$

13: **end while**

Here N^{add} denotes the number of newly added basis vectors. This number is not necessarily 1, but can be chosen as a fixed number, or to be calculated according to an error reduction tolerance, e.g. SVD truncation error, true RB-error, projection error, etc. If using N to denote the total number of basis vectors after the entire POD-Greedy algorithm, it can be seen that complexity N of the RB approximation remains unknown until the Greedy process is finished. For step 8, when constructing the reduced basis via the projection error at magic point μ_n, if an error reduction tolerance e_{r}^{to} is to be used, algorithm 3 might be applied to determine the number of newly added basis vectors.

Again the following clarification is made to differentiate POD-Greedy algorithms in this thesis: a POD-Greedy algorithm with any error indicators is named as the *standard POD-Greedy* algorithm; if equipped with true RB-error, the algorithm is named as the *reference POD-Greedy* algorithm (as opposed to the *proposed POD-Greedy* algorithm in the subsequent chapters), which is considered as the optimum benchmark in this thesis.

Algorithm 3 The RB enrichment algorithm (determination of N^{add})

Input: e_{r}^{to} (User defined enrichment tolerance)

Output: N^{add}

1: Initialize $N^{\text{add}} = 1$ ▷ Initialisation

2: Define $\boldsymbol{e}(\mu_n) := \|\boldsymbol{U}(\mu_n) - \boldsymbol{U}^{\text{r}}(\mu_n)\|_{\text{F}}$

3: **while** $\boldsymbol{e}(\mu_n) \geq e_{\text{r}}^{\text{to}}$ **do**

4: $\quad \boldsymbol{e}_{\text{proj}}(\mu_n) := \boldsymbol{U}(\mu_n) - \boldsymbol{\Phi}_n \boldsymbol{\Phi}_n^T \boldsymbol{U}(\mu_n)$

5: $\quad \boldsymbol{\phi}_{n+N^{\text{add}}} := POD_{N^{\text{add}}}\left(\boldsymbol{e}_{\text{proj}}(\mu_n)\right)$ ▷ POD

6: $\quad \boldsymbol{\Phi}_{n+N^{\text{add}}} := \boldsymbol{\Phi}_n \bigcup \boldsymbol{\phi}_{n+N^{\text{add}}}$ ▷ Basis enrichment

7: $\quad$ Set $N^{\text{add}} := N^{\text{add}} + 1$

8: **end while**

POD-Greedy algorithm numerical example and results

Example setting In order to demonstrate the *reference POD-Greedy* algorithm (using true RB-error as an error indicator), the same simple fixed beam model as introduced in section 2.2.4 is inherited. A few modifications are made as follows: (i) a dynamic force is applied at the same node with amplitude shown in fig. 2.9; (ii) for time integration, the total time is set as $T = 4.9s$ and time step length is set as $\Delta t = 0.1s$, which results in 50 time steps; (iii) Rayleigh damping coefficient b is added as the second parameter. With affine parameter dependence, the damping matrix is generated by affine expansion, i.e. $\boldsymbol{C}_j := b\boldsymbol{K}_j$, where $\boldsymbol{K}_j$ is the j^{th} affine stiffness operator. In this example, the parameters only varies in the central inclusion, thus $\boldsymbol{C}_{\text{inclusion}} := b\boldsymbol{K}_{\text{inclusion}}$. The parameter domain $\mathcal{P} := [10^{-1}, 10^{1}] \times [10^{-1}, 10^{1}]$. A training set $\mathcal{P}^{\text{train}}$ is created with a moderate discretisation: $\{\mu_i, b_j\}_{i,j=1}^{17,17}$, results in $N_{\text{train}} = 289$ samples. The initial magic point, is randomly chosen as $[\mu_1, b_1] = [10^{-1}, 10^{-1}]$. Again for testing purpose, 10 Greedy iterations are performed and the *reference POD-Greedy* algorithm is compared with other sampling methods. These sampling approaches include Quasi-random sampling (Halton and Sobol sequences), pseudorandom sampling, Latin Hypercube sampling. In order to ensure generality, random-type approaches (pseudorandom and Latin Hypercube) are repeated for 5 times. The magic points are searched hierarchically in a Greedy manner, i.e. generate $[\mu_1, b_1]$ in first iteration, $[\mu_2, b_2]$ in the second iteration, etc. The settings of other sampling methods in this example are adapted from section 2.2.4.

Algorithm 3 is applied to determine the number of basis vectors being added at each Greedy iteration, where $e_{\text{r}}^{\text{to}} = 0.6$ is set as the error reduction ratio at the magic points, as a results, each iteration may generate different number of basis vectors. The true RB-error (eq. (2.21)) is calculated and used as an error indicator. Equation (2.25) is used to compute the error via the error-residual relation.

Numerical results The following remarks are made from the experiment results:

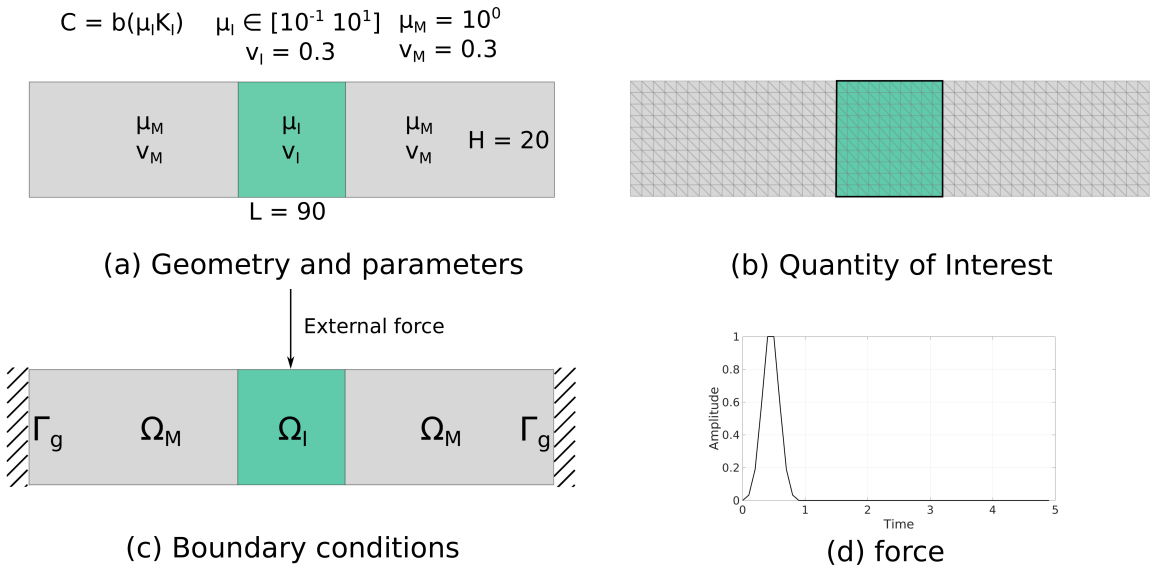


Figure 2.9: The 2D beam model

- Results suggest that the output RB-space X_N is much larger, while the overall convergence is much slower than the Greedy procedure (maximum error exceeds 10^{-8} in Greedy procedure while only 10^{-2} is achieved in POD-Greedy algorithm). This slow convergence somehow fits our expectation due to the parameter and time complexity of the dynamic problem.

- The application of POD results in 2 aspects: (i) the error at each magic point is not necessarily reduced to 0 (one may force the error to be reduced to 0, however the resulting basis would be significantly large); (ii) due to the partial reduction of error, a magic point might be selected more than once in POD-Greedy algorithm, which simply implies that this particular magic point location weights more than others.

- The regions of large error are shown in fig. 2.10, where results indicates that it is more difficult to approximate the less damped regions in the parametric domain.

In fact 9 out of 10 Greedy magic points are selected at $b = 10^{-1}$. This is possibly because the higher frequencies are damped by the stiffness proportional damping, leaving only the lower frequency modes, which are the key components of the reduced basis. As a result, highly-damped regions possess low error.

- POD-Greedy algorithm converges more rapidly than all statistically based approaches in this experiment. Statistically based sampling approaches may reach the same maximum relative error as the *reference POD-Greedy* method, however the resulting reduced bases are much larger. In several cases when the bases are small, the *reference POD-Greedy* method is not necessarily the optimum, i.e. it converges slower than other sampling approaches (see fig. 2.10: $N < 4$ in Halton and Sobol, $N < 11$ in pseudorandom, $N < 6$ in Latin hypercube). This is completely normal due to the fact that error at magic point is not reduced to 0 in *reference POD-Greedy* method.

2.2.6 ROM for parametric dynamic problems

Large-scale dynamic systems are often need to be simulated to study the nature of complex physical phenomena. When the system is parametric, the computational cost can be prohibitive due to high-demands of each individual simulation. This burden might be alleviated by applying ROM to generate and solve low-dimensional reduced order models. This thesis focuses on dealing with large-scale dynamic problems, thus here state-of-art projection-based ROM methods are reviewed for parametric dynamic problems [2, 33, 43, 63, 64, 65, 66, 96, 97]. A POD-Greedy algorithm is presented in [33] where an 'hp'-refinement is applied to perform rapid online evaluation of parametric parabolic PDEs. The parameter domain is partitioned into a set of subdomains by an 'h'-refinement and the approximation space is then expanded by a 'p'-refinement. This 'hp' certified RB method results in higher "*offline*" cost and "*online*" computational savings, the latter is relatively important in terms of real-time applications. The domain

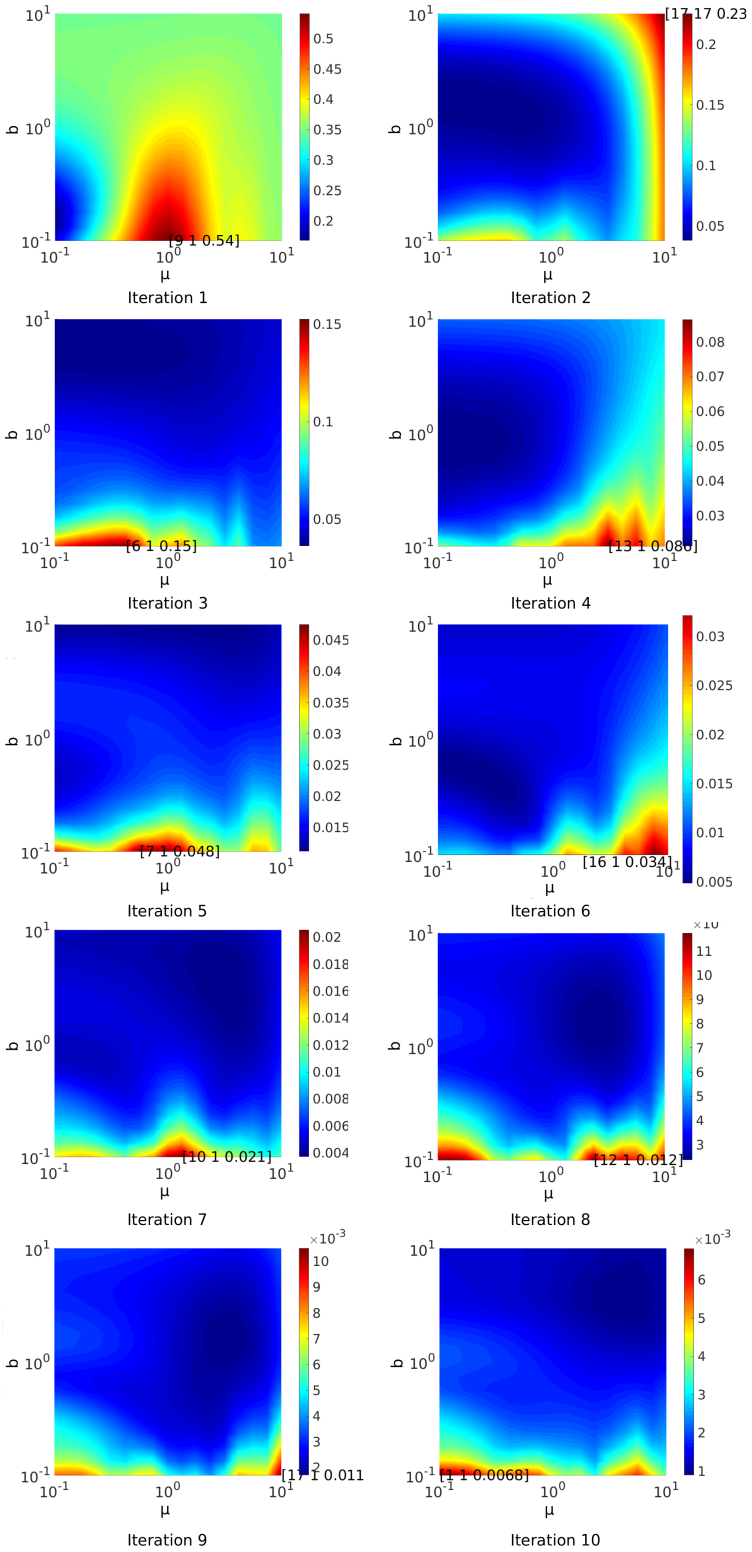
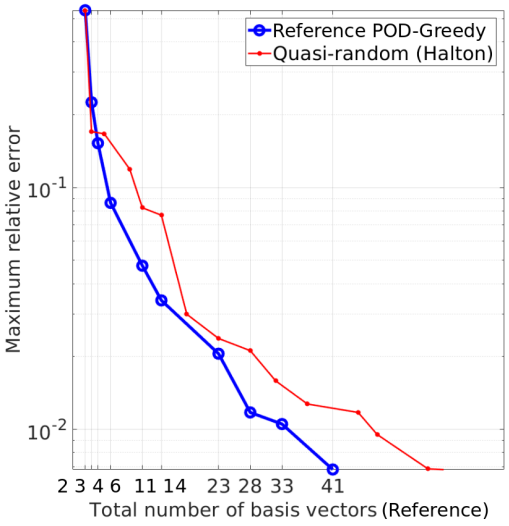
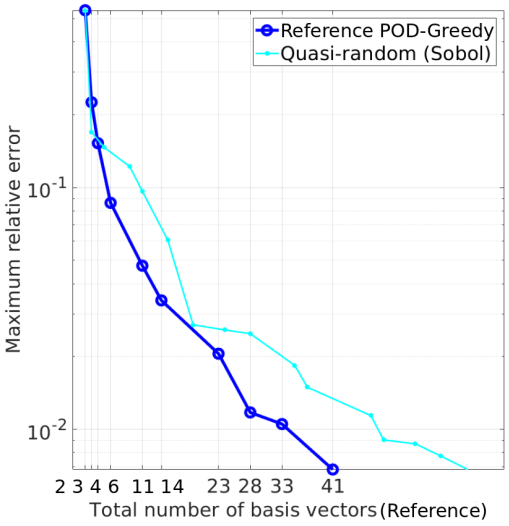


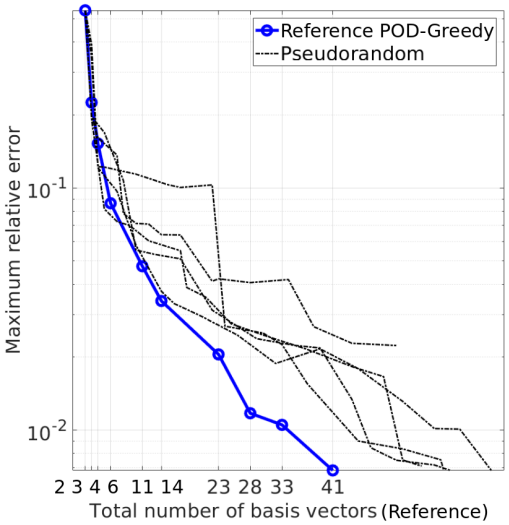
Figure 2.10: The error response surfaces for the first 10 POD-Greedy iterations.



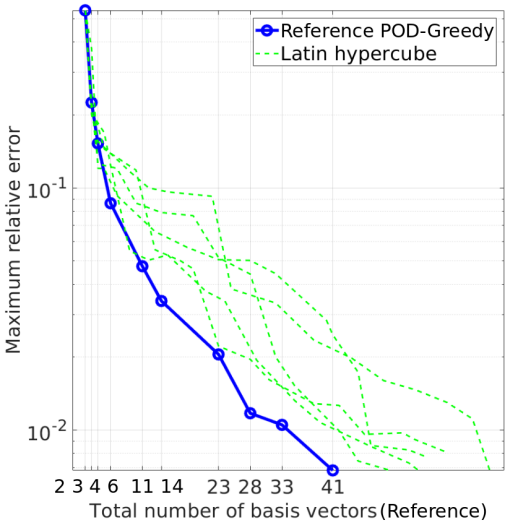
(a) POD-Greedy vs Halton sequence



(b) POD-Greedy vs Sobol sequence



(c) POD-Greedy vs pseudorandom sampling



(d) POD-Greedy vs Latin Hypercube sampling

Figure 2.11: Compare the *reference POD-Greedy* algorithm with other sampling approaches, the former shows the most rapid convergence.

partitioning technique is applied in [43], which allows to use arbitrary time steps in each subdomain. This study considers local scale properties in different structure zones in both space and time. A two-time-scale (coarse and fine time scales) approach allows the iterations to be performed on the coarse time scale. Furthermore, number of iterations in the coarse time intervals are reduced due to application of two space scales (coarse and fine mesh).

The spectral function type of methods which aims at predicting transient response of randomly parametric structural dynamic systems are proposed in [63, 66, 65], where the reduced space is spanned by a set of orthogonal eigenbasis, namely spectral functions. The eigenbasis are eigenvectors of system matrix equipped with diagonal terms as preconditioner, therefore physical vibration modes can be used as eigenmodes. In [66], direct Monte-Carlo Simulation (MCS) is taken as the benchmark solution and Polynomial Chaos (PC) approach is presented as comparisons to show the effectiveness of the proposed method. Numerical results indicate that using high-order spectral modes agree well with direct MCS, while fourth-order PC method fails to reproduce the outputs. Another application of the spectral function method is proposed in [65], where author proves that higher order spectral functions lead to more accurate approximations, with a higher computational cost. This high cost is alleviated by application of a Bayesian metamodel, results in a hybrid approach. Again numerical examples show good match of responses between spectral function method and direct MCS as a benchmark. The spectral function method extends its application to dynamic control problems in conjunction with Balanced truncation method in [64], where principle modes of the controllability Gramian are utilised to construct the low-dimensional subspace.

2.3 Nonintrusive techniques in ROM

Nonintrusive techniques can be applied in both ROM "*offline*" and "*online*" stages. More specifically, in ROM "*offline*" stage, computations of exact solutions by source

code can be replaced by utilisation of commercial software or open-source code packages, e.g. Abaqus, ANSYS, Nastran, Fluidity; in ROM "*online*" stage, varieties of numerical methods can be used to compute the unknown reduced variable to suitably replace the standard Galerkin approach, such as radial basis function (RBF) approximation method [4, 108, 117], neural network, Smolyak sparse grid collocation method [74, 116], least square fitting method [74], etc.

First the nonintrusive applications in ROM "*online*" stage are addressed in this section. Standard ROM requires a Galerkin projection to construct the reduced system and to compute the unknown reduced variables (or namely POD coefficients in [74, 116]). A nonintrusive algorithm is developed in the above 2 researches to improve such computations. Instead of using the Galerkin approach, the reduced variables are interpolated on a Smolyak sparse grid such that the number of nodes is only a polynomial function of dimensions. As a result, the curse of dimensionality is cured. Another least square fitting approach is proposed in [74] to compute the reduced variables. Advantage of such approach is that linear solvers are used to replace full simulations, thus the approach is nonintrusive and the computational cost is reduced. A second-order Taylor series method is applied in [116]: the POD coefficients at new time steps are calculated with a second order Taylor expansion. This implementation does not require to solve the governing equation of the original system for parametric problems thus are easy to be implemented. Radial basis function (RBF) method is popularly utilised as a nonintrusive approach to fit the nonlinear cases when Galerkin approach is inappropriate. An example of application can be found in [4], where ROM is constructed for time-dependent problems with parametric governing equations, boundary and initial conditions. This is achieved through a 2-level approach: POD snapshots are obtained from solving the problem at a set of parameter points, and the POD coefficients are approximated with the RBF approach. A comprehensive review of nonintrusive ROM in finite element based structural problems can be found in [81], in which comparisons of outputs between ROM source code and commercial software (Nastran and Abaqus) are given to show close or

matching results. For nonintrusive applications in ROM "*offline*" stage, [121] introduces a nonintrusive PGD scheme using Abaqus to solve linear elasto-mechanical problems externally. In this research the stiffness matrices are exported from Abaqus and imported into Matlab to solve the inexpensive parametric problems, while the expensive mechanical problems are isolated and solved in Abaqus to obtain the spatial unknowns. Once finished these unknowns are imported into Matlab as reduced basis. Notice that a detailed procedure of nonintrusive technique utilising Abaqus is introduced in this work.

2.4 Metamodel based stochastic analysis for high-dimensional parametric problems

Surrogate based uncertainty quantification algorithm plays as an important tool in terms of determining the input-output (IO) relation of high-dimensional, many parameter systems.The repetitive feature of parametric problems requires utilisation of model approximations, i.e. model of the model (such as ROM of the FE model) [26]. These are often referred to as metamodels. Varies of methods have been developed to tackle this type of problems. Based on the fact that the input variables are often correlated, the high dimensional model representation (HDMR) [70, 71, 72, 73, 109] is an effective approach to deal with high-dimensional input-output systems. When learning the IO behaviour of high-dimensional systems, HDMR techniques can be applied to reduce the sampling effort by decomposing the output variance. Typically response surfaces are generated to evaluate the nonlinear functional behaviour between the input and output. The model output is expressed as an expansion in terms of the input variables, and a second order expansion is often satisfying for many high-dimensional systems. The high-dimensional integrals in the expansion may be carried out by random sampling techniques, thus namely RS (random sampling)-HDMR. When the number of samples are moderate, RS-HDMR is an efficient tool to provide reliable global uncertainty assessments [73]. The high-order terms in RS-HDMR can be approximated as products of low-order functions,

namely *low-order term* product (lp)-RS-HDMR, hence direct evaluation of high-order terms can be circumvented, and the sampling effort can be as well reduced [70]. However this approximation violates the orthogonality properties of the components, thus to overcome this, orthonormal polynomial approximations are applied [71]. The accuracy is also improved by preserving the orthogonality. Polynomial chaos expansion (PCE) [19, 77, 111, 113, 120] constructs a stochastic space spanned by a set of orthogonal bases, such that the random variables can be projected on. The orthogonal bases might be generated by Gram Schmidt orthogonalisation. Instead of applying standard Monte Carlo simulation which may require large number of simulations, Latin Hypercube sampling might be used in conjunction with PCE [19] to find the key component in uncertainty quantification.

For complex system, Artificial neural network (ANN) works as an efficient representation [25, 41, 79, 90]. Basic elements of ANN are neurons, which combine inputs and perform non-linear operations to obtain the final results. Benefits of ANN is that prior specification of fitting function is not needed, and almost all kinds of non-linear functions can be approximated with ANN. Moving least squares (MLS) [10, 57, 69, 118] employs a weighted interpolation function to reconstruct the continuous response surfaces. During this reconstruction process, a set of unorganised, experimental points are selected and fitted using a 2nd order polynomial, such that the weighted least-square error can be minimised. High-dimensional data can be modelled by flexible regression applying Multivariate adaptive regression splines (MARS) [23, 37, 84, 110]. By choosing a set of piecewise linear basis functions to approximate the response function, MARS produces continuous models with continuous derivatives. Moreover, knot of MARS are allowed to bend, thus complex behaviours of the function can be modelled. MARS is considered to be accurate as well as cost-effective in terms of metamodel construction. When the input data is scattered, i.e. grid data is absence, Radial based function (RBF) [11, 31, 48, 86] which interpolates surrogates can be used to model multivariate functions. RBF is particularly attractive in many cases as it is applicable independent of dimension. The

interpolation is a linear combination of weights and basis functions, where the functions fit exactly at the given sample points. RBF can be applied in ANN as active functions, called Radial basis function network. RBF has its shortcoming as the metamodel may change significantly with different basis function and/or parameters.

2.5 Summary

In this chapter popular ROM techniques have been reviewed, including Modal analysis and POD for non-parametric problems (a-priori methods), snapshot-POD, global-PODGreedy procedure and POD-Greedy algorithm for parametric problems (a-posteriori methods). In snapshot-POD, statistically based sampling approaches are reviewed, which may be applied during sampling of the snapshots. In Greedy procedure, different error indicators are introduced, then a simple fixed beam example under static load is used to demonstrate its optimality in terms of maximum error convergence (section 2.2.4). The same beam model with dynamic load is used in section 2.2.5, where POD-Greedy algorithm application is investigated. In both tests statistically based approaches are presented as comparisons. These tests are important as they aid readers to understand the theoretical foundations of these thesis: POD, Greedy procedure and POD-Greedy sampling algorithm. Finally metamodels for high-dimensional parameter problem are reviewed, providing a broader introduction for this area of work.

Chapter 3

Full Space-time Representation of the Newmark Method

In this chapter a full space-time representation based on the classical Newmark Method is introduced. The Newmark method is an implicit, iterative method to solve dynamic problems in a step-by-step manner. However, with the full space-time representation one can see that the dynamic problem may be solved in an elliptic way after an assembly process. The representation is crucial in this thesis in terms of derivation of the new error indicator, the proposed POD-Greedy algorithm and the "*error in the error*" *indicator* introduced in the subsequent chapters. First the full representation for the exact problem is derived in section 3.2, then this is extended to reduced order problems in section 3.3.

3.1 Preliminary: vectorisation of space-time responses

Given $\underline{\boldsymbol{u}}^{\mathrm{h}}(t_n) = \underline{\boldsymbol{u}}^{\mathrm{h}}_n$, FE space-time acceleration, velocity, displacement and force matrix are defined as follows:

$$\begin{aligned}
\ddot{\boldsymbol{U}}^{\mathrm{h}}(t) &= \left[\underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}}_0 \quad \cdots \quad \underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}}_n \quad \cdots \quad \underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}}_{N_t}\right], t \in [0,T] \\
\dot{\boldsymbol{U}}^{\mathrm{h}}(t) &= \left[\underline{\dot{\boldsymbol{u}}}^{\mathrm{h}}_0 \quad \cdots \quad \underline{\dot{\boldsymbol{u}}}^{\mathrm{h}}_n \quad \cdots \quad \underline{\dot{\boldsymbol{u}}}^{\mathrm{h}}_{N_t}\right], t \in [0,T] \\
\boldsymbol{U}^{\mathrm{h}}(t) &= \left[\underline{\boldsymbol{u}}^{\mathrm{h}}_0 \quad \cdots \quad \underline{\boldsymbol{u}}^{\mathrm{h}}_n \quad \cdots \quad \underline{\boldsymbol{u}}^{\mathrm{h}}_{N_t}\right], t \in [0,T] \\
\boldsymbol{F}^{\mathrm{h}}(t) &= \left[\underline{\boldsymbol{f}}^{\mathrm{h}}_0 \quad \cdots \quad \underline{\boldsymbol{f}}^{\mathrm{h}}_n \quad \cdots \quad \underline{\boldsymbol{f}}^{\mathrm{h}}_{N_t}\right], t \in [0,T]
\end{aligned} \tag{3.1}$$

where $\underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}}_n$, $\underline{\dot{\boldsymbol{u}}}^{\mathrm{h}}_n$ and $\underline{\boldsymbol{u}}^{\mathrm{h}}_n$ denote acceleration, velocity and displacement vector at time step n, respectively. Performing vectorisation, there is (see fig. 3.1 for vectorisation of acceleration, velocity and displacement):

$$\begin{aligned}
\underline{\ddot{\boldsymbol{U}}}^{\mathrm{h}} &= \left[\underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}T}_0 \quad \cdots \quad \underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}T}_n \quad \cdots \quad \underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}\ T}_{N_t}\right]^T, \\
\underline{\dot{\boldsymbol{U}}}^{\mathrm{h}} &= \left[\underline{\dot{\boldsymbol{u}}}^{\mathrm{h}T}_0 \quad \cdots \quad \underline{\dot{\boldsymbol{u}}}^{\mathrm{h}T}_n \quad \cdots \quad \underline{\dot{\boldsymbol{u}}}^{\mathrm{h}\ T}_{N_t}\right]^T, \\
\underline{\boldsymbol{U}}^{\mathrm{h}} &= \left[\underline{\boldsymbol{u}}^{\mathrm{h}T}_0 \quad \cdots \quad \underline{\boldsymbol{u}}^{\mathrm{h}T}_n \quad \cdots \quad \underline{\boldsymbol{u}}^{\mathrm{h}\ T}_{N_t}\right]^T, \\
\underline{\boldsymbol{F}}^{\mathrm{h}} &= \left[\underline{\boldsymbol{f}}^{\mathrm{h}T}_0 \quad \cdots \quad \underline{\boldsymbol{f}}^{\mathrm{h}T}_n \quad \cdots \quad \underline{\boldsymbol{f}}^{\mathrm{h}\ T}_{N_t}\right]^T
\end{aligned} \tag{3.2}$$

define space and time response vector at time step n as follows:

$$\underline{\boldsymbol{x}}^{\mathrm{h}}_n = \left[\underline{\ddot{\boldsymbol{u}}}^{\mathrm{h}T}_n \quad \underline{\dot{\boldsymbol{u}}}^{\mathrm{h}T}_n \quad \underline{\boldsymbol{u}}^{\mathrm{h}T}_n\right]^T \tag{3.3}$$

thus for the entire time domain, the FE *full response vector* reads:

$$\underline{\boldsymbol{X}}^{\mathrm{h}} = \left[\underline{\boldsymbol{x}}^{\mathrm{h}T}_0 \quad \cdots \quad \underline{\boldsymbol{x}}^{\mathrm{h}\ T}_{N_t}\right]^T \tag{3.4}$$

dimension of $\underline{\boldsymbol{X}}^{\mathrm{h}}$ is very large, it is emphasised that it is only necessary for derivations

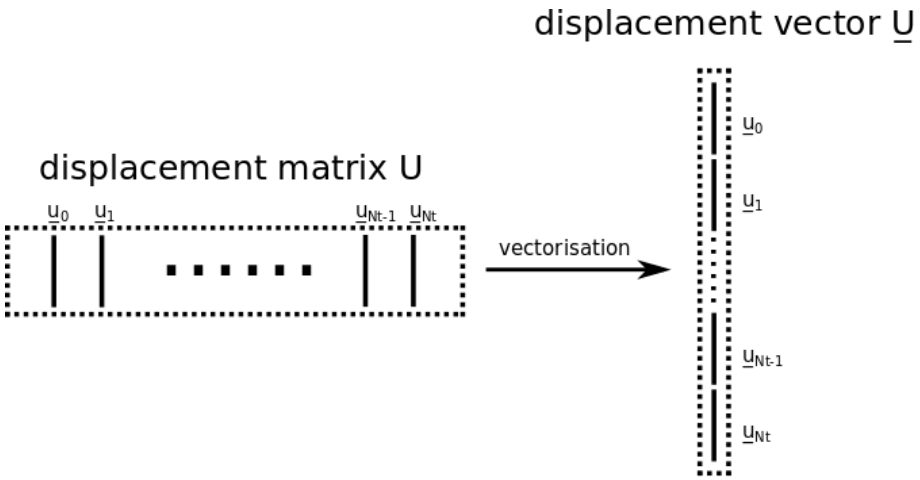


Figure 3.1: Vectorisation of displacement matrix.

in Chapter 4, and will not be used in solution of real problems. Schematically see fig. 3.2 for vectorisation of full response vector.

3.2 The full-scale case

A set of coefficients are defined for Newmark method as follows:

$$a_0 = \frac{1}{\beta \Delta t^2},\ a_1 = \frac{\gamma}{\beta \Delta t},\ a_2 = \frac{1}{\beta \Delta t},\ a_3 = \left(\frac{1}{2\beta} - 1\right),\ a_4 = \left(\frac{\gamma}{\beta} - 1\right), a_5 = \frac{\Delta t}{2}\left(\frac{\gamma}{\beta} - 2\right)$$

thus eq. (1.7) becomes:

$$\begin{cases} \underline{\dot{\boldsymbol{u}}}_n^{\mathrm{h}} = a_1 \underline{\boldsymbol{u}}_n^{\mathrm{h}} - a_1 \underline{\boldsymbol{u}}_{n-1}^{\mathrm{h}} - a_4 \underline{\dot{\boldsymbol{u}}}_{n-1}^{\mathrm{h}} - a_5 \underline{\ddot{\boldsymbol{u}}}_{n-1}^{\mathrm{h}},\ 0 \le n \le N_t \\ \underline{\ddot{\boldsymbol{u}}}_n^{\mathrm{h}} = a_0 \underline{\dot{\boldsymbol{u}}}_n^{\mathrm{h}} - a_0 \underline{\boldsymbol{u}}_{n-1}^{\mathrm{h}} - a_2 \underline{\dot{\boldsymbol{u}}}_{n-1}^{\mathrm{h}} - a_3 \underline{\ddot{\boldsymbol{u}}}_{n-1}^{\mathrm{h}},\ 0 \le n \le N_t \end{cases} \tag{3.5}$$

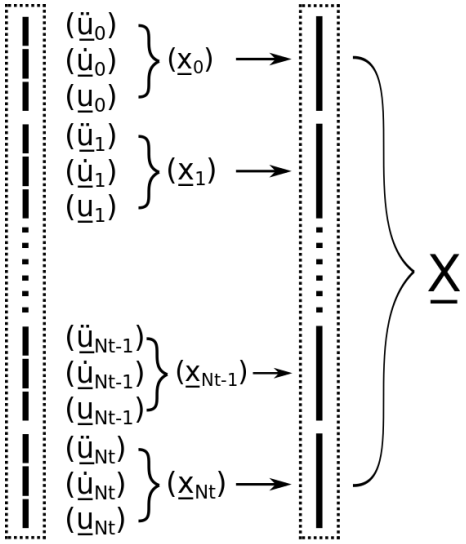


Figure 3.2: Vectorisation of full response vector.

couple eq. (3.5) with eq. (1.6) and rewrite in matrix form:

$$\underbrace{\begin{bmatrix} \boldsymbol{M} & \boldsymbol{C} & \boldsymbol{K} \\ \mathbf{0} & \mathbf{I} & -a_1\mathbf{I} \\ \mathbf{I} & \mathbf{0} & -a_0\mathbf{I} \end{bmatrix}}_{\boldsymbol{H}^s} \underbrace{\begin{bmatrix} \ddot{\underline{\boldsymbol{u}}}_n^{\mathrm{h}} \\ \dot{\underline{\boldsymbol{u}}}_n^{\mathrm{h}} \\ \underline{\boldsymbol{u}}_n^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{x}}_n^{\mathrm{h}}} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ a_5\mathbf{I} & a_4\mathbf{I} & a_1\mathbf{I} \\ a_3\mathbf{I} & a_2\mathbf{I} & a_0\mathbf{I} \end{bmatrix}}_{\boldsymbol{H}^f} \underbrace{\begin{bmatrix} \ddot{\underline{\boldsymbol{u}}}_{n-1}^{\mathrm{h}} \\ \dot{\underline{\boldsymbol{u}}}_{n-1}^{\mathrm{h}} \\ \underline{\boldsymbol{u}}_{n-1}^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{x}}_{n-1}^{\mathrm{h}}} = \underbrace{\begin{bmatrix} \underline{\boldsymbol{f}}_n^{\mathrm{h}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}}_{\underline{\boldsymbol{g}}_n^{\mathrm{h}}} \tag{3.6}$$

where $\mathbf{I} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ denotes the square identity matrices and $\mathbf{0}$ denotes all-zero matrices and vectors. Hence the general form for time step n can be written as:

$$\boldsymbol{H}^s \underline{\boldsymbol{x}}_n^{\mathrm{h}} + \boldsymbol{H}^f \underline{\boldsymbol{x}}_{n-1}^{\mathrm{h}} = \underline{\boldsymbol{g}}_n^{\mathrm{h}} \tag{3.7}$$

in addition, initial condition is introduced as follows:

$$\underbrace{\begin{bmatrix} \boldsymbol{M} & \boldsymbol{C} & \boldsymbol{K} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\widetilde{\boldsymbol{H}}^s} \underbrace{\begin{bmatrix} \underline{\ddot{\boldsymbol{u}}}_0^{\mathrm{h}} \\ \underline{\dot{\boldsymbol{u}}}_0^{\mathrm{h}} \\ \underline{\boldsymbol{u}}_0^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{x}}_0^{\mathrm{h}}} = \underbrace{\begin{bmatrix} \underline{\boldsymbol{f}}_0^{\mathrm{h}} \\ \underline{\dot{\boldsymbol{u}}}_0^{\mathrm{h}} \\ \underline{\boldsymbol{u}}_0^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{g}}_0^{\mathrm{h}}} \tag{3.8}$$

the general form reads:

$$\widetilde{\boldsymbol{H}}^s \underline{\boldsymbol{x}}_0^{\mathrm{h}} = \underline{\boldsymbol{g}}_0^{\mathrm{h}} \tag{3.9}$$

Coupling eq. (3.9) and eq. (3.7) and assemble the outputs in direction of time results in the full space-time representation of Newmark method as follows:

$$\underbrace{\begin{bmatrix} \widetilde{\boldsymbol{H}}^s & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \boldsymbol{H}^f & \boldsymbol{H}^s & \cdots & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{H}^f & \boldsymbol{H}^s \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \underline{\boldsymbol{x}}_0^{\mathrm{h}} \\ \underline{\boldsymbol{x}}_1^{\mathrm{h}} \\ \vdots \\ \vdots \\ \underline{\boldsymbol{x}}_{N_t}^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{X}}^{\mathrm{h}}} = \underbrace{\begin{bmatrix} \underline{\boldsymbol{g}}_0^{\mathrm{h}} \\ \underline{\boldsymbol{g}}_1^{\mathrm{h}} \\ \vdots \\ \vdots \\ \underline{\boldsymbol{g}}_{N_t}^{\mathrm{h}} \end{bmatrix}}_{\underline{\boldsymbol{G}}^{\mathrm{h}}} \tag{3.10}$$

therefore it can be seen that $\widetilde{\boldsymbol{H}}^s, \boldsymbol{H}^s$ denotes the diagonal entries of $\boldsymbol{A}$, and $\boldsymbol{H}^f$ are the off-diagonal entries of $\boldsymbol{A}$. Equation (3.10) can be written as:

$$\boldsymbol{A}\underline{\boldsymbol{X}}^{\mathrm{h}} = \underline{\boldsymbol{G}}^{\mathrm{h}} \tag{3.11}$$

now the full space-time representation of Newmark method is obtained. Compare to the conventional step-by-step Newmark integration method, this full space-time representation allows one to solve dynamic problems in one go. However, this is only necessary in the theoretical derivation due to the large size of $\boldsymbol{A}$ ($\boldsymbol{A} \in \mathbb{R}^{3\mathcal{N}N_t \times 3\mathcal{N}N_t}$). $\boldsymbol{A}$ is named as

the *dynamic operator*. $\boldsymbol{A}$ contains parametric system matrices $\boldsymbol{M}(\mu)$, $\boldsymbol{C}(\mu)$ and $\boldsymbol{K}(\mu)$, thus $\boldsymbol{A}$ is also affine parametrised. $\underline{\boldsymbol{X}}^{\mathrm{h}}$ is named as the *full response vector* and $\underline{\boldsymbol{G}}^{\mathrm{h}}$ is named as the *full force vector*.

Remark 1. *An elasto-dynamic problem can be solved using the form of* $\boldsymbol{A}\underline{\boldsymbol{X}}^{h} = \underline{\boldsymbol{G}}^{h}$*, as in elasto-statics. More interestingly, the dynamic operator* $\boldsymbol{A}$ *shares some similar properties with a stiffness matrix.* $\boldsymbol{A}$ *is sparse, positive-definite, but is non-symmetric.*

The following equivalence between the conventional Newmark method and ful space-time representation holds:

$$\begin{array}{c} \boldsymbol{A}^{-1}(\mu)\underline{\boldsymbol{G}}^{\mathrm{h}} \\ \downarrow \\ \underline{\boldsymbol{X}}^{\mathrm{h}}(\mu) \end{array} \quad := \quad \begin{array}{c} [\boldsymbol{F}^{\mathrm{h}}, \boldsymbol{M}(\mu), \boldsymbol{C}(\mu), \boldsymbol{K}(\mu)] \\ \downarrow \\ [\ddot{\boldsymbol{U}}^{\mathrm{h}}(\mu), \dot{\boldsymbol{U}}^{\mathrm{h}}(\mu), \boldsymbol{U}^{\mathrm{h}}(\mu)] \end{array}$$

3.2.1 A SDOF example

Consider a SDOF example from [7], in this section the results obtained from standard form and full space-time form of Newmark method will be presented and compared to give reader a comprehensive understanding of the full space-time Newmark representation. In this example, 100 time steps are considered with time step $\Delta t = 0.28s$, thus total time $T = 28s$. Force is only applied at the initial time step. Assume Rayleigh damping coefficients $a = 0.1$, $b = 0.1$, the mass, damping, stiffness matrices and force are given by:

$$\boldsymbol{M} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} 0.8 & -0.2 \\ -0.2 & 0.5 \end{bmatrix}, \quad \boldsymbol{K} = \begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix}, \quad \boldsymbol{F}_0 = \begin{bmatrix} 0 \\ 10 \end{bmatrix} \boldsymbol{F}_t = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

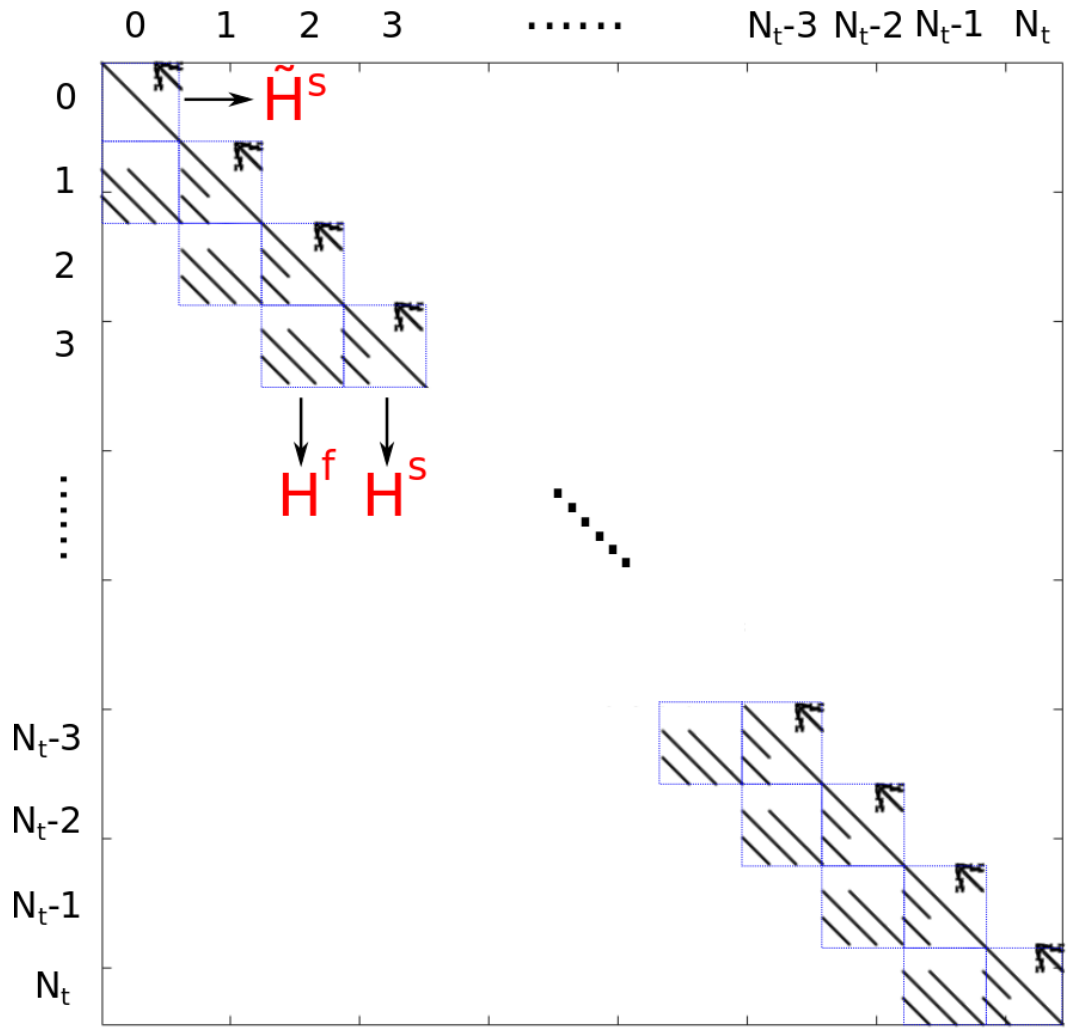


Figure 3.3: Sparsity of the *dynamic operator* $\boldsymbol{A}$.

the set of integration constants are:

$$a_0 = 51.0, \quad a_1 = 7.14, \quad a_2 = 14.3, \quad a_3 = 1.00,$$
$$a_4 = 1.00, \quad a_5 = 0.00, \quad a_6 = 0.14, \quad a_7 = 0.14$$

applying zero initial conditions, solving the given system with standard step-by-step Newmark method gives (showing results of first 5 time steps):

Time	0	Δt	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
$\ddot{u}_x$	0	0.263	0.387	0.271	-0.0266	-0.380
$\ddot{u}_y$	10	-1.28	-2.08	-2.18	-1.64	-0.748
$\dot{u}_x$	0	0.0368	0.128	0.220	0.254	0.197
$\dot{u}_y$	0	1.22	0.751	0.155	-0.380	-0.714
u_x	0	0.00515	0.0282	0.0768	0.143	0.206
u_y	0	0.171	0.447	0.574	0.542	0.389

Now the full space-time representation is being applied, first block components of the dynamic operator are obtained:

$$\widetilde{\boldsymbol{H}}^s = \begin{bmatrix} 2 & 0 & 0.8 & -0.2 & 6 & -2 \\ 0 & 1 & -0.2 & 0.5 & -2 & 4 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{H}^s = \begin{bmatrix} 2 & 0 & 0.8 & -0.2 & 6 & -2 \\ 0 & 1 & -0.2 & 0.5 & -2 & 4 \\ 0 & 0 & 1 & 0 & -7.14 & 0 \\ 0 & 0 & 0 & 1 & 0 & -7.14 \\ 1 & 0 & 0 & 0 & -51.0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -51.0 \end{bmatrix},$$

$$\boldsymbol{H}^f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 7.14 & 0 \\ 0 & 0 & 0 & 1 & 0 & 7.14 \\ 1 & 0 & 14.29 & 0 & 51.0 & 0 \\ 0 & 1 & 0 & 14.29 & 0 & 51.0 \end{bmatrix}$$

the dynamic operator is assembled by aligning $\widetilde{\boldsymbol{H}}^s$, $\boldsymbol{H}^s$ on the diagonal blocks, and $\boldsymbol{H}^f$ on the off-diagonal blocks. Generate $\underline{\boldsymbol{G}}^{\mathrm{h}}$ by vectorising $\boldsymbol{F}^{\mathrm{h}}(t)$, the *full response vector* can be obtained by solving $\boldsymbol{A}\underline{\boldsymbol{X}}^{\mathrm{h}} = \underline{\boldsymbol{G}}^{\mathrm{h}}$ (showing results of first 5 time steps, $\underline{\boldsymbol{X}}^{\mathrm{h}}$ has been reshaped to space-time form to enhance visibility):

Time	0	Δt	$2\Delta t$	$3\Delta t$	$4\Delta t$	$5\Delta t$
$\ddot{u}_x$	-3.820×10^{-16}	0.263	0.387	0.271	-0.0266	-0.380
$\ddot{u}_y$	10	-1.28	-2.08	-2.18	-1.64	-0.748
$\dot{u}_x$	2.877×10^{-16}	0.0368	0.128	0.220	0.254	0.197
$\dot{u}_y$	5.595×10^{-16}	1.22	0.751	0.155	-0.380	-0.714
u_x	3.502×10^{-16}	0.00515	0.0282	0.0768	0.143	0.206
u_y	7.277×10^{-16}	0.171	0.447	0.574	0.542	0.389

which equal to solution of step-by-step solutions, except initial condition due to machine precision. See fig. 3.4 for the output y-displacement of the SDOF example.

3.3 The reduced-order case

The full space-time representation of the Newmark method is also applicable to a parametric RB-model. Dimension of the problem is still large, only that the reduced order system is injected into the problem now. The space-time reduced basis approximation

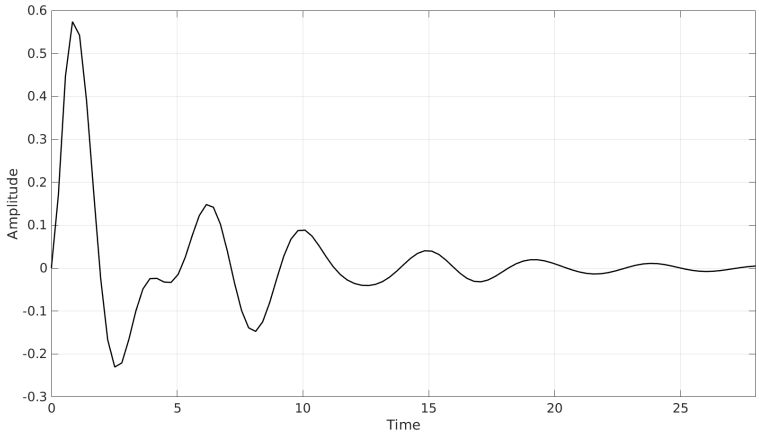


Figure 3.4: Y-displacement of the SDOF example.

of acceleration, velocity and displacement $\ddot{\boldsymbol{U}}^{\mathrm{r}}(\mu)$, $\dot{\boldsymbol{U}}^{\mathrm{r}}(\mu)$, $\boldsymbol{U}^{\mathrm{r}}(\mu)$ are defined as follows:

$$
\begin{cases}
\ddot{\boldsymbol{U}}^{\mathrm{r}}(\mu) = \boldsymbol{\Phi}\ddot{\boldsymbol{\alpha}}(\mu) = \left[\boldsymbol{\Phi}\underline{\ddot{\boldsymbol{\alpha}}}_0(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\ddot{\boldsymbol{\alpha}}}_n(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\ddot{\boldsymbol{\alpha}}}_{N_t}(\mu)\right] \\
\qquad\qquad = \left[\underline{\ddot{\boldsymbol{u}}}_0^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\ddot{\boldsymbol{u}}}_n^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\ddot{\boldsymbol{u}}}_{N_t}^{\mathrm{r}}(\mu)\right], 0 \le n \le N_t \\
\dot{\boldsymbol{U}}^{\mathrm{r}}(\mu) = \boldsymbol{\Phi}\dot{\boldsymbol{\alpha}}(\mu) = \left[\boldsymbol{\Phi}\underline{\dot{\boldsymbol{\alpha}}}_0(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\dot{\boldsymbol{\alpha}}}_n(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\dot{\boldsymbol{\alpha}}}_{N_t}(\mu)\right] \\
\qquad\qquad = \left[\underline{\dot{\boldsymbol{u}}}_0^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\dot{\boldsymbol{u}}}_n^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\dot{\boldsymbol{u}}}_{N_t}^{\mathrm{r}}(\mu)\right], 0 \le n \le N_t \\
\boldsymbol{U}^{\mathrm{r}}(\mu) = \boldsymbol{\Phi}\boldsymbol{\alpha}(\mu) = \left[\boldsymbol{\Phi}\underline{\boldsymbol{\alpha}}_0(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\boldsymbol{\alpha}}_n(\mu) \quad \dots \quad \boldsymbol{\Phi}\underline{\boldsymbol{\alpha}}_{N_t}(\mu)\right] \\
\qquad\qquad = \left[\underline{\boldsymbol{u}}_0^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\boldsymbol{u}}_n^{\mathrm{r}}(\mu) \quad \dots \quad \underline{\boldsymbol{u}}_{N_t}^{\mathrm{r}}(\mu)\right], 0 \le n \le N_t
\end{cases}
\tag{3.12}
$$

here $\underline{\ddot{\boldsymbol{\alpha}}}_n$, $\underline{\dot{\boldsymbol{\alpha}}}_n$, $\underline{\boldsymbol{\alpha}}_n \in \mathbb{R}^N$ are column reduced variable vectors at time step n. Since dynamic problems are the main concern of this thesis, the reduced variables are separated into acceleration (the second derivative), velocity (the first derivative) and displacement relative terms. Injecting the reduced order model into the full space-time representation

and replacing the exact solutions with RB-approximations result in the following:

$$\underbrace{\begin{bmatrix}\underline{\boldsymbol{f}}_n^{\mathrm{h}}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}}_{\underline{\boldsymbol{g}}_n^{\mathrm{h}}} \approx \underbrace{\begin{bmatrix}\boldsymbol{M} & \boldsymbol{C} & \boldsymbol{K}\\ \mathbf{0} & \mathbf{I} & -a_1\mathbf{I}\\ \mathbf{I} & \mathbf{0} & -a_0\mathbf{I}\end{bmatrix}}_{\boldsymbol{H}^s}\begin{bmatrix}\boldsymbol{\Phi}\underline{\ddot{\boldsymbol{\alpha}}}_n\\ \boldsymbol{\Phi}\underline{\dot{\boldsymbol{\alpha}}}_n\\ \boldsymbol{\Phi}\underline{\boldsymbol{\alpha}}_n\end{bmatrix} + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0}\\ a_5\mathbf{I} & a_4\mathbf{I} & a_1\mathbf{I}\\ a_3\mathbf{I} & a_2\mathbf{I} & a_0\mathbf{I}\end{bmatrix}}_{\boldsymbol{H}^f}\begin{bmatrix}\boldsymbol{\Phi}\underline{\ddot{\boldsymbol{\alpha}}}_{n-1}\\ \boldsymbol{\Phi}\underline{\dot{\boldsymbol{\alpha}}}_{n-1}\\ \boldsymbol{\Phi}\underline{\boldsymbol{\alpha}}_{n-1}\end{bmatrix} \tag{3.13}$$

separate reduced basis and reduced variable vectors as follows:

$$\underbrace{\begin{bmatrix}\underline{\boldsymbol{f}}_n^{\mathrm{h}}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}}_{\underline{\boldsymbol{g}}_n^{\mathrm{h}}} \approx \underbrace{\begin{bmatrix}\boldsymbol{M} & \boldsymbol{C} & \boldsymbol{K}\\ \mathbf{0} & \mathbf{I} & -a_1\mathbf{I}\\ \mathbf{I} & \mathbf{0} & -a_0\mathbf{I}\end{bmatrix}}_{\boldsymbol{H}^s}\underbrace{\begin{bmatrix}\boldsymbol{\Phi} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Phi} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}\end{bmatrix}}_{\boldsymbol{\psi}}\underbrace{\begin{bmatrix}\underline{\ddot{\boldsymbol{\alpha}}}_n\\ \underline{\dot{\boldsymbol{\alpha}}}_n\\ \underline{\boldsymbol{\alpha}}_n\end{bmatrix}}_{\underline{\boldsymbol{\pi}}_n} + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0}\\ a_5\mathbf{I} & a_4\mathbf{I} & a_1\mathbf{I}\\ a_3\mathbf{I} & a_2\mathbf{I} & a_0\mathbf{I}\end{bmatrix}}_{\boldsymbol{H}^f}\underbrace{\begin{bmatrix}\boldsymbol{\Phi} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Phi} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}\end{bmatrix}}_{\boldsymbol{\psi}}\underbrace{\begin{bmatrix}\underline{\ddot{\boldsymbol{\alpha}}}_{n-1}\\ \underline{\dot{\boldsymbol{\alpha}}}_{n-1}\\ \underline{\boldsymbol{\alpha}}_{n-1}\end{bmatrix}}_{\underline{\boldsymbol{\pi}}_{n-1}} \tag{3.14}$$

the reduced variable vectors at time step n are now assembled in a column, $\underline{\boldsymbol{\pi}}_n$ is used to denote this column vector. The above matrix form now becomes:

$$\underline{\boldsymbol{g}}_n^{\mathrm{h}} \approx \boldsymbol{H}^s\boldsymbol{\psi}\underline{\boldsymbol{\pi}}_n + \boldsymbol{H}^f\boldsymbol{\psi}\underline{\boldsymbol{\pi}}_{n-1} \tag{3.15}$$

moreover, the initial step also possesses an approximation that:

$$\underbrace{\begin{bmatrix}\underline{\boldsymbol{f}}_0^{\mathrm{h}}\\ \underline{\dot{\boldsymbol{u}}}_0^{\mathrm{h}}\\ \underline{\boldsymbol{u}}_0^{\mathrm{h}}\end{bmatrix}}_{\underline{\boldsymbol{g}}_0^{\mathrm{h}}} \approx \underbrace{\begin{bmatrix}\boldsymbol{M} & \boldsymbol{C} & \boldsymbol{K}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}}_{\widetilde{\boldsymbol{H}}^s}\underbrace{\begin{bmatrix}\boldsymbol{\Phi} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Phi} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}\end{bmatrix}}_{\boldsymbol{\psi}}\underbrace{\begin{bmatrix}\underline{\ddot{\boldsymbol{\alpha}}}_0\\ \underline{\dot{\boldsymbol{\alpha}}}_0\\ \underline{\boldsymbol{\alpha}}_0\end{bmatrix}}_{\underline{\boldsymbol{\pi}}_0} \tag{3.16}$$

i.e.

$$\underline{\boldsymbol{g}}_0^{\mathrm{h}} \approx \widetilde{\boldsymbol{H}}^s\boldsymbol{\psi}\underline{\boldsymbol{\pi}}_0 \tag{3.17}$$

let $\underline{\boldsymbol{g}}_n^{\mathrm{r}} = \boldsymbol{H}^s \boldsymbol{\psi} \underline{\boldsymbol{\pi}}_n + \boldsymbol{H}^f \boldsymbol{\psi} \underline{\boldsymbol{\pi}}_{n-1}$, $\underline{\boldsymbol{g}}_0^{\mathrm{r}} = \widetilde{\boldsymbol{H}}^s \boldsymbol{\psi} \underline{\boldsymbol{\pi}}_0$. Similar to eq. (3.10), couple eq. (3.15) and eq. (3.17) and write the results in terms of time incrementation:

$$\underbrace{\begin{bmatrix} \underline{\boldsymbol{g}}_0^{\mathrm{r}} \\ \underline{\boldsymbol{g}}_1^{\mathrm{r}} \\ \vdots \\ \vdots \\ \underline{\boldsymbol{g}}_{N_t}^{\mathrm{r}} \end{bmatrix}}_{\underline{\boldsymbol{G}}^{\mathrm{r}}} = \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{H}}^s & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \boldsymbol{H}^f & \boldsymbol{H}^s & \cdots & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{H}^f & \boldsymbol{H}^s \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{\psi} & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\psi} & \cdots & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \cdots & \boldsymbol{\psi} \end{bmatrix}}_{\boldsymbol{\Psi}} \underbrace{\begin{bmatrix} \underline{\boldsymbol{\pi}}_0 \\ \underline{\boldsymbol{\pi}}_1 \\ \vdots \\ \vdots \\ \underline{\boldsymbol{\pi}}_{N_t} \end{bmatrix}}_{\underline{\boldsymbol{\Pi}}} \tag{3.18}$$

the mathematical form reads

$$\underline{\boldsymbol{G}}^{\mathrm{r}} = \boldsymbol{A} \boldsymbol{\Psi} \underline{\boldsymbol{\Pi}} \xrightarrow{\boldsymbol{\Psi}\underline{\boldsymbol{\Pi}} = \underline{\boldsymbol{X}}^{\mathrm{r}}} \underline{\boldsymbol{G}}^{\mathrm{r}} = \boldsymbol{A} \underline{\boldsymbol{X}}^{\mathrm{r}} \tag{3.19}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{3N(N_t+1) \times 3N(N_t+1)}$ denotes the *assembled reduced basis matrix*, $\underline{\boldsymbol{\Pi}} \in \mathbb{R}^{3N(N_t+1)}$ is the *full reduced variable vector* aligned in time and $\underline{\boldsymbol{G}}^{\mathrm{r}}$ is the *approximated full force vector*. Equip each term with the parametric form, a similar equivalence as aforementioned in section 3.2 can be obtained:

$$\begin{array}{c} \boldsymbol{A}^{-1}(\mu) \underline{\boldsymbol{G}}^{\mathrm{r}}(\mu) \\ \downarrow \\ \underline{\boldsymbol{X}}^{\mathrm{r}}(\mu) \end{array} \quad := \quad \begin{array}{c} [\boldsymbol{F}^{\mathrm{r}}(\mu), \boldsymbol{M}(\mu), \boldsymbol{C}(\mu), \boldsymbol{K}(\mu)] \\ \downarrow \\ [\ddot{\boldsymbol{U}}^{\mathrm{r}}(\mu), \dot{\boldsymbol{U}}^{\mathrm{r}}(\mu), \boldsymbol{U}^{\mathrm{r}}(\mu)] \end{array}$$

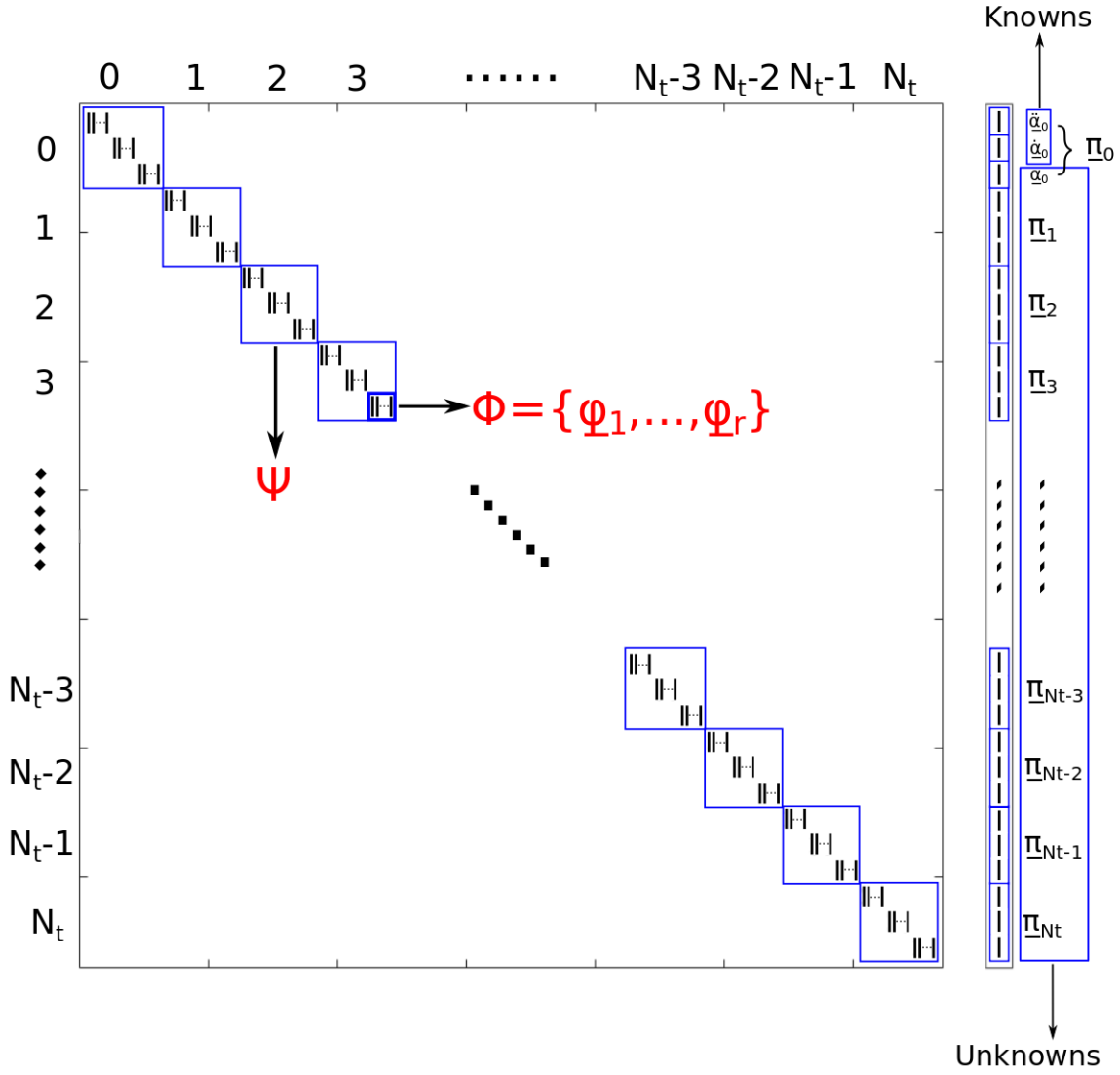


Figure 3.5: The *assembled reduced basis matrix* $\boldsymbol{\Psi}$ and *full reduced variable vector* $\underline{\boldsymbol{\Pi}}$. Components of $\boldsymbol{\Psi}$ aligns on the diagonal entries.

Chapter 4

Development of A New Error Indicator for *POD-Greedy* Sampling Algorithm

In this chapter, the major contribution is introduced: a new error indicator for POD-Greedy algorithm. This chapter begins with an introduction of the rationale behind this new error indicator, i.e. discussion of existing approaches to improve performances of POD-Greedy algorithm. Then the exact (section 4.2) and approximated (section 4.3) full error vector are derived based on the full space-time representation of the Newmark method. The derivation is then extended to approximated displacement error (section 4.4) by applying transformations which utilise step-by-step Newmark method and Duhamel's integral. A preliminary comparison of proposed error indicator and residual indicator is presented in section 4.6 to show power of such approach. Finally SVD and POD are applied to the proposed indicator to achieve additional speed-up (section 4.7, section 4.8).

4.1 Introduction

Snapshot-based methods require a carefully chosen set of samples to form a representative reduced space thereby to generate a good basis. This can be achieved by applying the well-established methods: the Greedy procedure [44, 91, 106] for static problems, or the extension: POD-Greedy algorithm [47, 62] for dynamic problems. The snapshots are evaluated exhaustively on a training set, which is required to be as large as possible, such that no key information is dropped out. A reduced basis is then constructed by compressing the obtained information. The size of the training set is crucial to the speed of *standard POD-Greedy* algorithm due to the fact that many full problems need to be solved when an *a-posteriori* error estimator is not available. More specifically, a large training set may result in prohibitively high computational cost, especially when dealing with dynamic problems. These challenges lead to a class of problems namely training set treatment. A lot of research has been conducted to generate good training set, as well as reducing the workload of the *standard POD-Greedy* algorithm, for example, adaptive training set extension and parameter domain partitions [33, 34, 46], multistage greedy algorithm [101] and randomisation [50].

The size of training set might be a function of numerous variables, such as material parameters, physical structure components, parametric domain discretisations, boundary conditions, etc. A possible solution to restrain the training set size is the multistage Greedy algorithm introduced in [101], which uses a coarse, small training set to replace the fine, large one, and the Greedy algorithm is operated using this small training set to generate the reduced basis. The large training set is then searched to confirm that the generated basis is sufficient. For the parametric sample points where the reduced basis shows insufficient performance, the Greedy algorithm is rerun only over these points and the new basis vectors would be generated using information produced from the new Greedy iterations. A necessary ingredient of multistage Greedy algorithm is the rigorous and sharp *a-posteriori* error bounds to guarantee reliability of of the results.

A drawback of such approaches is that overfitting tends to happen when the training set is not sufficiently large, i.e. even if the overall error converges nicely, the error for individual parametric values may still be large. Hence a 'smarter' algorithm is in need to determine the appropriate training set size and locations, such that overfitting and high computational cost can be both prevented.

Adaptive training set extension (see [46]) aims to prevent overfitting by setting up an extra validation set m_{val} (small and randomly chosen), and an extra tolerance γ_{tol}. First an Early Stopping Greedy algorithm is applied until either Greedy error tolerance tol^{err} or the extra tolerance γ_{tol} is reached. Then if the Greedy error tolerance is exceeded, the training set will be locally refined in the region where the error indicator reaches maximum value. The initial training set is chosen to be very small, and the refinement only happens where needed. The cost of the Greedy training progress is reduced by precisely determining the number and location of the training set points. Another solution is adaptive parameter domain partitioning. The goal is to resolve the problem of large basis size such that the ROM "*online*" complexity can be reduced. POD-Greedy algorithm may result in a very large basis in order to achieve the desired Greedy error tolerance tol^{err}, as a result the "*online*" cost becomes prohibitively high as it requires operations with full matrices. Ultimately the cost of a single online operation may be higher than the cost of solving the original exact problem if the basis size is too large. Adaptive parameter domain partitioning limits the size of the reduced basis by setting up a maximum basis size, $n_{\max}$. This is achieved by partitioning the parameter domain and generating small reduced bases on each subdomain. Again, the Early Stopping Greedy algorithm is induced if $n_{\max}$ is exceed, which indicates that the solution is too complex to be recast by a small basis. Then the parameter domain is refined using similar approaches to [33, 34] until prescribed tolerance and basis size limit is reached. Adaptive training set extension and parameter domain partitioning can be suitably combined to yield reduced ROM "*online*" computational cost. The time-interval may also be adaptively partitioned if it's too large, see [28, 29].

A model-constrained adaptive sampling strategy based on a standard Greedy sampling algorithm for large-scale systems with many parameters is proposed in [12]. The goal of the methodology is to determine the appropriate magic point locations. It has been proven that the strategy would not sample the same point in the parametric space. The adaptive strategy utilises gradient-based optimisation, and assumes that no *a-posteriori* error estimators are available. Instead, the squared norm of the residual is used as an error indicator instead of true error, to limit the computational cost. Using such indicator can be risky, as radically it is different from using true error as the indicator, and the true error is expensive but accurate. Numerical results in [12] show that using squared norm of the residual as indicator is approximately an order of magnitude less expensive than using true error as the indicator, thus the numerical cost is effectively reduced. Also convergence of the new strategy is faster compared to statically based methods. Though a major drawback is that the error convergence of using the squared residual indicator is slower than using the true error indicator, unless a large basis is generated. The convergence of residual and true RB-error indicators are too diversified, thus application of the residual indicator is not sufficiently convincing. This might more be intuitive but a better performance can be achieved. Another drawback of a residual indicator is that an insufficient number of cases are tested to ensure generality of the strategy, i.e. evidence provided to prove that using the residual error indicator would achieve a good performance is not conclusive enough.

Dynamic or hyperbolic problems are complex intrinsically, for example, hyperbolic problems that contain sharp gradients or discontinuities with respect to the parameter are difficult to be approximated accurately, as conventional reduced order models tend to smooth out the key features or suffer oscillations at or near these sharp regions. One solution is to use the local parametric domain refinement applied in [47, 55, 114], where the discontinuous regions are highly-discretised, so that local features can be well-represented. However, this might not be suitable for many parameter cases due to possible high computational cost. Snapshot-POD may be used to construct a reduced basis, such that

solutions at other points can be approximated with it. The rationale behind this is that the solution varies smoothly in most physical domain and the regions that contain discontinuity can be relatively small. In [22], the gradient of the approximations is used to describe the discontinuities, and the exact problem is solved at these discontinuities to improve accuracy of the reduced order model. Boundary and initial conditions might be arbitrary on the subregions, hence appropriate solvers are needed to treat such special conditions.

The above approaches all aim to improve the performance of the (POD-)Greedy algorithm, however, the following problem remains unsolved: given a dynamic model with a large, preset, uniformly structured training set, how to develop an improved POD-Greedy algorithm which evaluates the error over the large training set efficiently? With many parameter settings, it is almost impossible to evaluate all of the parameter sample points in the training set, as the number of points increases exponentially due to the curse of dimensionality. For example, if there are 10 parameters and each possesses only 5 sample points, the resulting training set contains 9765625 points. This is computationally prohibitive for standard (POD-) Greedy algorithm. To the author's knowledge, the existing methods are insufficient to deal with such problems due to lack of sharp, rigorous error bound for dynamic problems. In the cases where a posteriori error estimator is unavailable, using true error as indicator guarantees accuracy but may suffer prohibitively high numerical cost. Therefore this chapter presents a new error indicator which tackles the problem and allows the user to evaluate a very large training set. Standard POD-Greedy algorithm requires a full sweep on a parameter training set, which can possibly result in a large number of exact solutions. Moreover the size of the training set needs to be carefully chosen: a large size leads to precise sweep but possible slow evaluations; a small size leads to quick evaluation but important information might be left out. The goal of the new error indicator is to allow users to evaluate a very large training set without losing the calculating efficiency. Our solution is to interpolate the *dynamic operator* inverse, the rationale behind this choice will be explained, as well as the procedures to

achieve it.

This chapter is organised as follows: first the true RB-error vector (the exact error vector) is derived using the full Newmark representation developed in chapter 3. Then the true RB-error vector for dynamic problems is approximated by interpolating the inverse of the *dynamic operator*. The approximated displacement error is then derived. More specifically, a sequence of unit impulse responses by applying the standard Newmark step-by-step method. An error norm square is then computed to be the interpolated term. Once the above procedures are introduced, the method is demonstrated by conducting numerical experiments based on the simple beam model and showing that it accurately predicts error convergence. The proposed error indicator is then compared with the residual indicator to show its feasibility. In section 4.7 and section 4.8, speed of the proposed indicator is further improved without losing accuracy, this is achieved by compressing the impulse responses and perform POD on collection of reduced variable vectors.

4.2 Exact full error vector

Now the full force vectors $\underline{\boldsymbol{G}}^{\mathrm{h}}$ has been derived in eq. (3.11) and $\underline{\boldsymbol{G}}^{\mathrm{r}}$ (eq. (3.19)), the parametric *full residual vector* $\underline{\boldsymbol{R}}(\mu)$ reads:

$$\underline{\boldsymbol{R}}(\mu) = \underline{\boldsymbol{G}}^{\mathrm{h}} - \underline{\boldsymbol{G}}^{\mathrm{r}}(\mu) = \boldsymbol{A}(\mu)\underline{\boldsymbol{E}}(\mu) \tag{4.1}$$

the parametric *exact full error vector* $\underline{\boldsymbol{E}}(\mu)$ reads (assuming $\underline{\boldsymbol{G}}^{\mathrm{h}}$ is deterministic):

$$\begin{aligned} \underline{\boldsymbol{E}}(\mu) &= \boldsymbol{A}^{-1}(\mu)\underline{\boldsymbol{R}}(\mu) \\ &= \boldsymbol{A}^{-1}(\mu)(\underline{\boldsymbol{G}}^{\mathrm{h}} - \underline{\boldsymbol{G}}^{\mathrm{r}}(\mu)) \\ &= \underline{\boldsymbol{X}}^{\mathrm{h}}(\mu) - \underline{\boldsymbol{X}}^{\mathrm{r}}(\mu) \end{aligned} \tag{4.2}$$

here the system is restricted to admit the affine form (see section 2.2.1). The full space-time representation of Newmark method can be conveniently integrated into the *reference POD-Greedy* algorithm. The *error response surface* can be constructed by evaluating the functional relationship between value μ and the norm of $\underline{\boldsymbol{E}}(\mu)$, $\forall \mu \in \mathcal{P}$. Notice that the *exact full error vector* evaluates error in the entire space and time domain, i.e. it contains the acceleration, velocity and displacement true RB-error. In order to quickly identify parametric problems in different scenarios, the following distinctions are made:

- **Full representation (FR)** – solve the dynamic problem using $\boldsymbol{A}(\mu)\underline{\boldsymbol{X}}^{\mathrm{h}}(\mu) = \underline{\boldsymbol{G}}^{\mathrm{h}}(\mu)$, output vector $\underline{\boldsymbol{X}}^{\mathrm{h}}(\mu) \in \mathbb{R}^{3\mathcal{N}N_t \times 1}$.
- **Standard representation (SR)** – solve the dynamic problem using step-by-step Newmark method, output $\boldsymbol{U}^{\mathrm{h}}(\mu) \in \mathbb{R}^{\mathcal{N} \times N_t}$.
- **Reduced order representation (RR)** – solve the dynamic problem under an RB frame using step-by-step Newmark method, output $\boldsymbol{\alpha}(\mu) \in \mathbb{R}^{N \times N_t}$.

these distinctions will be applied to clarify scales of system operators, space-time responses and forces. They will be labelled after the associated terms, so that quick guidance can be provided for readers to identify the different representations of the dynamic problems.

4.3 Approximated full error vector

The *reference POD-Greedy* algorithm requires a carefully evaluated setting of $\mathcal{P}^{\mathrm{train}}$ size, otherwise the computational cost might be prohibitively high. More specifically, the size of $\mathcal{P}^{\mathrm{train}}$ needs to be small enough to ensure a rapid calculation, and large enough to ensure $\mathcal{P}^{\mathrm{train}}$ is representative. This can be solved by training set treatment, see

section 2.2.4 for existing methods. Alternatively, a new error indicator is proposed, where approximations of $\underline{\boldsymbol{E}}(\mu)$ are computed by interpolating the parametric *dynamic operator* inverse: $\boldsymbol{A}^{-1}(\mu)$, so that the evaluation process over $\mathcal{P}^{\text{train}}$ can be greatly accelerated. As a result, users can select a much larger $\mathcal{P}^{\text{train}}$ meaning detailed information of the model can be obtained. The inverse of the dynamic operator $\boldsymbol{A}^{-1}(\mu)$ are chosen to be interpolated for the following reasons:

- To maintain the robust error estimate $\underline{\boldsymbol{E}}(\mu) = \boldsymbol{A}^{-1}(\mu)\underline{\boldsymbol{R}}(\mu)$, so that the error is evaluated in the entire time domain.

- $\boldsymbol{A}^{-1}(\mu)$ is the only suitable term to be approximated: expanding the full error vector as follows:

$$\underline{\boldsymbol{E}}(\mu) = \boldsymbol{A}^{-1}(\mu)\underline{\boldsymbol{R}}(\mu) = \boldsymbol{A}^{-1}(\mu)\Big(\underline{\boldsymbol{G}}^{\text{h}} - \underline{\boldsymbol{G}}^{\text{r}}(\mu)\Big) = \boldsymbol{A}^{-1}(\mu)\Big(\underline{\boldsymbol{G}}^{\text{h}} - \boldsymbol{A}(\mu)\boldsymbol{\Psi}\underline{\boldsymbol{\Pi}}(\mu)\Big)$$

  It can be seen that $\underline{\boldsymbol{G}}^{\text{h}}$ is a known term, $\boldsymbol{A}(\mu)$ is affine parametric, $\boldsymbol{\Psi}$ can be computed *a-posteriori* and $\underline{\boldsymbol{\Pi}}(\mu)$ is inexpensive to be directly computed. As a result, $\underline{\boldsymbol{R}}(\mu)$ should not be approximated. The only term left to be approximated is $\boldsymbol{A}^{-1}(\mu)$.

- $\boldsymbol{A}^{-1}(\mu)$ is non-affine, thus affine expansion can not be utilised to obtain a separated form.

denote Lagrange sample point with $\{\mu_i\}_{i=1}^{N_i}$, $\mu_i \in \mathcal{P}^i \subset \mathcal{P}$. $\mathcal{P}^i$ is the interpolation sample domain. Define the collection of exact solutions at the Lagrange samples by $X_i = \left\{\boldsymbol{U}^{\text{h}}(\mu_i)\right\}_{i=1}^{N_i}$. The approximated *dynamic operator* inverse at a parameter point is given by:

$$\widehat{\boldsymbol{A}^{-1}}(\mu) = \sum_{i=1}^{N_i} l_i(\mu)\boldsymbol{A}_i^{-1} \tag{4.3}$$

where $l_i(\mu)$ denotes piecewise linear Lagrange polynomial, $\boldsymbol{A}_i^{-1}$ denotes pre-computed *dynamic operator* inverse at interpolation sample points. Therefore the approximated *full error vector* reads (assume that $\underline{\boldsymbol{G}}^{\mathrm{h}}$ is deterministic):

$$\begin{aligned}\widehat{\underline{\boldsymbol{E}}}(\mu) &= \widehat{\boldsymbol{A}^{-1}}(\mu)\underline{\boldsymbol{R}}(\mu) \\ &= \sum_{i=1}^{N_i} l_i(\mu)\boldsymbol{A}_i^{-1}\underline{\boldsymbol{R}}(\mu) \\ &= \sum_{i=1}^{N_i} l_i(\mu)\boldsymbol{A}_i^{-1}\left(\underline{\boldsymbol{G}}^{\mathrm{h}} - \underline{\boldsymbol{G}}^{\mathrm{r}}(\mu)\right)\end{aligned} \tag{4.4}$$

apply affine dependence, substitute eq. (3.19), the reduced full force vector reads:

$$\begin{aligned}\underline{\boldsymbol{G}}^{\mathrm{r}}(\mu) &= \boldsymbol{A}(\mu)\underline{\boldsymbol{X}}^{\mathrm{r}}(\mu) \\ &= \sum_{j=1}^{N_j} \gamma_j(\mu)\boldsymbol{A}_j\boldsymbol{\Psi}\underline{\boldsymbol{\Pi}}_j(\mu) \\ &= \sum_{j=1}^{N_j} \gamma_j(\mu)\boldsymbol{A}_j \sum_{\mathrm{r}=1}^{N} \boldsymbol{\Psi}_{\mathrm{r}}\underline{\boldsymbol{\Pi}}_{j\mathrm{r}}(\mu)\end{aligned} \tag{4.5}$$

notice that the *full reduced variable vector* is transformed to an affine-dependent form. Substituting eq. (4.5) back to eq. (4.4) results in:

$$\widehat{\underline{\boldsymbol{E}}}(\mu) = \sum_{i=1}^{N_i} l_i(\mu)\left(\boldsymbol{A}_i^{-1}\underline{\boldsymbol{G}}^{\mathrm{h}} - \sum_{j=1}^{N_j}\sum_{\mathrm{r}=1}^{N} \gamma_j(\mu)\boldsymbol{A}_i^{-1}\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathrm{r}}\underline{\boldsymbol{\Pi}}_{j\mathrm{r}}(\mu)\right) \tag{4.6}$$

due to the number of parameters involved, the interpolation might be multi-variate depending on the number of parameters. The interpolation problem needs to be solved sequentially for each parameter. See [89, 100] for multivariate interpolation. In order to solve eq. (4.6) and obtain the approximated *exact full error vector*, let:

$$\begin{aligned}\mathsf{Term}_i^{\mathrm{h}} &= \boldsymbol{A}_i^{-1}\underline{\boldsymbol{G}}^{\mathrm{h}}, \\ \mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}} &= \boldsymbol{A}_i^{-1}\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathrm{r}}\underline{\boldsymbol{\Pi}}_{j\mathrm{r}}(\mu)\end{aligned} \tag{4.7}$$

eq. (4.6) now becomes:

$$\widehat{\underline{\boldsymbol{E}}}(\mu) = \sum_{i=1}^{N_i} l_i(\mu)\Big(\mathsf{Term}_i^{\mathrm{h}} - \sum_{j=1}^{N_j}\sum_{\mathrm{r}=1}^{N} \gamma_j(\mu)\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}\Big) \tag{4.8}$$

the unknowns are $\mathsf{Term}_i^{\mathrm{h}}$ and $\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}$. $\mathsf{Term}_i^{\mathrm{h}}$ is simply given by:

$$\mathsf{Term}_i^{\mathrm{h}} = \boldsymbol{A}_i^{-1}\underline{\boldsymbol{G}}^{\mathrm{h}} = \underline{\boldsymbol{X}}_i^{\mathrm{h}} \tag{4.9}$$

where $\underline{\boldsymbol{X}}_i^{\mathrm{h}}$ denotes the *full response vector* obtained by applying the *full force vector* $\underline{\boldsymbol{G}}^{\mathrm{h}}$ on the structure. $\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}$ is not as straight forward, as more terms are involved. Notice that the simplest affine term in $\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}$ is $\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathrm{r}}$, moreover, the multiplication between the inverse of *dynamic operator* $\boldsymbol{A}_i^{-1}$ and other terms is equivalent to applying a force on the dynamic system $\boldsymbol{A}_i$. The force vector reads:

$$\underline{\boldsymbol{G}}_{j\mathrm{r}}(\mu) = \boldsymbol{A}_j\boldsymbol{\Psi}_{\mathrm{r}}\underline{\boldsymbol{\Pi}}_{j\mathrm{r}}(\mu) \tag{4.10}$$

apply the force vector on dynamic system $\boldsymbol{A}_i$, $\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}$ is given by:

$$\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}} = \boldsymbol{A}_i^{-1}\underline{\boldsymbol{G}}_{j\mathrm{r}}(\mu) = \underline{\boldsymbol{X}}_{ij\mathrm{r}}^{\mathrm{r}}(\mu) \tag{4.11}$$

here $\underline{\boldsymbol{X}}_{ij\mathrm{r}}^{\mathrm{r}}(\mu)$ denotes the *full response vector* $\boldsymbol{X}$ associated with $\mathsf{Term}_{ij\mathrm{r}}^{\mathrm{r}}$ for the i^{th} interpolation point, j^{th} affined term, and r^{th} reduced basis vector. Therefore eq. (4.8) becomes (in the full space-time representation):

$$\widehat{\underline{\boldsymbol{E}}}(\mu) = \sum_{i=1}^{N_i} l_i(\mu)\Big(\underline{\boldsymbol{X}}_i^{\mathrm{h}} - \sum_{j=1}^{N_j}\sum_{\mathrm{r}=1}^{N} \gamma_j(\mu)\underline{\boldsymbol{X}}_{ij\mathrm{r}}^{\mathrm{r}}(\mu)\Big) \quad \textbf{[FR]} \tag{4.12}$$

again we emphasise that $\boldsymbol{A}^{-1}$ is not needed to be built in real applications, but it is necessary for theoretical derivation of this thesis.

4.4 Approximated displacement error

In a parametric dynamic problem with reduced order modelling, often the displacement error is the quantity of interest, which is measured by the numerical distance between the finite element solution $\boldsymbol{U}^{\mathrm{h}}(\mu)$ and the displacement $\boldsymbol{U}^{\mathrm{r}}(\mu)$ obtained from an RB-model, i.e. Frobenius norm of the true RB-error $\|\boldsymbol{e}(\mu)\|_{\mathrm{F}} := \left\|\boldsymbol{U}^{\mathrm{h}}(\mu) - \boldsymbol{U}^{\mathrm{r}}(\mu)\right\|_{\mathrm{F}}$. In order to minimize this quantity, one possible approach is to solve eq. (4.12) and extract the displacement error from the *exact full error vector* $\underline{\boldsymbol{E}}$. Although this is mathematically correct and essential for theoretical derivation of approximated error, in reality this is computationally prohibited due to the large size of $\boldsymbol{A}$ ($\boldsymbol{A} \in \mathbb{R}^{3\mathcal{N}N_t \times 3\mathcal{N}N_t}$). Therefore a different approach has to be utilised, which is the step-by-step Newmark method for an equivalent and practical computation.

4.4.1 Practical approach using Newmark method

Step-by-step Newmark method can be applied to solve the approximated displacement error. In section 4.3 the solution has been recast into 2 parts: $\mathsf{Term}_i^{\mathrm{h}}$ and $\mathsf{Term}_{ijr}^{\mathrm{r}}$. The Newmark method is integrated to solve for each term individually.

(i) Recall that $\mathsf{Term}_i^{\mathrm{h}} = \boldsymbol{A}_i^{-1}\underline{\boldsymbol{G}}^{\mathrm{h}}$, $\underline{\boldsymbol{G}}^{\mathrm{h}}$ is vectorised from external force $\boldsymbol{F}^{\mathrm{h}}$. $\boldsymbol{A}_i$ is the *dynamic operator* equipped with system matrices $\boldsymbol{M}_i$, $\boldsymbol{C}_i$, $\boldsymbol{K}_i$. Solving $\mathsf{Term}_i^{\mathrm{h}}$ is equivalent to using Newmark method to solve:

$$\boldsymbol{M}_i\ddot{\boldsymbol{U}}_i^{\mathrm{h}} + \boldsymbol{C}_i\dot{\boldsymbol{U}}_i^{\mathrm{h}} + \boldsymbol{K}_i\boldsymbol{U}_i^{\mathrm{h}} = \boldsymbol{F}^{\mathrm{h}} \tag{4.13}$$

which gives the following transformation from the *full response vector* to displacement matrix:

$$\begin{aligned} &(\text{Solving } \boldsymbol{A}_i \underline{\boldsymbol{X}}_i^{\mathrm{h}} = \underline{\boldsymbol{G}}^{\mathrm{h}}) \Longrightarrow \underline{\boldsymbol{X}}_i^{\mathrm{h}} \quad \textbf{[FR]} \\ &\qquad \text{is transformed to} \\ &(\text{Newmark solution}) \Longrightarrow \boldsymbol{U}_i^{\mathrm{h}} \quad \textbf{[SR]} \end{aligned} \tag{4.14}$$

(ii) Recall that $\mathsf{Term}_{ij\mathfrak{r}}^{\mathrm{r}} = \boldsymbol{A}_i^{-1}\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathfrak{r}}\underline{\boldsymbol{\Pi}}_{j\mathfrak{r}}(\mu)$, first look at eq. (4.10), which defines the force vector in high-dimension. Notice that $\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathfrak{r}}$ results in a parameter-independent matrix which contains repeated column vectors. $\underline{\boldsymbol{\Pi}}_{j\mathfrak{r}}(\mu)$ is the *full reduced variable vector* for j^{th} affined term and $\mathfrak{r}^{\mathrm{th}}$ reduced basis vector. The parameter-independent *full impulse matrix* reads (see fig. 4.1 for a schematic illustration):

$$\boldsymbol{G}_{j\mathfrak{r}}^{imp} = \boldsymbol{A}_j\boldsymbol{\Psi}_{\mathfrak{r}} \tag{4.15}$$

in order to utilise Newmark method, the vectors in $\boldsymbol{G}_{j\mathfrak{r}}^{imp}$ are used as the non-zero components of unit impulses, the other components are just zero vectors. $\boldsymbol{F}_{j\mathfrak{r}}^{imp,m}$, $\boldsymbol{F}_{j\mathfrak{r}}^{imp,c}$ and $\boldsymbol{F}_{j\mathfrak{r}}^{imp,k}$ are used to denote mass, damping and stiffness associated impulses, respectively. Each sparse unit impulse has 2 components: unit impulse vectors and zero vectors. The mathematical form reads:

$$\begin{aligned} \boldsymbol{F}_{j\mathfrak{r}}^{imp,m} &= \begin{cases} \boldsymbol{M}_j\underline{\boldsymbol{\phi}}_{\mathfrak{r}}, & t = 0 \\ 0, & t > 0 \end{cases} \\ \boldsymbol{F}_{j\mathfrak{r}}^{imp,c} &= \begin{cases} \boldsymbol{C}_j\underline{\boldsymbol{\phi}}_{\mathfrak{r}}, & t = 0 \\ 0, & t > 0 \end{cases} \\ \boldsymbol{F}_{j\mathfrak{r}}^{imp,k} &= \begin{cases} \boldsymbol{K}_j\underline{\boldsymbol{\phi}}_{\mathfrak{r}}, & t = 0 \\ 0, & t > 0 \end{cases} \end{aligned} \tag{4.16}$$

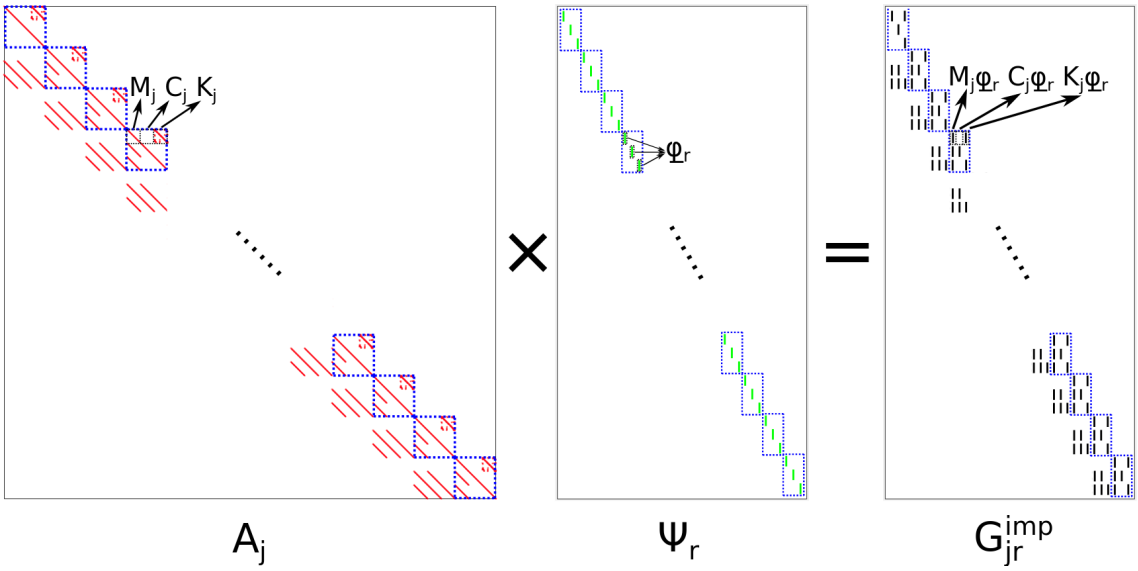


Figure 4.1: Derivation of the *full impulse matrix* $\boldsymbol{G}_{j\mathfrak{r}}^{imp}$, key components are column impulse vectors $\boldsymbol{M}_j\underline{\boldsymbol{\phi}}_{\mathrm{r}}$, $\boldsymbol{C}_j\underline{\boldsymbol{\phi}}_{\mathrm{r}}$ and $\boldsymbol{K}_j\underline{\boldsymbol{\phi}}_{\mathrm{r}}$.

from eq. (4.15) to eq. (4.16), the force term is transformed from full representation to standard representation, which indicates that now, the step-by-step Newmark method could be utilised. In $\mathsf{Term}_{ij\mathfrak{r}}^{\mathrm{r}}$, $\boldsymbol{A}_j\boldsymbol{\Psi}_{\mathrm{r}}$ is multiplied with $\boldsymbol{A}_i^{-1}$, which is equivalent to applying the unit impulses to the dynamic system corresponding to the interpolation sample point μ_i. Now step-by-step Newmark method is used to solve:

$$\begin{aligned}
\boldsymbol{M}_i\ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,m} + \boldsymbol{C}_i\dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,m} + \boldsymbol{K}_i\boldsymbol{U}_{ij\mathfrak{r}}^{imp,m} &= \boldsymbol{F}_{j\mathfrak{r}}^{imp,m} \\
\boldsymbol{M}_i\ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,c} + \boldsymbol{C}_i\dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,c} + \boldsymbol{K}_i\boldsymbol{U}_{ij\mathfrak{r}}^{imp,c} &= \boldsymbol{F}_{j\mathfrak{r}}^{imp,c} \\
\boldsymbol{M}_i\ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,k} + \boldsymbol{C}_i\dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,k} + \boldsymbol{K}_i\boldsymbol{U}_{ij\mathfrak{r}}^{imp,k} &= \boldsymbol{F}_{j\mathfrak{r}}^{imp,k}
\end{aligned} \tag{4.17}$$

and the following relationship holds:

$$\begin{aligned}
\boldsymbol{F}_{j\mathfrak{r}}^{imp} &= \boldsymbol{F}_{j\mathfrak{r}}^{imp,m} + \boldsymbol{F}_{j\mathfrak{r}}^{imp,c} + \boldsymbol{F}_{j\mathfrak{r}}^{imp,k} \\
\ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp} &= \ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,m} + \ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,c} + \ddot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,k} \\
\dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp} &= \dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,m} + \dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,c} + \dot{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp,k} \\
\boldsymbol{U}_{ij\mathfrak{r}}^{imp} &= \boldsymbol{U}_{ij\mathfrak{r}}^{imp,m} + \boldsymbol{U}_{ij\mathfrak{r}}^{imp,c} + \boldsymbol{U}_{ij\mathfrak{r}}^{imp,k}
\end{aligned} \tag{4.18}$$

solving eq. (4.17) results in the unit impulse responses, which include accelerations, velocities and displacements. Once eq. (4.17) is solved by the Newmark method, one may extract and deal only with displacements $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$ (all 3 equations need to be solved even if only displacement components are needed). $\boldsymbol{U}_i^{\mathrm{h}}$ and $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$ are the pre-computed impulse responses.

Unlike solving $\mathsf{Term}_i^{\mathrm{h}}$, the matrix formulation of the full space-time representation indicates a coupled relationship between impulse responses $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$ and reduced variables $\boldsymbol{\alpha}$. In other words, solving $\mathsf{Term}_{ij\mathfrak{r}}^{\mathrm{r}}$ requires a discrete Duhamel's integral between these 2 terms, which will be address in the following section. See fig. 4.2 for an example of unit impulse response (model adapted from section 2.2.4).

4.4.2 Discrete Duhamel's integral of unit impulse responses and reduced variables

The discrete Duhamel's integral of the impulse responses and reduced variables is defined as the sum of the product of two functions – shifted responses and the reduced variables:

$$\boldsymbol{U}_{ij\mathfrak{r}}(\mu) = \sum_{n=0}^{N_t} \left(\boldsymbol{U}_{ij\mathfrak{r}}^{imp,m} \delta_{tn} \ddot{\boldsymbol{\alpha}}_{\mathfrak{r}}(t_n;\mu) + \boldsymbol{U}_{ij\mathfrak{r}}^{imp,c} \delta_{tn} \dot{\boldsymbol{\alpha}}_{\mathfrak{r}}(t_n;\mu) + \boldsymbol{U}_{ij\mathfrak{r}}^{imp,k} \delta_{tn} \boldsymbol{\alpha}_{\mathfrak{r}}(t_n;\mu) \right) \tag{4.19}$$

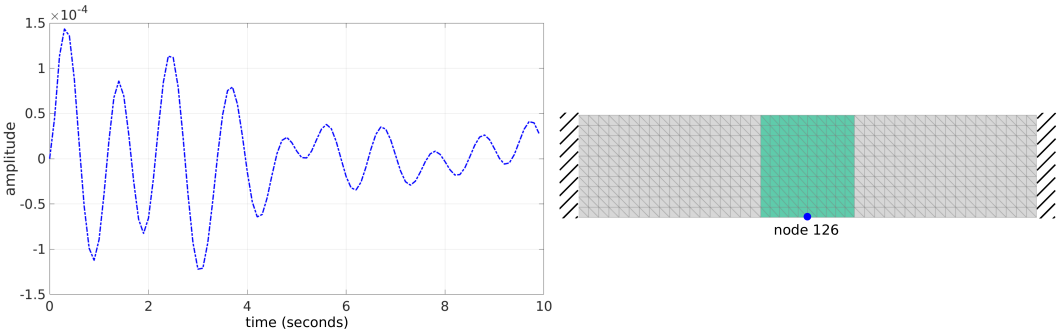


Figure 4.2: An example of unit impulse response $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$. Left sub-figure showing the amplitude of the response, which is the y-displacement of the centre node (right sub-figure showing the node location).

where δ_{tn} is the Kronecker Delta symbol. For simplicity of notations, mass, damping and stiffness components of reduced variables are no longer separated, and $\mathbf{a}_{j\mathfrak{r}}(t_n)$ is used to denote the reduced variables associated with acceleration, velocity and displacement. Thus eq. (4.19) becomes:

$$\boldsymbol{U}_{ij\mathfrak{r}}(\mu) = \sum_{n=0}^{N_t} \boldsymbol{U}_{ij\mathfrak{r}}^{imp} \delta_{tn} \mathbf{a}_{j\mathfrak{r}}(t_n;\mu) \quad \textbf{[SR]} \tag{4.20}$$

notice that $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}\delta_{tn}$ denotes the shift of displacement $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$ in time. Therefore the *full response vector* is transformed into displacement matrix. The shift of unit impulse responses is demonstrated in fig. 4.3.

$$\begin{gathered}\left(\text{Solving } \boldsymbol{A}_i \underline{\boldsymbol{X}}_{ij\mathfrak{r}}^{\mathfrak{r}}(\mu) = \boldsymbol{G}_{j\mathfrak{r}}^{imp}\right) \Longrightarrow \underline{\boldsymbol{X}}_{ij\mathfrak{r}}^{\mathfrak{r}}(\mu) \qquad \textbf{[FR]} \\ \text{is transformed to} \\ \left(\text{Newmark solution with shift in time}\right) \Longrightarrow \sum_{n=0}^{N_t} \boldsymbol{U}_{ij\mathfrak{r}}^{imp} \delta_{tn} \mathbf{a}_{j\mathfrak{r}}(t_n;\mu) \quad \textbf{[SR]}\end{gathered} \tag{4.21}$$

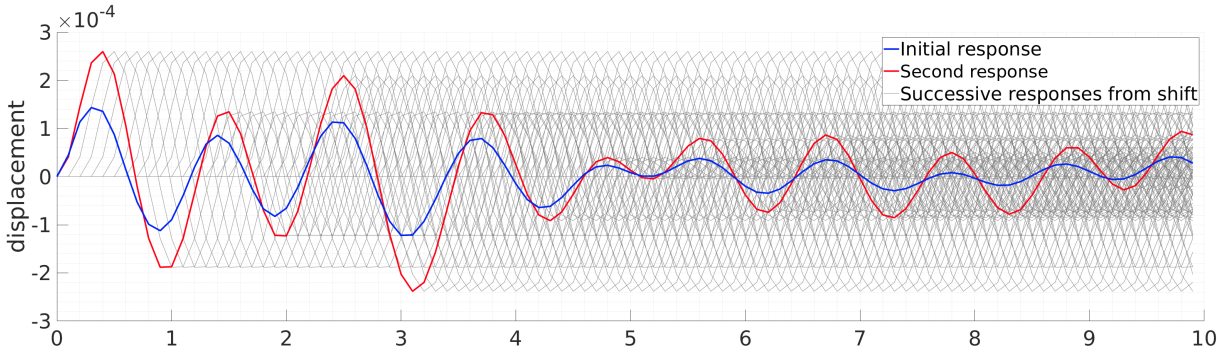


Figure 4.3: The shifted impulse responses response for a single degree of freedom: the blue curve denotes the response $\boldsymbol{U}^{imp}_{ij\mathfrak{r}1}$ same as shown in fig. 4.2, which is obtained by applying impulse at the initial time step; the red curve is the response $\boldsymbol{U}^{imp}_{ij\mathfrak{r}n}$ obtained by applying impulse at the second time step; once $\boldsymbol{U}^{imp}_{ij\mathfrak{r}n}$ is obtained, the rest responses (grey curves) are computed by shifting $\boldsymbol{U}^{imp}_{ij\mathfrak{r}n}$. The parametric displacement $\boldsymbol{U}_{ij\mathfrak{r}}(\boldsymbol{\mu})$ is reconstructed by shifting the impulse response and multiplying with the associated reduced variables, then performing a summation over time.

Remark For the i^{th} interpolation sample, the j^{th} affined term, the r^{th} reduced basis vector, only 2 responses need to be computed:

$$\boldsymbol{U}_{ij\mathfrak{r}n}, n = 1, 2 \tag{4.22}$$

where n denotes the time step number. This is because applying the impulse on the initial and second time step results in two different responses. However for $n \geq 2$, only 1 response, $\boldsymbol{U}_{ij\mathfrak{r}2}$, is required to be computed by applying Newmark method, the rest are obtained by shifting, as $\boldsymbol{U}^{imp}_{ij\mathfrak{r}n}$ is a unit impulse response which does not change in time, see fig. 4.3. This benefits the proposed error indicator by keeping the number of exact solutions under control thus preventing prohibited computational cost.

4.4.3 Approximated displacement error

It has been proven that the *full response vector* can be transformed to displacements, now substituting relationships eq. (4.21) and section 4.4.1 into eq. (4.12), the approximated displacement error in space and time is given by:

$$\hat{\boldsymbol{e}}(\mu) = \sum_{i=1}^{N_i} l_i(\mu)\Big(\boldsymbol{U}_i^{\mathrm{h}} - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\gamma_j(\mu)\boldsymbol{U}_{ij\mathfrak{r}}^{imp}\delta_{tn}\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\Big) \quad \textbf{[SR]} \tag{4.23}$$

A decomposition of the proposed POD-Greedy algorithm can be preliminarily utilised:

POD-Greedy *basis processing* stage:

- compute reduced basis $\boldsymbol{\phi}$,
- generate impulses, compute and shift the responses $\boldsymbol{U}_i^{\mathrm{h}}$ and $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$,

POD-Greedy *parameter sweep* stage:

- compute reduced variables,
- interpolate the pre-computed responses, multiply the results with corresponding affine coefficients and reduced variables, sum the results up.
- evaluate the error norm.

4.5 Calculation of error norm square

The approximated displacement error is measured using Frobenius norm. This can be calculated by evaluating the norm directly: $\|\hat{\boldsymbol{e}}(\mu)\|_{\mathrm{F}} = \sqrt{\mathrm{tr}\left(\left(\hat{\boldsymbol{e}}(\mu)\right)^T\left(\hat{\boldsymbol{e}}(\mu)\right)\right)}$. However, if interpolating the pre-computed responses $\boldsymbol{U}^{imp}_{ij\mathfrak{r}}$ as shown in eq. (4.23), the *parameter sweep* stage cost can be prohibitively high, as a number of responses would be interpolated for each individual parameter value. This can be circumvented by calculating the square of the approximated error norm in the POD-Greedy *basis processing* stage and interpolating this term in the POD-Greedy *parameter sweep* stage. In order to do this, the first step is expanding the square of the error norm as follows:

$$
\begin{aligned}
\|\hat{\boldsymbol{e}}(\mu)\|^2_{\mathrm{F}} &= \mathrm{tr}\left(\left(\hat{\boldsymbol{e}}(\mu)\right)^T\left(\hat{\boldsymbol{e}}(\mu)\right)\right) \\
&= \mathrm{tr}\Bigg(\Big(\sum_{i=1}^{N_i} l_i(\mu)\Big(\boldsymbol{U}^{\mathrm{h}}_i - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\gamma_j(\mu)\boldsymbol{U}^{imp}_{ij\mathfrak{r}}\delta_{tn}\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\Big)^T \\
&\qquad\qquad \Big(\boldsymbol{U}^{\mathrm{h}}_i - \sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t}\gamma_{j'}(\mu)\boldsymbol{U}^{imp}_{ij'\mathfrak{r}'}\delta_{tn'}\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu)\Big)\Big)\Bigg) \\
&= \mathrm{tr}\Bigg(\sum_{i=1}^{N_i} l_i(\mu)\Big(\left(\boldsymbol{U}^{\mathrm{h}}_i\right)^T\left(\boldsymbol{U}^{\mathrm{h}}_i\right) \\
&\quad - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\gamma_j(\mu)\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\left(\boldsymbol{U}^{imp}_{ij\mathfrak{r}}\delta_{tn}\right)^T\left(\boldsymbol{U}^{\mathrm{h}}_i\right) \\
&\quad - \sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t}\left(\boldsymbol{U}^{\mathrm{h}}_i\right)^T\left(\boldsymbol{U}^{imp}_{ij'\mathfrak{r}'}\delta_{tn'}\right)\gamma_{j'}(\mu)\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu) \\
&\quad - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t}\gamma_j(\mu)\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\left(\boldsymbol{U}^{imp}_{ij\mathfrak{r}}\delta_{tn}\right)^T\left(\boldsymbol{U}^{imp}_{ij'\mathfrak{r}'}\delta_{tn'}\right)\gamma_{j'}(\mu)\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu)\Big)\Bigg)
\end{aligned}
\tag{4.24}
$$

Equation (4.24) requires the computation of the matrix product trace, which can be avoided by vectorising the approximated error and computing the vector products. This is achieved by vectorizing the pre-computed responses hence obtain $\underline{\boldsymbol{U}} \in \mathbb{R}^{\mathcal{N}N_t}$, eq. (4.24)

becomes:

$$
\begin{aligned}
\|\hat{e}(\mu)\|_{\mathrm{F}}^2 &= \left\|\underline{\hat{e}}(\mu)\right\|_{\mathrm{F}}^2 \\
&= \left(\underline{\hat{e}}(\mu)\right)^T \left(\underline{\hat{e}}(\mu)\right) \\
&= \sum_{i=1}^{N_i} l_i(\mu) \Bigg(\left(\underline{\boldsymbol{U}}_i^{\mathrm{h}} - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t} \gamma_j(\mu)\underline{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp}\delta_{tn}\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\right)^T \\
&\qquad \left(\underline{\boldsymbol{U}}_i^{\mathrm{h}} - \sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t} \gamma_{j'}(\mu)\underline{\boldsymbol{U}}_{ij'\mathfrak{r}'}^{imp}\delta_{tn'}\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu)\right)\Bigg) \\
&= \sum_{i=1}^{N_i} l_i(\mu)\left(\underline{\boldsymbol{U}}_i^{\mathrm{h}}\right)^T\left(\underline{\boldsymbol{U}}_i^{\mathrm{h}}\right) \\
&- \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t} \gamma_j(\mu)\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\left(\underline{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp}\delta_{tn}\right)^T\left(\underline{\boldsymbol{U}}_i^{\mathrm{h}}\right) \\
&- \sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t} \left(\underline{\boldsymbol{U}}_i^{\mathrm{h}}\right)^T\left(\underline{\boldsymbol{U}}_{ij'\mathfrak{r}'}^{imp}\delta_{tn'}\right)\gamma_{j'}(\mu)\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu) \\
&- \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\sum_{j'=1}^{N_j}\sum_{\mathfrak{r}'=1}^{N}\sum_{n'=0}^{N_t} \gamma_j(\mu)\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\left(\underline{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp}\delta_{tn}\right)^T\left(\underline{\boldsymbol{U}}_{ij'\mathfrak{r}'}^{imp}\delta_{tn'}\right)\gamma_{j'}(\mu)\mathbf{a}_{j'\mathfrak{r}'}(t_{n'};\mu)\Bigg)
\end{aligned}
\tag{4.25}
$$

eq. (4.25) requires $O(N_i(N_jN_tN)^2)$ operations. Note that the response matrices in eq. (4.24) and vectors in eq. (4.25) are formulated in different ways when being shifted in time for the Duhamel's integral, see fig. 4.4. The following section explains the computational procedures to compute eq. (4.25) and how this procedure can reduce numerical cost of the POD-Greedy *parameter sweep*.

Computational procedures Equation (4.25) illustrates the expanded form of the approximated error norm square. In order to reduced the numerical cost of the Greedy *parameter sweep* stage, one needs to perform some special treatments to the pre-computed responses and scalar coefficients. Both the *basis processing* stage and the *parameter sweep* stage need to be modified, the treatments are explained as follows: first adding the following steps to the POD-Greedy *basis processing* stage,

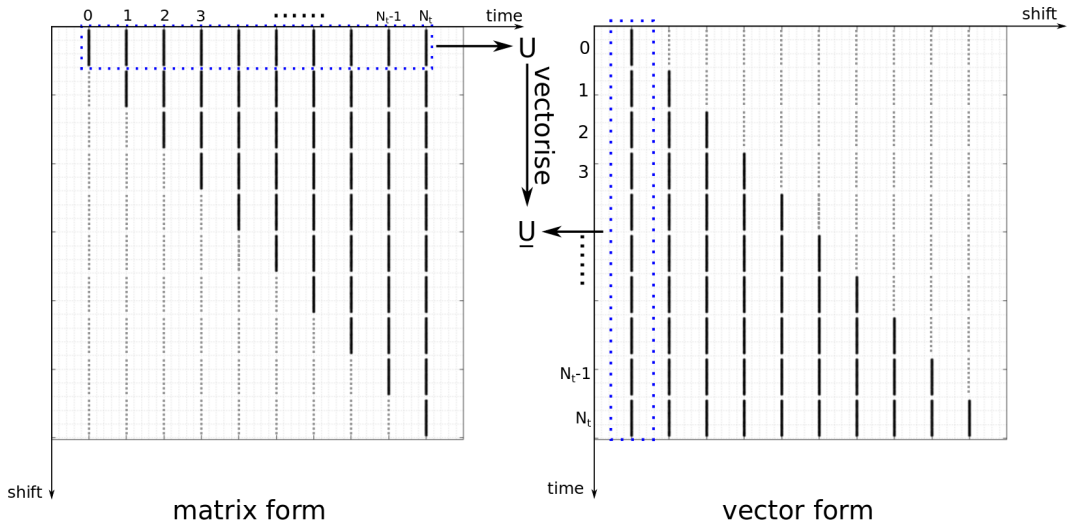


Figure 4.4: Shift of displacement in matrix and vector form. The grey dash lines indicate that the displacement vectors are replaced by zeros after shift.

- once the impulse responses are computed, vectorize them to obtain $\underline{\boldsymbol{U}}^{f}$ and $\underline{\boldsymbol{U}}^{imp}$ for the i^{th} interpolation sample, align these vectors in a matrix as:

$$\begin{cases} \boldsymbol{M}_i = \left[\underline{\boldsymbol{U}}_i^{\text{h}}, -\underline{\boldsymbol{U}}_{ij\text{r}}^{imp}\delta_{tn}\right]_{j,r=1,n=0}^{N_j,N,N_t} \\ \boldsymbol{M}_i \in \mathbb{R}^{NN_t\times(N_jNN_t)}. \end{cases} \tag{4.26}$$

- then performing:

$$\mathsf{M}_i^{\mathbf{trans}} = (\boldsymbol{M}_i)^T \boldsymbol{M}_i \tag{4.27}$$

the symmetric matrix $\mathsf{M}_i^{\mathbf{trans}}$ is then obtained, which is named as the *displacement vector product matrix*. $\mathsf{M}_i^{\mathbf{trans}} \in \mathbb{R}^{(N_jNN_t)\times(N_jNN_t)}$.

In the POD-Greedy *parameter sweep* stage,

- since the reduced variables $\mathbf{a}_{j\text{r}}(\mu)$ and the space-time responses are coupled, for each parameter value μ, the steps are: (i) compute $\mathbf{a}_{j\text{r}}(\mu)$, (ii) multiply the associated

affine-dependent coefficients $\gamma_j(\mu)$, (iii) align the scalar values in a column vector follow the same order as eq. (4.26), such that:

$$\begin{cases} \left(\underline{\mathbf{a}}(\mu)\right)^T = \left[1, \gamma_j(\mu)\mathbf{a}_{jrn}(\mu)\right]_{j,r=1,n=0}^{N_j,N,N_t} \\ \underline{\mathbf{a}} \in \mathbb{R}^{(N_j N_t N)\times 1} \end{cases} \tag{4.28}$$

- the parametric *displacement vector product matrix* is obtained by interpolation:

$$\mathsf{M}_i^{\mathbf{trans}}(\mu) = l_i(\mu)\mathsf{M}_i^{\mathbf{trans}} \tag{4.29}$$

- eq. (4.25) becomes:

$$\begin{aligned} \|\hat{e}(\mu)\|_{\mathrm{F}}^2 &= \mathrm{tr}\left(\left(\hat{e}(\mu)\right)^T\left(\hat{e}(\mu)\right)\right) \\ &= \left(\underline{\hat{e}}(\mu)\right)^T\left(\underline{\hat{e}}(\mu)\right) \\ &= \left(\underline{\mathbf{a}}(\mu)\right)^T\left(\mathsf{M}_i^{\mathbf{trans}}(\mu)\right)\underline{\mathbf{a}}(\mu) \\ &= \left(\underline{\mathbf{a}}(\mu)\right)^T l_i(\mu)\left(\mathsf{M}_i^{\mathbf{trans}}\right)\underline{\mathbf{a}}(\mu) \end{aligned} \tag{4.30}$$

  which gives the Frobenius norm of the approximated displacement error.

So now the terms being interpolated in the *parameter sweep* stage are the square symmetric *displacement vector product matrix* $\mathsf{M}_i^{\mathbf{trans}} \in \mathbb{R}^{(N_j N N_t)\times(N_j N N_t)}$. Since $N_j N \ll \mathcal{N}$, dimension of $\mathsf{M}_i^{\mathbf{trans}} \ll$ dimension of $\boldsymbol{U}$. Compare with interpolating responses $\boldsymbol{U} \in \mathbb{R}^{\mathcal{N}\times N_t}$, interpolating $\mathsf{M}_i^{\mathbf{trans}}$ is much more efficient.

4.6 Preliminary comparison between proposed error indicator and residual as error indicator

Now that the main calculation procedures of approximated displacement error are derived, it is necessary to investigate the comparative results of approximated and exact displacement error, such that our theory can be verified and nature of the new error estimate can be better understood. A test based on the simple beam model in section 2.2.5

is set up, except this time the number of time steps are further simplified to $N_t = 10$, and the time quantity of interest is set to be time steps $[3, 5, 7]$. Again a discretised training set $\{\mu_i, b_j\}_{i,j=1}^{17,17}$ is used, results in 289 samples. 10 POD-Greedy iterations are performed, calculating both true RB-error (eq. (2.21), displacement error) and approximated displacement error (eq. (4.23)). Let the number of initial basis vectors $N^{\text{init}} = 2$ and enriched basis vectors $N^{\text{add}} = 2$, thus 10 Greedy iterations result in $N = 20$ basis vectors. The test contains the following components:

- the error response surfaces constructed using true RB-error indicator and the proposed approximated error indicator are plotted and compared, they are expected to match in the interpolation samples.

- an alternative to true RB-error is provided in [12]: using squared norm of the residual as an error indicator. More specifically, in step $4-5$ of algorithm 2, $||\boldsymbol{R}||_2^2$ is measured to indicate the error (see eq. (2.25) for residual). Numerical efficiency is greatly improved as no exact solution needs to be solved. However, results in [12] is not satisfying as convergence of the residual indicator does not approach the true RB-error. In this experiment, performance of different error indicators is evaluated by comparing the output convergence of (i) true RB-error indicator, (ii) proposed approximated error indicator eq. (4.23), (iii) residual indicator.

Output 1: error response surfaces

Error response surfaces are constructed for true RB-error (the exact displacement error) and approximated displacement error, respectively, and the distance are also evaluated. Surfaces of Greedy iterations 1-5 are displayed in fig. 4.6. A 3 interpolation sample set is being used in this test, see fig. 4.5 for the training interpolation sample set.

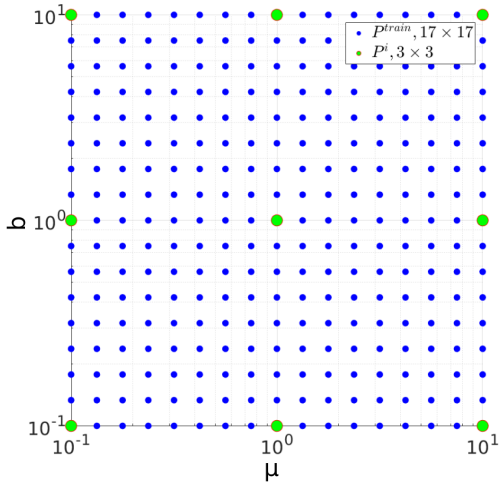


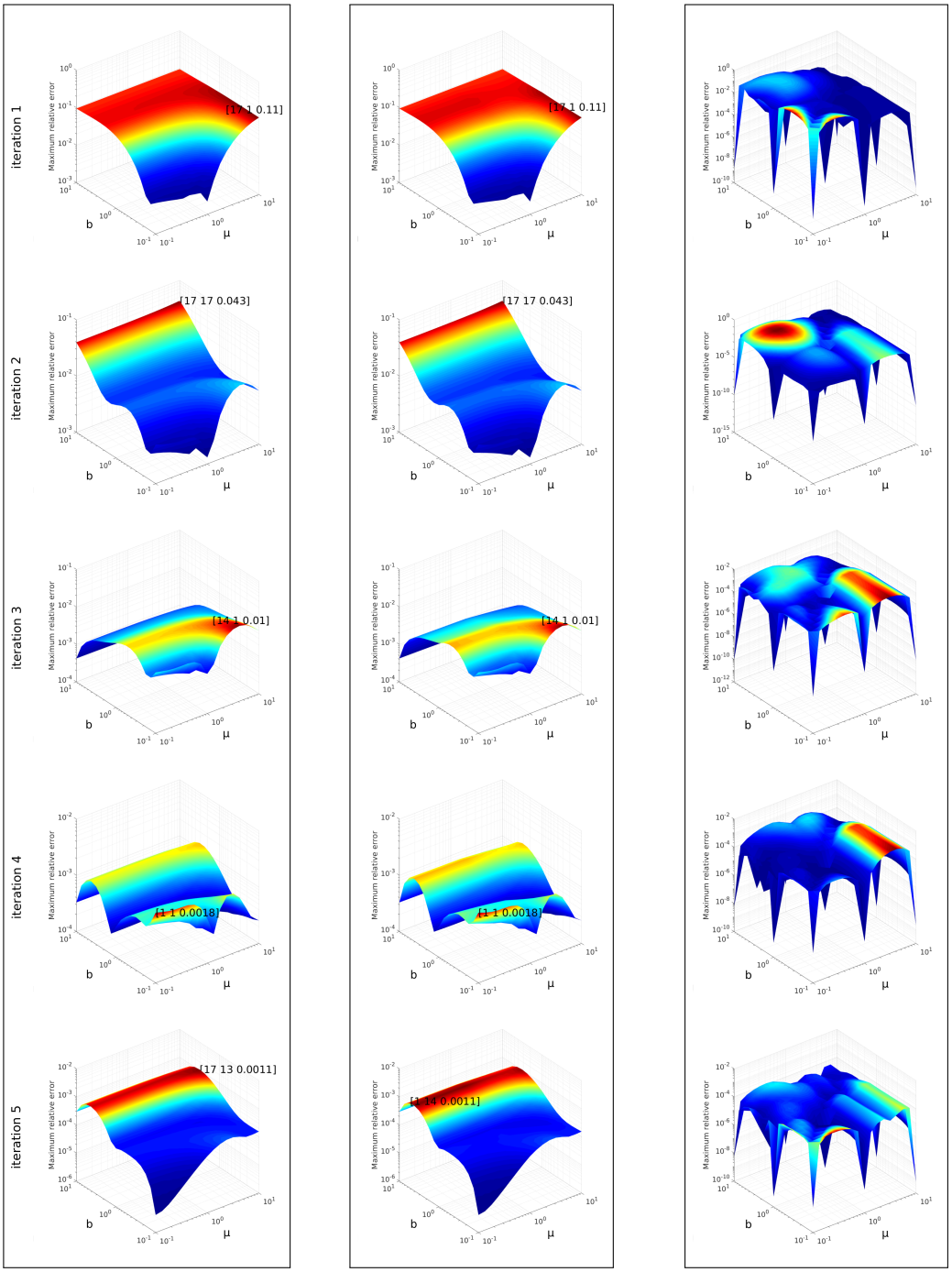
Figure 4.5: The training set $\mathcal{P}^{\text{train}}$ and interpolation sample set $\mathcal{P}^{i}$.

Output 2: maximum relative error convergence

The maximum relative error convergence using 3 error indicators are proposed in fig. 4.7. For residual indicator, in order to guarantee a equal comparison, first the POD-Greedy algorithm is performed using the residual as error indicator, then the true RB-error convergence is evaluated based on the output magic points and reduced basis. 2 initial magic points $[\mu, b] = [10^{-1}, 10^{-1}]$ and $[\mu, b] = [10^{1}, 10^{1}]$ are chosen as the cases to test the indicators. The following remarks can be made by observing the results:

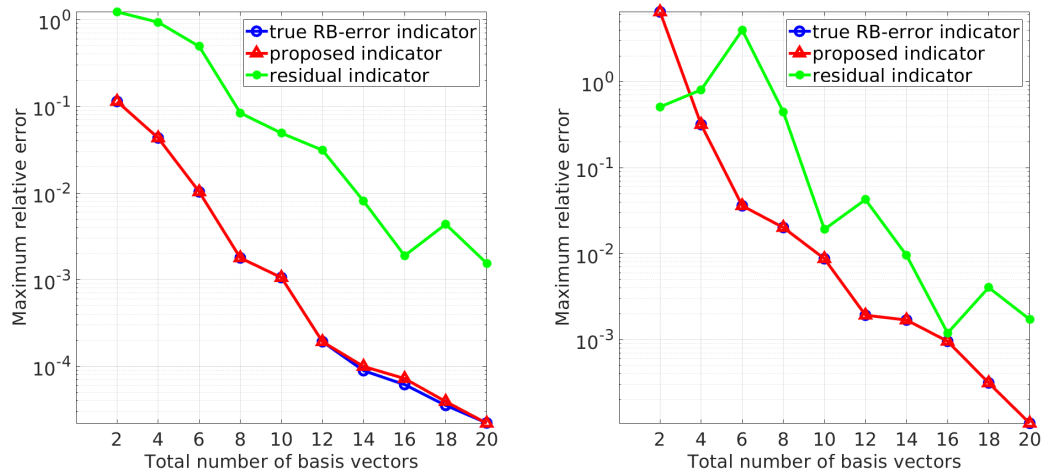
- convergence of the proposed error indicator approaches the true RB-error indicator for both test cases due to the good approximation of error response surfaces as shown in fig. 4.6. Hence the proposed error indicator is proven to be effective at this stage.

- convergence of residual indicator generally achieves an order of magnitude smaller than the other 2 options after 10 POD-Greedy iterations. One may also notice



(a) True RB-error response surfaces (b) Proposed error response surfaces (c) Response surfaces of the difference

Figure 4.6: Evolution of error response surfaces for Greedy iteration 1-5, initial magic point $[\mu, b] = [10^{-1}, 10^{-1}]$. The maximum error points (magic points) are labelled on the surfaces. The result shows that even with linear interpolation, the approximated response surfaces match relatively well with the exact surfaces. Notice that in fig. 4.6c, the numerical distance approaches 0 at the interpolation samples. The surfaces are shown in 3D view for better visibility.



(a) Initial magic point: $[\mu, b] = [10^{-1}, 10^{-1}]$ (b) Initial magic point: $[\mu, b] = [10^{1}, 10^{1}]$

Figure 4.7: Compare the maximum error convergences obtained from the 3 error indicators, results showing 2 different initial magic points as 2 test cases.

that for the three error indicators in this case, selecting the same initial magic point does not lead to the same error value. The reason is: residual is obtained by applying eq. (2.25), which results in a totally different quantity from true-RB error or the approximation. Therefore even the initial error is different. Performance of residual indicator is also not satisfying as the convergence trajectory diverges from the reference. This might be intuitively expected since they are different. However the residual indicator is much less resource intensive than the other 2 choices, thus results in a general trade-off: if accuracy of the approximation is the priority, then either true RB-error or proposed indicator should be used; otherwise residual indicator can only be chosen as an indicator when low numerical cost and fast evaluation speed are priorities.

4.7 Space-time reduction on impulse responses

For the proposed algorithm, one of the main computational cost is the computation and storage of the exact solutions $\boldsymbol{U}_i^{\mathrm{h}}$ and $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$. The aim is to further optimise computational cost by compressing these exact solutions with Singular Value Decomposition. The original displacement $\boldsymbol{U} \in \mathbb{R}^{\mathcal{N}\times N_t}$, is decomposed into orthonormal matrices $\boldsymbol{V_L} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$, $\boldsymbol{V_R} \in \mathbb{R}^{N_t\times N_t}$ and singular matrix $\Sigma \in \mathbb{R}^{\mathcal{N}\times N_t}$. Σ is a rectangular matrix and the diagonal entries of its upper square contain decreasingly ordered singular values $\{\sigma_i\}_{i=1}^{n}$.

Applying SVD to the pre-computed displacements results in:

$$\begin{aligned}\boldsymbol{U}_i^{\mathrm{h}} &= \boldsymbol{V}_{iL}^{\mathrm{h}}\boldsymbol{\Sigma}_i^{\mathrm{h}}\left(\boldsymbol{V}_{iR}^{\mathrm{h}}\right)^T \\ \boldsymbol{U}_{ij\mathfrak{r}}^{imp} &= \boldsymbol{V}_{ij\mathfrak{r}L}^{imp}\boldsymbol{\Sigma}_{ij\mathfrak{r}}^{imp}\left(\boldsymbol{V}_{ij\mathfrak{r}R}^{imp}\right)^T\end{aligned} \tag{4.31}$$

in order to optimize computational storage, a low-rank approximation of the displacements is required. Therefore a truncated SVD is performed by selecting the first k ($N_k \ll \mathcal{N}$) largest singular values and the corresponding left and right singular vectors and ignoring the rest: $\boldsymbol{U} \approx \tilde{\boldsymbol{U}} = \tilde{\boldsymbol{V}}_L\tilde{\boldsymbol{\Sigma}}(\tilde{\boldsymbol{V}}_R)^T$. The approximated displacements are (assume N_k^{resp} singular vectors are being used):

$$\begin{aligned}\boldsymbol{U}_i^{\mathrm{h}} \approx \tilde{\boldsymbol{U}}_i^{\mathrm{h}} &= \tilde{\boldsymbol{V}}_{iL}^{\mathrm{h}}\tilde{\boldsymbol{\Sigma}}_i^{\mathrm{h}}\left(\tilde{\boldsymbol{V}}_{iR}^{\mathrm{h}}\right)^T = \sum_{k=1}^{N_k^{\mathrm{resp}}}\Sigma_{ik}^{\mathrm{h}}\boldsymbol{V}_{iLk}^{\mathrm{h}}\left(\boldsymbol{V}_{iRk}^{\mathrm{h}}\right)^T \\ \boldsymbol{U}_{ij\mathfrak{r}}^{imp} \approx \tilde{\boldsymbol{U}}_{ij\mathfrak{r}}^{imp} &= \tilde{\boldsymbol{V}}_{ij\mathfrak{r}L}^{imp}\tilde{\boldsymbol{\Sigma}}_{ij\mathfrak{r}}^{imp}\left(\tilde{\boldsymbol{V}}_{ij\mathfrak{r}R}^{imp}\right)^T = \sum_{k=1}^{N_k^{\mathrm{resp}}}\Sigma_{ijrk}^{imp}\boldsymbol{V}_{ijrLk}^{imp}\left(\boldsymbol{V}_{ijrRk}^{imp}\right)^T\end{aligned} \tag{4.32}$$

substituting eq. (4.32) in eq. (4.23) ends up with the following approximation of $\hat{\boldsymbol{e}}(\mu)$:

$$\hat{e}(\mu) \approx \sum_{i=1}^{N_i} l_i(\mu)\left(\tilde{\boldsymbol{V}}_{iL}^{\mathrm{h}}\tilde{\boldsymbol{\Sigma}}_i^{\mathrm{h}}\left(\tilde{\boldsymbol{V}}_{iR}^{\mathrm{h}}\right)^T - \sum_{j=1}^{N_j}\sum_{\mathfrak{r}=1}^{N}\sum_{n=0}^{N_t}\gamma_j(\mu)\tilde{\boldsymbol{V}}_{ij\mathfrak{r}L}^{imp}\tilde{\boldsymbol{\Sigma}}_{ij\mathfrak{r}}^{imp}\left(\tilde{\boldsymbol{V}}_{ij\mathfrak{r}R}^{imp}\delta_{tn}\right)^T\mathbf{a}_{j\mathfrak{r}}(t_n;\mu)\right) \tag{4.33}$$

application of SVD on pre-computed displacements leads to the following benefits:

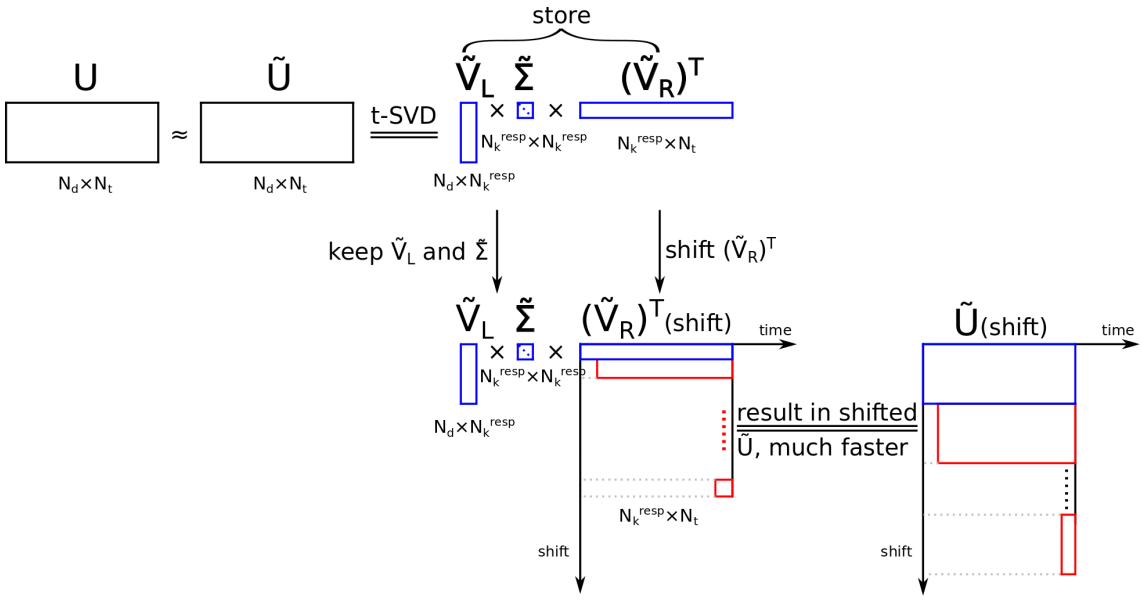


Figure 4.8: Store SVD vectors to optimize storage, shift right singular matrix to accelerate the shift.

- after truncation, left singular matrix $\tilde{\boldsymbol{V}}_{\boldsymbol{L}} \in \mathbb{R}^{\mathcal{N} \times N_k^{\text{resp}}}$, sparse singular value matrix $\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{N_k^{\text{resp}} \times N_k^{\text{resp}}}$ and right singular matrix $\tilde{\boldsymbol{V}}_{\boldsymbol{R}} \in \mathbb{R}^{N_t \times N_k^{\text{resp}}}$ are stored. Since $N_k^{\text{resp}} \ll \mathcal{N}$, computational storage cost is greatly reduced compared with storing the full response matrices.

- during the Duhamel's integral, instead of shifting the entire displacement matrix in time (see section 4.4.2), now the right singular matrix $\tilde{\boldsymbol{V}}_{\boldsymbol{R}}$ is shifted. Due to the small size of $\tilde{\boldsymbol{V}}_{\boldsymbol{R}}$, the shift efficiency is greatly improved.

4.8 POD on reduced variables

4.8.1 Analysis of POD-Greedy *parameter sweep* stage cost with the new error indicator

In *POD-Greedy* algorithm, the Greedy iteration yields 2 results: the enrichment of the reduced basis $\boldsymbol{\phi}$, and the upgrade of the reduced variables $\boldsymbol{\alpha}$. However, there is a fundamental difference between these two results: the reduced basis is enriched hierarchically, while the reduced variable is updated to complete new data. See fig. 4.9 for a schematic illustration. This difference leads to both an advantage and a disadvantage for the new error estimate:

- advantage: when calculating the pre-computed responses, due to the hierarchical structure, only the ones associated with the N^{add} newly added basis vectors need to be computed at each Greedy iteration. More specifically, the number of newly computed exact solutions for each Greedy iteration is $N_i \times N_j \times N^{\text{add}} \times 2$ (see section 4.4.2 for why this number is doubled here). Compare with the *reference POD-Greedy* algorithm which requires N_{train} exact solutions each iteration, the proposed method pays off if (i) the FE model is large, (ii) the training set is highly-discretised, i.e. N_{train} is large.

- disadvantage: in the *parameter sweep* stage, when evaluating the displacement vector product matrix $\mathsf{M}_i^{\mathbf{trans}}$, all the information from the previous Greedy iterations needs to be stored. This is because the upgrade of the reduced variables results in completely new values, and the pre-computed displacements $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$ need to be coupled with the reduced variables $\mathbf{a}_{j\mathfrak{r}}(t_n;\boldsymbol{\mu})$. Moreover, the full $\mathsf{M}_i^{\mathbf{trans}}$ needs to be interpolated in the *parameter sweep*, which can be expensive if the reduced basis possesses a complex structure.

  The dimensional increase of $\mathsf{M}_i^{\mathbf{trans}}$ due to the re-generation of reduced variables during the Greedy procedure is further explained here: after N_G Greedy itera-

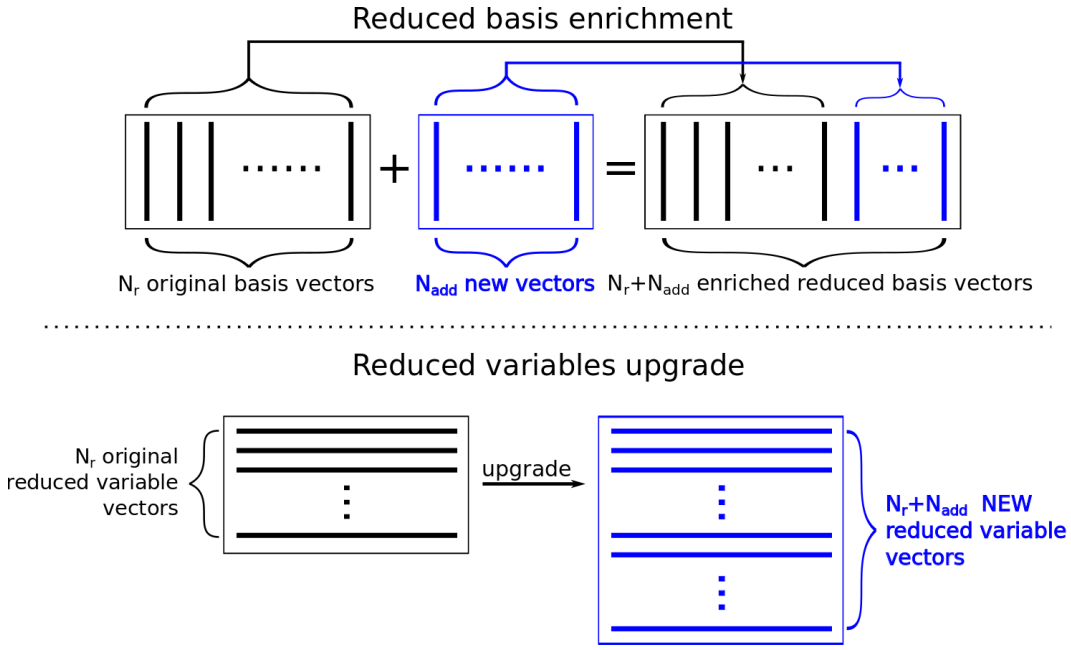


Figure 4.9: The difference between reduced basis enrichment and reduced variable upgrade: former is hierarchical and latter is not.

tions, the dimension of the reduced variables associated with the j^{th} affined term increases from $N \times N_t$ to $(N + N_G N^{\text{add}}) \times N_t$. Dimension of $\mathsf{M}_i^{\mathbf{trans}}$ also increases from $\left(N_j N_t N\right) \times \left(N_j N_t N\right)$ to $\left(N_j N_t (N + N_G N^{\text{add}})\right) \times \left(N_j N_t (N + N_G N^{\text{add}})\right)$. In *POD-Greedy parameter sweep* stage, one has to deal with the full matrix $\mathsf{M}_i^{\mathbf{trans}}$, and the dimension of this matrix increases by $N_j N_t N^{\text{add}}$ within each *POD-Greedy* iteration. The computational cost of the *parameter sweep* increases quickly with increasing Greedy iterations, which is the limitation of the proposed POD-Greedy algorithm so far. See fig. 4.10 for a schematic explanation of how $\mathsf{M}_i^{\mathbf{trans}}$ dimension increases with reduced basis enrichment.

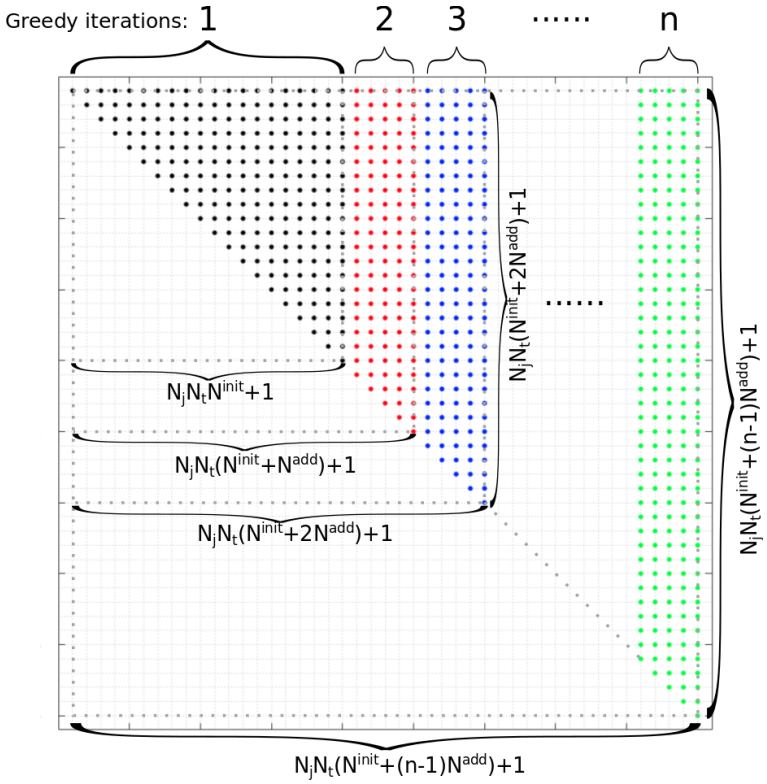


Figure 4.10: Dimension of $\mathsf{M}_i^{\mathbf{trans}}$, which increases with enrichment of reduced basis, each dot denotes the vector product $\left(\underline{\boldsymbol{U}}_{ij\tau}^{imp}\delta_{tn}\right)^T\left(\underline{\boldsymbol{U}}_{ij'\tau'}^{imp}\delta_{tn'}\right)$.

4.8.2 Speed-up the POD-Greedy *parameter sweep* stage: POD on reduced variables

The limitation of the proposed POD-Greedy algorithm so far lies in the cost of parameter sweep. When dealing with the training samples, the algorithm should maintain a reasonably low-cost for each parametric value to guarantee a quick sweep. This is yet limited due to dimensional increase of $\mathsf{M}_i^{\mathbf{trans}}$. This problem can be tackled by (i) shifting more workload from *parameter sweep* stage to *basis processing* stage, (ii) keeping size of $\mathsf{M}_i^{\mathbf{trans}}$ a constant. Proper Orthogonal Decomposition is chosen to be applied to the collection of reduced variables, so that the dimension of the matrix being interpolated remains a constant after each Greedy iteration.

The procedure is as follows: first collect the parameter-dependent full reduced variable vectors $\underline{\mathbf{a}}$ over the entire training set $\mathcal{P}^{\text{train}}$, N_{train} denotes number of collected vectors. The reduced variable vector $\underline{\mathbf{a}}$ is obtained by vectoring the reduced variable matrix $\mathbf{a}$, see eq. (4.20), thus $\underline{\mathbf{a}} \in \mathbb{R}^{N_j N_t N}$. The collection is:

$$\begin{cases} \mathcal{W}_\alpha = \left[\underline{\mathbf{a}}(\mu_i)\right]_{i=1}^{N_{\text{train}}}, \mu_i \in \mathcal{P}^{\text{train}} \\ \mathcal{W}_\alpha \in \mathbb{R}^{(N_j N_t N)\times N_{\text{train}}} \end{cases} \tag{4.34}$$

$\mathcal{W}_\alpha$ is named as the *reduced variable vector matrix*. Invoking truncated SVD and select the first N_k ($N_k \ll N_{\text{train}}$) largest singular values and the corresponding left and right singular vectors (assume N_k^{rv} singular vectors are being used here) gives:

$$\mathcal{W}_\alpha \approx \tilde{\mathcal{W}}_\alpha = \tilde{\boldsymbol{V}}_{\alpha L}\tilde{\boldsymbol{\Sigma}}_\alpha\left(\tilde{\boldsymbol{V}}_{\alpha R}\right)^T = \sum_{k=1}^{N_k^{\text{rv}}} \tilde{\boldsymbol{\Sigma}}_{\alpha k}\tilde{\boldsymbol{V}}_{\alpha Lk}\left(\tilde{\boldsymbol{V}}_{\alpha Rk}\right)^T \tag{4.35}$$

where $\tilde{\mathcal{W}}_\alpha$ is the closest approximation of $\mathcal{W}_\alpha$ of order N_k^{rv}. $\tilde{\boldsymbol{V}}_{\alpha L} \in \mathbb{R}^{N_j N_t N \times N_k^{\text{rv}}}$, $\tilde{\boldsymbol{\Sigma}}_\alpha \in \mathbb{R}^{N_k^{\text{rv}}\times N_k^{\text{rv}}}$, $\tilde{\boldsymbol{V}}_{\alpha R} \in \mathbb{R}^{N_\mu \times N_k^{\text{rv}}}$. Recall that $\mathsf{M}_i^{\mathbf{trans}} \in \mathbb{R}^{N_j N_t N \times N_j N_t N}$, thus $\mathsf{M}_i^{\mathbf{trans}}$ and $\tilde{\boldsymbol{V}}_{\alpha L}$

can be coupled, which is a projection of $\mathsf{M}_i^{\mathbf{trans}}$ on the subspace spanned by $\tilde{\boldsymbol{V}}_{\alpha L}$:

$$\mathsf{M}_i^{\mathbf{trans},\mathrm{r}} = (\tilde{\boldsymbol{V}}_{\alpha L})^T (\mathsf{M}_i^{\mathbf{trans}}) \tilde{\boldsymbol{V}}_{\alpha L} \tag{4.36}$$

this is computed in the Greedy *basis processing* stage. In the Greedy *parameter sweep* stage, $\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}$ is interpolated and the result is multiplied with the singular value matrix and right singular vector matrix as follows:

$$\mathsf{M}^{\mathbf{trans},\mathrm{r}} = \tilde{\boldsymbol{V}}_{\alpha R} \tilde{\boldsymbol{\Sigma}}_{\alpha} l_i(\mu) \left(\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}\right) \tilde{\boldsymbol{\Sigma}}_{\alpha} \left(\tilde{\boldsymbol{V}}_{\alpha R}\right)^T \tag{4.37}$$

where $\mathsf{M}^{\mathbf{trans},\mathrm{r}} = \mathrm{diag}\left(m_1, \ldots, m_{N_{\mathrm{train}}}\right) \in \mathbb{R}^{N_{\mathrm{train}} \times N_{\mathrm{train}}}$. One may notice a major difference between the approximation $\mathsf{M}^{\mathbf{trans},\mathrm{r}}$ and our target $\|\hat{\boldsymbol{e}}(\mu)\|_{\mathrm{F}}^2$ (see eq. (4.30)): $\|\hat{\boldsymbol{e}}(\mu)\|_{\mathrm{F}}^2$ is the scalar value which denotes Frobenius norm of the approximated error at parametric value μ; $\mathsf{M}^{\mathbf{trans},\mathrm{r}}$ is an $N_{\mathrm{train}} \times N_{\mathrm{train}}$ matrix, and the i^{th} diagonal entry denotes Frobenius norm of the approximated error at parametric value μ_i. Therefore once $\mathsf{M}^{\mathbf{trans},\mathrm{r}}$ is obtained, the i^{th} diagonal entry of $\mathsf{M}^{\mathbf{trans},\mathrm{r}}$, m_i, needs to be extracted:

$$\|\hat{\boldsymbol{e}}(\mu_i)\|_{\mathrm{F}}^2 \approx m_i, \forall \mu_i \in \mathcal{P}^{\mathrm{train}} \tag{4.38}$$

eq. (4.37) interpolates the reduced *displacement vector product matrix* $\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}$, which solves the problem due to these facts: (i) dimension of $\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}$ is fixed to $N_k^{\mathrm{rv}} \times N_k^{\mathrm{rv}}$ hence no longer increases with Greedy iterations, (ii) $N_k^{\mathrm{rv}} \ll N_j N_t N$, thus the term being interpolated in the Greedy *parameter sweep* stage becomes much smaller, which allows one to evaluate a considerably large training set thus no key information is lost. If let $N_k^{\mathrm{rv}} = N_{\mathrm{train}}$, then eq. (4.37) should yield the same result as eq. (4.30).

4.9 Summary

A new error indicator for parametric elasto-dynamic problems are presented in this chapter, which allows rapid error evaluation and use of large training set. This chapter is started with the introduction of current training set treatment approaches, leading to the motivation of developing this new error indicator. In section 4.2, the exact full error vector is derived based on the full space-time Newmark representation. ROM is then involved given the approximated full error vector in section 4.3. The output of interest: displacement error, is derived in section 4.4. The full representation is decomposed into a set of impulse response problems, then the output impulse responses are linearly combined to form the displacement error indicator. Its use is preliminarily verified in section 4.6, showing accuracy and comparing it with residual as indicator. Finally reduction on impulse responses (section 4.7) and POD on reduced variables (section 4.8) are proposed to further reduce the numerical cost of the new error indicator.

Chapter 5

The "*Error in the error*" *indicator* in Dynamics

The chapter is organised as follows: in section 5.1, a new *error in the error indicator* is introduced. Two interpolation sample grids are generated for this indicator: a coarse 'slave' grid and a locally refined 'master' grid. The error indicator is driven by a user-defined tolerance; the numerical distance between 'master' and 'slave' response surfaces are evaluated: if the distance exceeds the tolerance, the algorithm performs the refinement and ceases the Greedy iterations. The refinement follows a principle: the two grids are refined locally according to the location of the maximum distance only within the interpolation sample block which possesses the largest error. Numerical examples are proposed in section 5.2 to demonstrate the error indicator and how training set size affects it.

5.1 Local $h-$refinement and the "*error in the error*" *indicator*

The $h-$ and $p-$ versions of finite element methods are methods which aim at improving the accuracy of standard FE simulation [5]. The $h-$refinement uses the same type of element but a finer mesh, while the $p-$refinement increases order of shape functions but keeps mesh size unchanged. The two approaches can be suitably combined to reach an optimised accuracy. A question naturally arises when introducing interpolation in the algorithm, that is, is a local domain $h-$refinement needed to refine the interpolation sample domain thus accuracy of the interpolation can be improved? In order to provide a solution, a new definition namely "*error in the error*" *indicator* is introduced to provide a solution. The original interpolation sample grid is $\mathcal{P}^i = \{\mu_i\}_{i=1}^{N_i}$. Two interpolation sample grids are specifically designed for this new error indicator: a coarse 'slave' grid $\hat{\mathcal{P}}^i$ and a locally refined 'master' grid $\hat{\hat{\mathcal{P}}}^i$. The following remarks are made:

- the new "*error in the error*" *indicator* requires a coarse 'slave' sample grid and a locally refined 'master' sample grid. $\hat{N}_b$ and $\hat{\hat{N}}_b$ blocks are generated by defining these 2 sample grids.
- dimensions of the error response surface equals to the number of affined parameters N_j. The new error estimate requires the parametric domain to be divided into $\hat{\hat{N}}_b$ and $\hat{N}_b$ blocks. For the initial Greedy iteration, $\hat{\hat{N}}_b = 2^{N_j}$, $\hat{N}_b = 1$. As a result, the number of interpolation sample points are $\hat{\hat{N}}_i = 2^{N_j}$ and $\hat{N}_i = 3^{N_j}$, respectively.
- once a Greedy iteration is performed, 2 error response surfaces $\hat{\hat{\boldsymbol{e}}}(\mu)$ and $\hat{\boldsymbol{e}}(\mu)$, $\forall \mu \in \mathcal{P}^{\text{train}}$ are constructed based on $\hat{\hat{\mathcal{P}}}^i$ and $\hat{\mathcal{P}}^i$, respectively. The following maximum relative numerical distance is defined as the "*error in the error*" *indicator*

$$\boldsymbol{e}^e := \max \left| \frac{||\hat{\hat{\boldsymbol{e}}}(\mu)||_F - ||\hat{\boldsymbol{e}}(\mu)||_F}{||\boldsymbol{U}(\mu_0)||_F} \right| \tag{5.1}$$

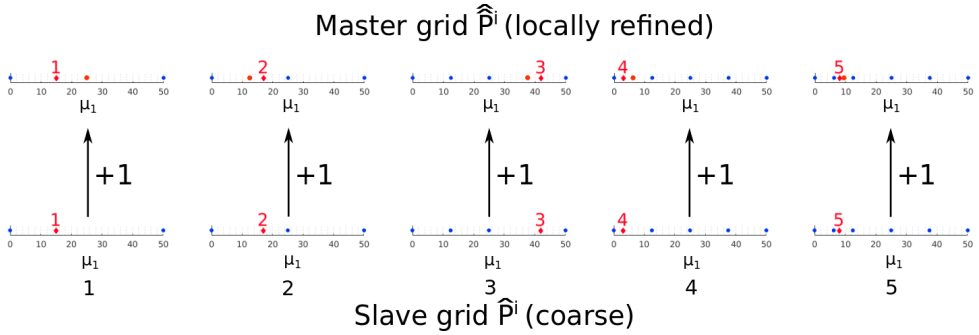
- $\boldsymbol{e}^e$ needs to be coupled with a user-defined tolerance tol^{ref}: if $\boldsymbol{e}^e > \text{tol}^{\text{ref}}$, the local $h-$refinement is performed at the interpolation block which possesses the largest error; otherwise another Greedy iteration is started.

The local refinement guarantees the reduction of the numerical distance between the master and slave surfaces. The reason to choose $\boldsymbol{e}^e$ as the error estimate is $\lim_{n \to inf} \hat{\boldsymbol{e}} = \boldsymbol{e}$, that is, an infinite refinement of the interpolation sample grid eventually leads to the exact error. Ultimately the local h-refinement results in full reproduction of the exact error, although in practice this is unachievable as excessive refinement leads to prohibitive computational cost. See [34] for local $h-$refinement.

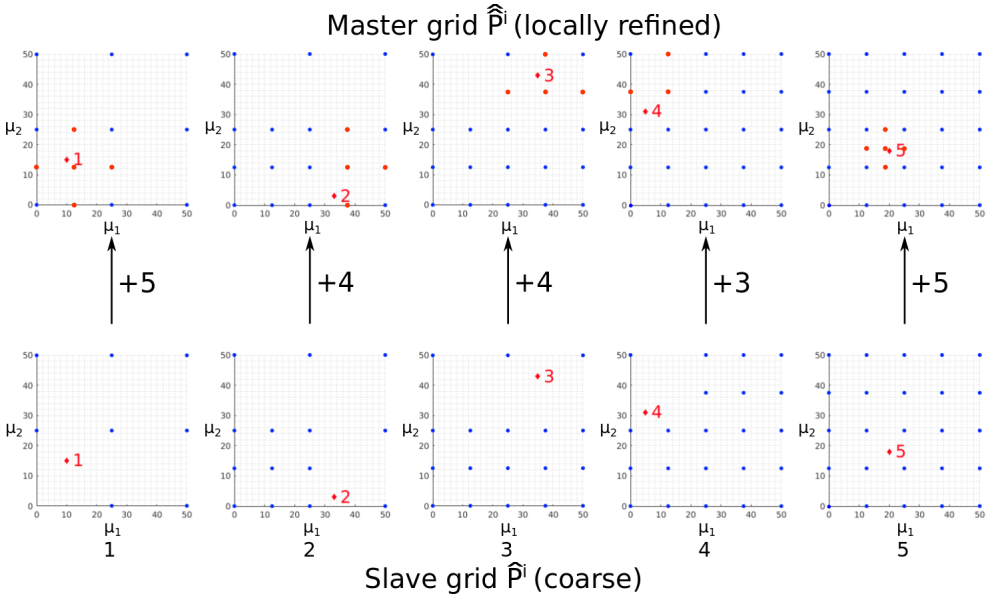
The following principle is followed when choosing refinement locations: first the parametric domain is searched over to obtain the sample block which possesses the largest error, then the distance of error response surfaces between $\hat{\hat{\boldsymbol{e}}}$ and $\hat{\boldsymbol{e}}$ are evaluated to see if it exceeds the pre-set tolerance. The rationales behind the principle are (i) this thesis aims to capture the magic points at the maximum error, thus a precise interpolation is required only in the associated region; (ii) if the region with low error is considered, the domain might be refined excessively which leads to unnecessary and possible prohibitive computational cost. The scalar quantity $||\boldsymbol{U}(\mu_0)||_F$ is chosen as denominator in $\boldsymbol{e}^e$ because this ensures value of the "*error in the error*" *indicator* is always reduced after the refinement. Another form of "*error in the error*" *indicator* is defined as follows: $\tilde{\boldsymbol{e}^e} := \max\left|\frac{||\hat{\hat{\boldsymbol{e}}}(\mu)||_F - ||\hat{\boldsymbol{e}}(\mu)||_F}{||\hat{\hat{\boldsymbol{e}}}(\mu)||_F}\right|$. Test results show that $\tilde{\boldsymbol{e}^e}$ is inappropriate as $||\hat{\hat{\boldsymbol{e}}}(\mu)||_F$ may also change it's value after the refinement, hence $\boldsymbol{e}^e$ may increase which cannot truly represent the refinement property (in the same interpolation block, the numerical distance must reduce after the refinement).

5.2 Numerical results

The model to evaluate performance of the "*error in the error*" *indicator* is inherited from section 4.6. In order to demonstrate the results visually, one needs to plot the response surfaces of (i) true RB-error, (ii) $\hat{\hat{\boldsymbol{e}}}$ approximation, (iii) $\hat{\boldsymbol{e}}$ approximation in 1 figure. Thus the model is further simplified to possess only 1 parameter, and 1 Greedy iteration is performed. The effect of "*error in the error*" indicator is investigated as a



(a) Demonstration of 5 local refinements for 1 parameter μ_1, each refinement adds 1 sample point.



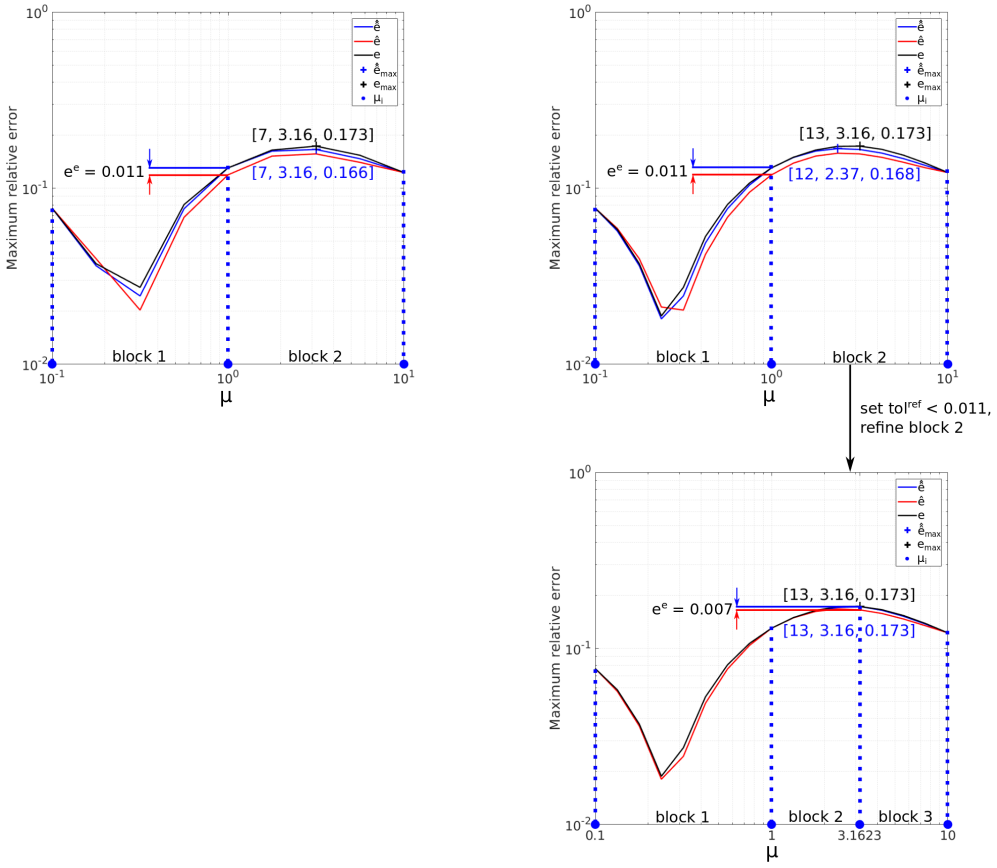
(b) Demonstration of 5 local refinements for 2 parameters $[\mu_1, \mu_2]$, each refinement may add 3, 4 or 5 sample points.

Figure 5.1: An example of 5 local refinements showing both slave grid $\hat{N}_b$ and master grid $\hat{\hat{N}}_b$. For each sub-figure, Bottom 5 figures show the coarse 'slave' grids and top 5 figures are the corresponding locally refined 'master' grids. The red diamonds with digits denote locations of the maximum distance which drives the local refinement. The blue dots are existing interpolation samples, the red dots denote the newly added vertices.

function of training set size N_{train}.

Observing evolutions of error response surfaces in fig. 5.2 and fig. 5.3, the following remarks are made:

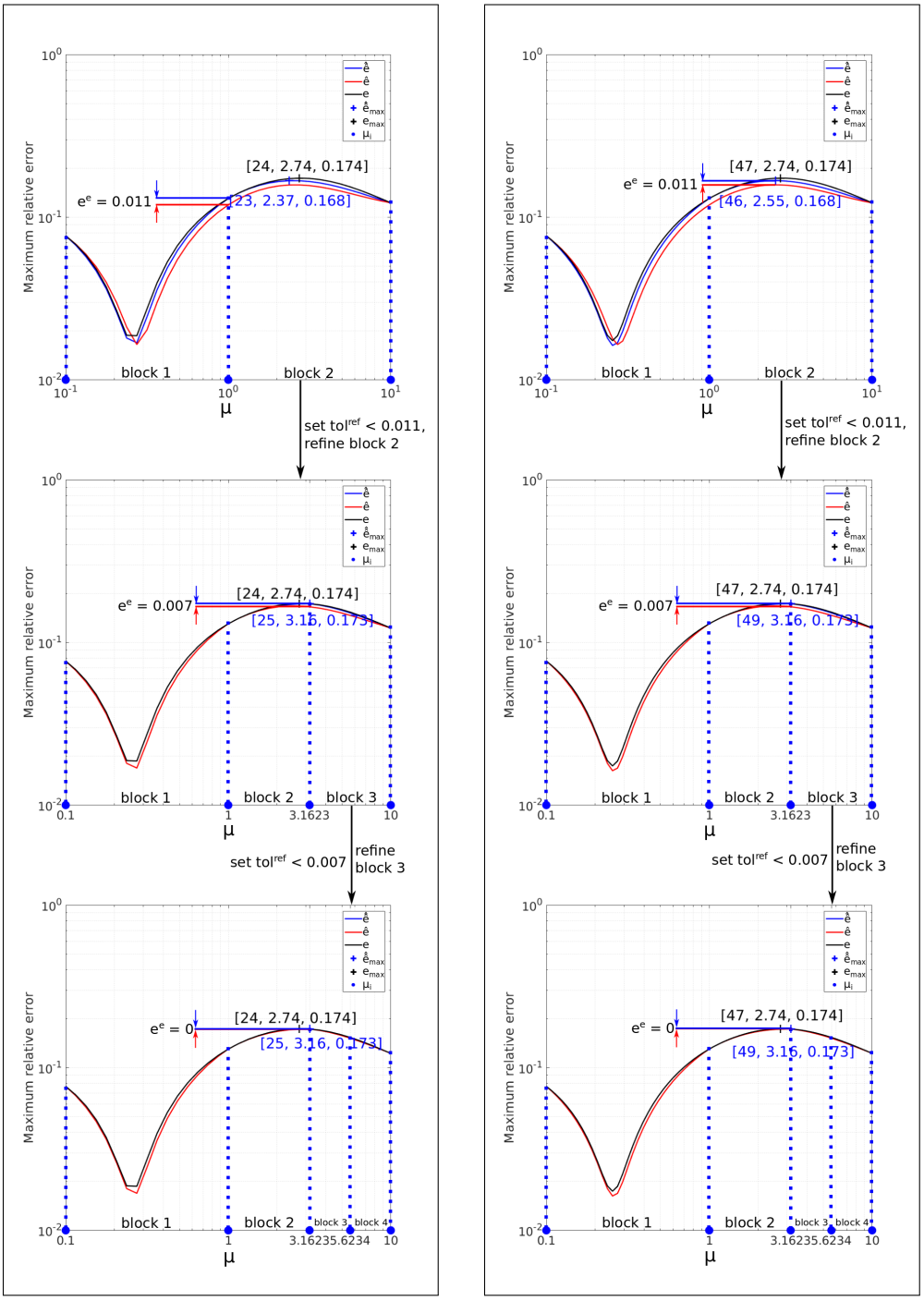
- the "*error in the error*" *indicator* $\boldsymbol{e}^e$ works as an effective guidance to the local $h-$ refinement. By loosening or restricting the user-defined tolerance tol^{ref}, $\boldsymbol{e}^e$ decides when to perform the refinement and which block to place the new sample.
- Before proceeding into the next Greedy iteration, $\hat{\boldsymbol{e}}$ is guaranteed to approach $\hat{\hat{\boldsymbol{e}}}$, thus $\boldsymbol{e}^e$ decreases; in rare cases, $\boldsymbol{e}^e$ increases when proceeding into new Greedy iterations due to changes in response surface shape.
- Training set size N_{train} may affect the interpolation: as N_{train} increases, the error response surface may become more complex, which indicates that more details of the response surface are required to be captured by the algorithm. As can be seen from fig. 5.3, in some cases even thorough refinement of the interpolation sample domain drops out key information and cannot lead to precise prediction of the magic point. A possible solution to this is utilising different interpolation methods.
- The "*error in the error*" *indicator* directly impacts numerical cost of the new error indicator, hence tol^{ref} needs to be carefully set to balance the cost and accuracy.



(a) $N_{\text{train}} = 9$, $\hat{\hat{\boldsymbol{e}}}$ captures the exact magic point without any refinement, thus tolerance tol^{ref} should be set loosely to avoid excessive cost.

(b) $N_{\text{train}} = 17$, Restricting tol^{ref} allows $\hat{\hat{\boldsymbol{e}}}$ to capture the exact magic point with 1 refinement.

Figure 5.2: Numerical results of applying "*error in the error*" *indicator* over training sets of $N_{\text{train}} = 9$ and $N_{\text{train}} = 17$. Information of the magic point is labelled on the surface following the form of $[n, \mu_n, \boldsymbol{e}]$, where n denotes the n^{th} sample point, μ_n is the associated parametric value, $\boldsymbol{e}$ is the maximum relative error.



(a) $N_{\text{train}} = 33$, restricting tol^{ref} leads to more and more accurate approximations of $\boldsymbol{e}$, however even with exhaustive refinement ($\boldsymbol{e}^e = 0$), $\hat{\hat{e}}$ still cannot capture the exact magic point.

(b) $N_{\text{train}} = 65$, similar phenomenon is observed as restricting the tolerance leads to relatively accurate approximation, however a slight difference between $\boldsymbol{e}$ and $\hat{\hat{e}}$ surface results in different magic points.

Figure 5.3: Error response surfaces of training sets with $N_{\text{train}} = 33$ and $N_{\text{train}} = 65$. Information of the magic point is again labelled on the surface following the form of $[n, \mu_n, \boldsymbol{e}]$.

Chapter 6

Proposed POD-Greedy Algorithm

Now that all components of the proposed POD-Greedy algorithm is established, it is necessary to summarize and present procedures and pseudo-code of the proposed adaptive POD-Greedy sampling algorithm in this chapter. To be more specific, the key components of the proposed algorithm are clarified: algorithm 2 equipped with true RB-error indicator is referred as the reference POD-Greedy algorithm which is the theoretical foundation; eq. (3.19) for the full space-time representation of Newmark method; eq. (4.12) for full representation of the approximated *exact full error vector*; eq. (4.25) and eq. (4.30) for Frobenius norm of the approximated displacement error matrix; eq. (4.33) for the approximated displacement matrix with decomposed singular vectors; eq. (4.37) for computation of the reduced *displacement vector product matrix*; plus chapter 5 for the "*error in the error*" *indicator*.

6.1 A complete computational procedure of the proposed POD-Greedy algorithm

Similar to the *reference POD-Greedy* algorithm, a decomposition is enabled for the proposed one, which contains the following parts

POD-Greedy *basis processing* stage: a series of expensive computations (**parameter-independent**):

1. generate impulses, compute impulse responses: $\boldsymbol{U}_i^{\mathrm{h}}$, $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$
2. compress $\boldsymbol{U}_i^{\mathrm{h}}$ and $\boldsymbol{U}_{ij\mathfrak{r}}^{imp}$, shift the right singular vectors to perform the Duhamel's integral
3. compute the *displacement vector product matrix* $\mathsf{M}_i^{\mathbf{trans}}$
4. compute the reduced variable vectors, obtain and decompose the reduced variable vector collection $\mathcal{W}_{\boldsymbol{\alpha}}$, multiply with associated affine coefficients
5. compress $\mathcal{W}_{\boldsymbol{\alpha}}$ to obtain the left and right singular vectors
6. perform Galerkin to obtain $\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}$
7. compute and enrich reduced basis $\boldsymbol{\Phi}$

POD-Greedy *parameter sweep* stage: exhaustively sweep parameter domain (**parameter-dependent**):

1. interpolate $\mathsf{M}_i^{\mathbf{trans},\mathrm{r}}$, multiply results with corresponding singular values and right singular vectors
2. evaluate the error norm and construct the error response surface

Error check ("*error in the error*" estimate, **parameter-independent**):

1. extract maximum error information
2. evaluate the "*error in the error*" *indicator* and compare with tolerance, if exceeds, perform $h-$refinement, otherwise perform Greedy iterations

6.2 Pseudocode

The pseudocode of the proposed *POD-Greedy* method is shown in algorithm 4:

Similar to section 2.2.4, detailed definition of different stages of a ROM approximation with proposed POD-Greedy algorithm is given as follows:

- ROM "*offline*" stage: basis construction
  - POD-Greedy *parameter sweep*: evaluate error for every single parameter value of $\mathcal{P}^{\text{train}}$, step 11-14.
  - POD-Greedy *basis processing*: step 4-10, 15-26.
    - calculate *displacement vector product matrix*: step 4-5.
    - calculate reduced *displacement vector product matrix*: step 6-10.
    - calculate "*error in the error*" *indicator*: step 15.
    - *basis processing*: step 17-23.
    - local $h-$refinement: step 25.
- ROM "*online*" stage: calculations of $\boldsymbol{U}^{\text{r}}(\boldsymbol{\mu})$ for single or multiple parameter values of interest $\boldsymbol{\mu}$.

Algorithm 4 Proposed POD-Greedy algorithm

Input: tol^{err} (User defined error tolerance), tol^{ref} ($h-$ refinement tolerance), N_k^{resp} for eq. (4.32) and N_k^{rv} for eq. (4.35),

Output: N, $\boldsymbol{\Phi}$

Choose $\mu_1 \in \mathcal{P}^{\text{train}}$, initialize $N = 1$, set $\mathcal{P}_1 = \{\mu_1\}$, $X_1 = \text{span}\{\boldsymbol{U}(\mu_1)\}$, set $\hat{\hat{\mathcal{P}}}_1^i$, $\hat{\mathcal{P}}^i{}_1$

Define $\boldsymbol{e}_{\text{gr}} := \max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$

while $\boldsymbol{e}_{\text{gr}} \geq \text{tol}^{\text{err}}$ **do**

  $\{\boldsymbol{M}_j \underline{\phi}_{\text{r}}, \boldsymbol{C}_j \underline{\phi}_{\text{r}}, \boldsymbol{K}_j \underline{\phi}_{\text{r}}\} \xrightarrow{\text{generate}} \boldsymbol{F}_{j\text{r}}^{imp} \xrightarrow{\text{Newmark}} \{\boldsymbol{U}_i^{\text{h}}, \boldsymbol{U}_{ij\text{r}}^{imp}\} \xrightarrow{\text{SVD}} \{\boldsymbol{V}_L, \Sigma, \boldsymbol{V}_R\}$ ▷ Use nonintrusive technique or source code

  $\boldsymbol{V}_R \delta_{tn} + \{\boldsymbol{V}_L, \Sigma\} \rightarrow \mathsf{M}_i^{\textbf{trans}}$ ▷ Equation (4.27)

  for $\mu \in \mathcal{P}^{\text{train}}$ **do**

    Construct $\mathcal{W}_\alpha$ ▷ Equation (4.34)

  end for

  $\mathcal{W}_\alpha \xrightarrow{\text{SVD}} \tilde{\boldsymbol{V}}_{\alpha L}, \tilde{\boldsymbol{\Sigma}}_\alpha, \tilde{\boldsymbol{V}}_{\alpha R}$

  $\mathsf{M}_i^{\textbf{trans}} \xrightarrow{\text{projection}} \mathsf{M}_i^{\textbf{trans},\text{r}}$ ▷ Equation (4.36)

  for $\mu \in \mathcal{P}^{\text{train}}$ **do**

    Interpolate $\mathsf{M}_i^{\textbf{trans},\text{r}}$ for $\hat{\hat{\mathcal{P}}}^i$ and $\hat{\mathcal{P}}^i$

    calculate $\sqrt{m_i} = \|\hat{e}(\mu_i)\|_{\text{F}} = \boldsymbol{e}_{\text{ind}}(\mu)$ ▷ Greedy *parameter sweep*

  end for

  Compute $\boldsymbol{e}^e$ ▷ "*error in the error*" *indicator*, eq. (5.1)

  if $\boldsymbol{e}^e \leq \text{tol}^{\text{ref}}$ **then**

    $\mu_{n+1} = \arg\max_{\mu \in \mathcal{P}^{\text{train}}} \boldsymbol{e}_{\text{ind}}(\mu)$ ▷ New samples (magic point)

    $\boldsymbol{e}_{\text{proj}}(\mu_{n+1}) := \boldsymbol{U}(\mu_{n+1}) - \boldsymbol{\Phi}_n \boldsymbol{\Phi}_n^T \boldsymbol{U}(\mu_{n+1})$

    $\phi_{n+N^{\text{add}}} := POD_{N^{\text{add}}}\left(\boldsymbol{e}_{\text{proj}}(\mu_{n+1})\right)$ ▷ POD

    $\boldsymbol{\Phi}_{n+N^{\text{add}}} := \boldsymbol{\Phi}_n \bigcup \phi_{n+N^{\text{add}}}$ ▷ Basis enrichment, algorithm 3

    $\mathcal{P}_{N+1} := \mathcal{P}_N \bigcup \{\mu_{n+1}\}$

    $X_{N+N^{\text{add}}} := X_N \bigoplus \phi_{n+N^{\text{add}}}$

    Set $N := N + N^{\text{add}}$

  else if $\boldsymbol{e}^e > \text{tol}^{\text{ref}}$ **then**

    Set $\hat{e} = \hat{\hat{e}}$, $\hat{\mathcal{P}}^i = \hat{\hat{\mathcal{P}}}^i$, $\hat{\hat{\mathcal{P}}}^i = \hat{\hat{\mathcal{P}}}^i \bigcup \{\mu_{i+1}\}$ ▷ h-refinement

  end if

end while

Chapter 7

Algorithm Validations

The ultimate goal of the proposed POD-Greedy algorithm is to reach similar (or preferably more rapid) maximum error convergence with the referenced one. The performance of the proposed algorithm is verified when a good convergence is obtained, that is to say, the proposed algorithm is accurate. Once the accuracy is validated, in next section the feasibility in terms of the overall execution time is validated: due to the substantial overhead, the proposed algorithm is not likely to reach a shorter runtime than the referenced one when the model is small. However, as the dimension of the model and number of parameters increases, runtime of the reference algorithm increases exponentially due to curse of dimensionality, while the proposed one increases much slower. If the proposed algorithm does not have the ability to evaluate a much larger training set with reasonable execution time, then it has no use as the *reference POD-Greedy* algorithm is a better alternative. In this chapter he following aspects are going to be proved: (i) the proposed algorithm is able to reach close convergence to the reference method, thus is accurate (section 7.1); (ii) the proposed algorithm is able to evaluate a large training set with reasonable accuracy and less runtime cost than the reference one when a large model is evaluated, thus is feasible to be used in real-life applications (section 7.2).

7.1 Accuracy of the proposed POD-Greedy algorithm

Due to the simple beam model structure, the *reference POD-Greedy* algorithm can be used as a comparison: the maximum error convergence of proposed POD-Greedy algorithm is expected to be close to the referenced one. In order to validate the accuracy of the proposed algorithm, the following test stage is designed and added to the proposed POD-Greedy algorithm: after each Greedy iteration is finished and $\mathbf{\Phi}$ is constructed, the true RB-error $\boldsymbol{e}(\mu)$ will be computed $\forall \mu \in \mathcal{P}^{\text{train}}$ and the error response surfaces will be constructed, then maximum true RB-error will be extracted to plot the convergence curve. It is emphasized that this stage is only for validation purpose, can be removed at any time, and is completely independent of the proposed POD-Greedy algorithm. The test stage requires exact solutions over the entire training set, while the proposed algorithm does not. The proposed method is tested with a simple example (the 2D fixed beam), then extended to a larger model (the 3D cantilever I beam).

7.1.1 The 2D fixed beam model

The simple fixed end beam as presented in fig. 2.9 is considered to verify the accuracy of the proposed POD-Greedy algorithm. The physical domain contains 2 regions: the matrix Ω_{M} and the inclusion Ω_{I}, where parameter value only varies in Ω_{I}. Dirichlet boundary conditions Γ_g are applied on both ends of the beam model, while a dynamic impulse force pointing to the $-y$ direction is applied to the midpoint of the upper edge. Density is set to be fixed to 0.01. The force is defined by a Gaussian function. For the contacted surfaces between inclusion and matrix, it is assumes that the coinciding nodes are rigidly connected. The output quantity of interest is defined as the displacement of the inclusion. The parameter is set to be $\mu \in [10^{-1}, 10^{1}]$. μ is set to be 1 for the matrix, thus the inclusion models a soft-to-hard region in the fixed beam model. The choice of μ_1 has direct impact on the maximum error convergence, therefore different μ_1 values should be tested. For this purpose, $10^{-1}, 10^{0}, 10^{1}$ are chosen to be the testing values for

μ_1. For the sake of simplicity, it is assumed that there is no damping, i.e. the Rayleigh parameters $a = b = 0$.

The *weak nonintrusive technique* is utilised in this experiment: the model geometry is constructed in Abaqus 6.14. The multi-dof dynamic model is meshed in Abaqus with 3-node linear elements, results in 514 nodes and 920 elements. The inclusion consists of 121 nodes and 200 elements. The affined system matrices (mass, damping and stiffness matrices) are obtained by importing the Abaqus .MTX files, then affine parameterisation is performed to obtain the parametric system matrices. However, in order to accelerate the process, not Abaqus but rather Matlab Newmark source code is used to obtain the pre-computed solutions for the simple beam model (see appendix A.2 for the *weak nonintrusive technique*).

7.1.2 Experiment 1: fix basis enrichment

The maximum error convergence of the proposed POD-Greedy algorithm is compared with the reference based on the single inclusion beam model. In this experiment, total time is set to be $T = 4.9$s, time step $\Delta t = 0.1$s, results in a total number of $N_t = 50$ time steps. If let $N^{\text{init}} = 10$ for the initial iteration and $N^{\text{add}} = 4$ for the successive iterations, the total number of Greedy iterations $N_G = 11$, results in $N^{\text{tot}} = 50$ basis vectors after all computations. The "*error in the error*" tolerance is set loosely to 0.25 to prevent the sample domain from being over-refined. Parameter μ is moderately discretised into $N_{\text{train}} = 129$ samples. The reason for choosing numbers 129 is that the interpolation samples are set to match the discretised training points, rather than falling between 2 values. The projection error $\boldsymbol{e}_{\text{proj}}(\mu_n)$ is compressed by POD to be used in the basis enrichment. A standard *Gram-Schmidt* process is applied to ensure the basis is orthonormalised.

In order to evaluate the performance of the proposed POD-Greedy algorithm, the error convergence is tested for the following 2 cases: (i) proposed POD-Greedy vs proposed POD-Greedy with initial refinement vs reference POD-Greedy (benchmark); (ii)

proposed POD-Greedy vs other sampling approaches (pseudorandom, uniform systematic (uniform structural grid), quasi-random, Latin Hypercube). An initial refinement of the interpolation samples might be performed to accelerate the convergence. In case (ii), the experiment is repeated 5 times for random type approaches (pseudorandom, Latin Hypercube) to ensure generality. The settings of other sampling methods in this example are adapted from section 2.2.4. Another parameter which impacts the performance of the reduced basis is the choice of initial magic point. The initial magic point might be selected randomly but in this experiment 3 choices are tested: $[10^{-1}, 10^{1}, 10^{0}]$, which represent the left and right boundaries and the central point of the parametric domain, respectively. Choosing these 3 test cases also ensures generality of the experiment.

Numerical results

Figure 7.1 reports the maximum error convergence and corresponding sample locations for reference and proposed POD-Greedy algorithms. As expected, convergence of the proposed POD-Greedy algorithm is generally slower but stays close to the referenced one. It can be seen from fig. 7.1 that (a): maximum error convergences of reference and proposed POD-Greedy algorithm are equally good until $N = 38$, then convergence of the proposed method reaches the bottleneck and becomes slow; the initial refinement contributes to the method as the sample locations are close to the referenced one, thus its convergence follows the blue curve more closely. However, again the convergence reaches the bottleneck after $N = 38$. The magic point location gives a possible reason for such bottleneck: after $N = 38$, the proposed algorithm keeps on capturing the same magic point for the successive 4 Greedy iterations, and reduction of the error is insufficient to capture another sample point due to the fixed enrichment of the basis size. Therefore a solution for this is fixing the reduction rate at each magic point rather than fixing the enrichment of the basis size. This will be demonstrated in section 7.1.3.

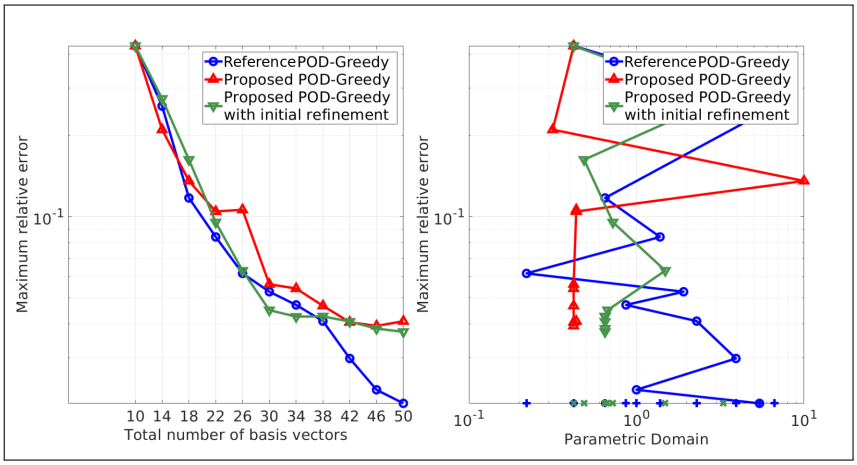
Proposed POD-Greedy algorithm performs better in fig. 7.1 (b) as it generally follows

referenced one quite closely. The initial refinement performs equally well as the reference, also the bias in $26 \leq N \leq 42$ is fixed. Proposed POD-Greedy algorithm shows its power in fig. 7.1 (c) as it reaches the same convergence as the reference until $N = 42$. Then it converges slightly slower.

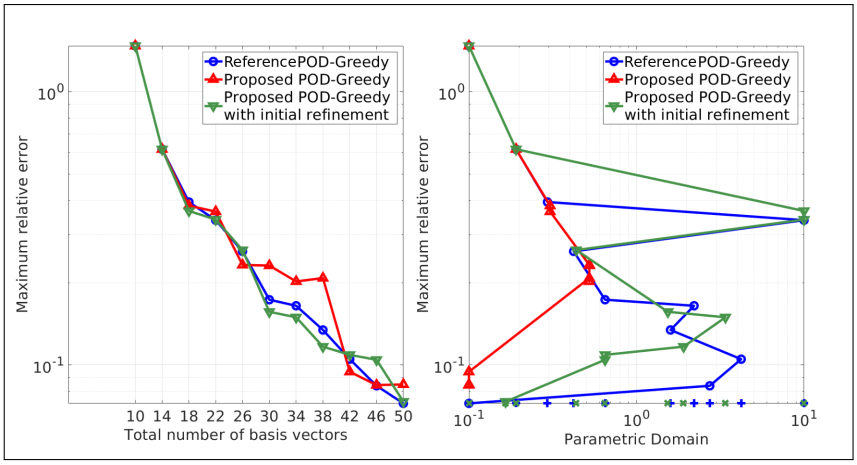
Next the comparison between proposed POD-Greedy algorithm and other statically based sampling approaches are being investigated. Figure 7.2 (a) shows that even the proposed POD-Greedy suffers the bottleneck in $N > 38$, it still converges more rapidly than pseudorandom, systematic and Latin Hypercube sampling. Quasi-random sampling reaches a lower maximum error at $N = 50$ but prior to that it converges slower than the proposed algorithm too. It can be seen from fig. 7.2 (b) and (c) that proposed POD-Greedy algorithm gives equally good reduced model as referenced one and surpasses all other sampling approaches at $N = 50$. Finally it can be seen that *reference POD-Greedy* converges faster and reaches lower maximum error for all cases, which is intuitively expected. Therefore it can be concluded that maximum error convergence reference POD-Greedy $>$ proposed POD-Greedy $>$ statically sampling approaches, and overall performance of the proposed POD-Greedy algorithm is relatively satisfying.

7.1.3 Experiment 2: fix error reduction tolerance

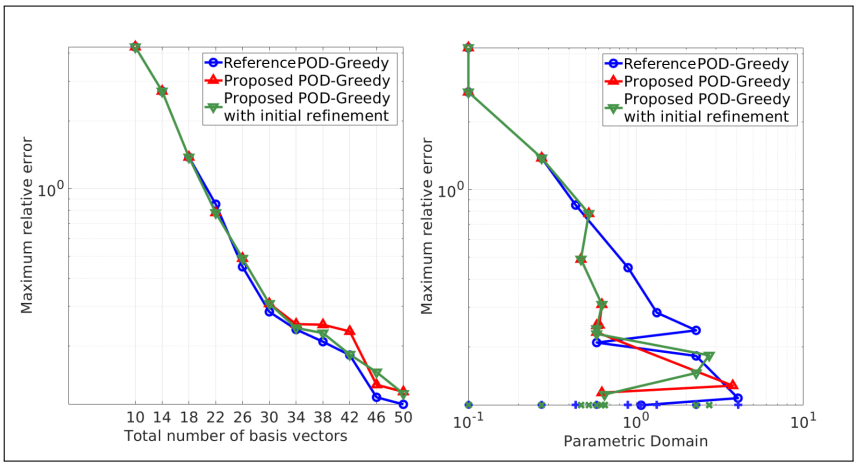
The general setting of experiment 2 is adapted from experiment 1, except that this time not a fixed number of N^{init} and N^{add} are chosen. Instead, algorithm 3 is applied at step 18 of algorithm 4 and the reduction tolerance is set to $e_{\text{r}}^{\text{to}} = 0.8$. More specifically, once a magic point is located and the associated snapshot is obtained, algorithm 3 is applied on the snapshot until the reduction tolerance is reached. The output of algorithm 3 is N^{add}, i.e. number of enriched basis vectors. Statistically based sampling approaches are shown as comparisons.



(a) $\mu_1 = 10^{-1}$

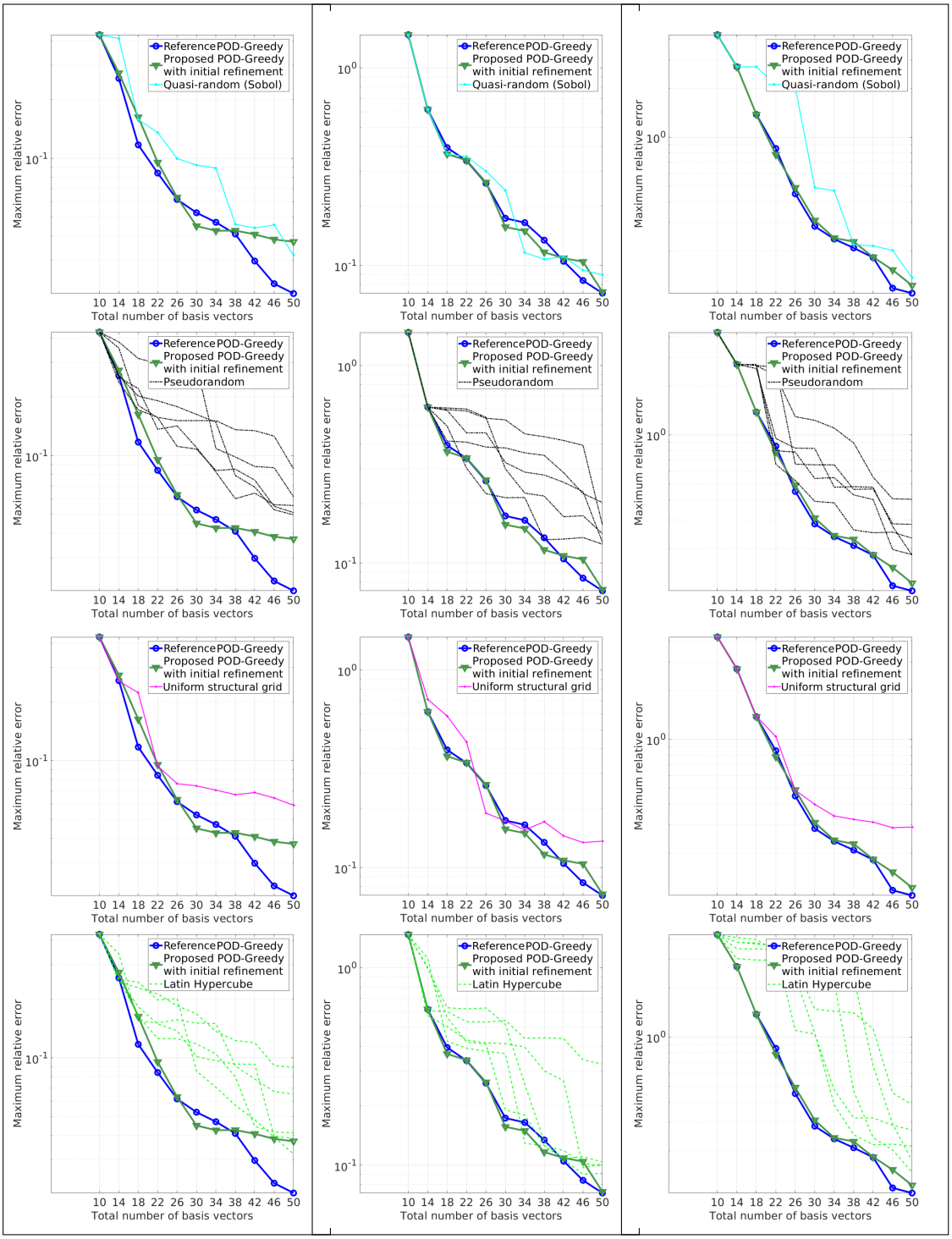


(b) $\mu_1 = 10^{0}$



(c) $\mu_1 = 10^{1}$

Figure 7.1: Compare the maximum error convergence (left) and corresponding magic point locations (right) for experiment 1. Convergence of reference and proposed methods are overall relatively close, except $\mu_1 = 10^{-1}$after 38 basis vectors.



(a) Initial magic point $\mu = 10^{-1}$ (b) Initial magic point $\mu = 10^{0}$ (c) Initial magic point $\mu = 10^{1}$

Figure 7.2: Compare the maximum error convergence between proposed POD-Greedy algorithm and other sampling methods for experiment 1. Proposed POD-Greedy algorithm converges more rapidly in most cases. Sobol sequence is a strong component as it converges fast in all 3 tests. This will be further investigated with the I beam model in experiment 2.

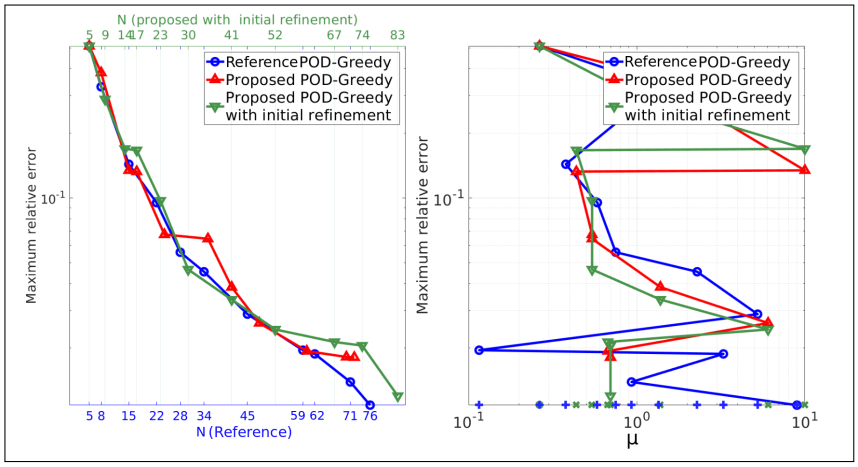
Numerical results

The maximum error convergence and corresponding sample locations are shown in fig. 7.3 and fig. 7.4, respectively. Due to the introduction of the reduction tolerance, this time the number of newly added basis vectors is no longer fixed. In fig. 7.3 (a), the proposed POD-Greedy algorithm reaches slightly larger maximum relative error with $N = 83$, while $N = 76$ basis vectors are generated with reference. Compared to the results in fig. 7.1 (a), this time the propose algorithm is able to capture non-repeated magic points and the aforementioned bottleneck is fixed which results in a better error reduction. Figure 7.3 (b) shows that convergence of the proposed algorithm follows relatively closely with *reference POD-Greedy* algorithm until $N = 70$. Again similar to fig. 7.1 (c), a very good match between reference and proposed POD-Greedy algorithm is shown in fig. 7.3 (c). The proposed one reaches slightly smaller maximum error with 2 more basis vectors.

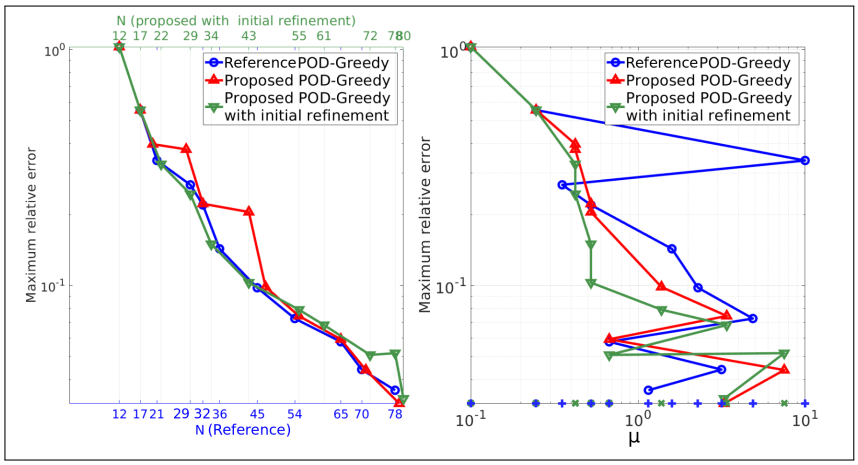
Figure 7.4 shows that proposed POD-Greedy algorithm converges more rapidly and reaches smaller maximum relative error with more compact bases than pseudorandom, systematic and Latin Hypercube sampling. An exception is quasi-random sampling: it shows a very close convergence to the reference and performs better than proposed algorithm when $N > 50$.

Speed-up

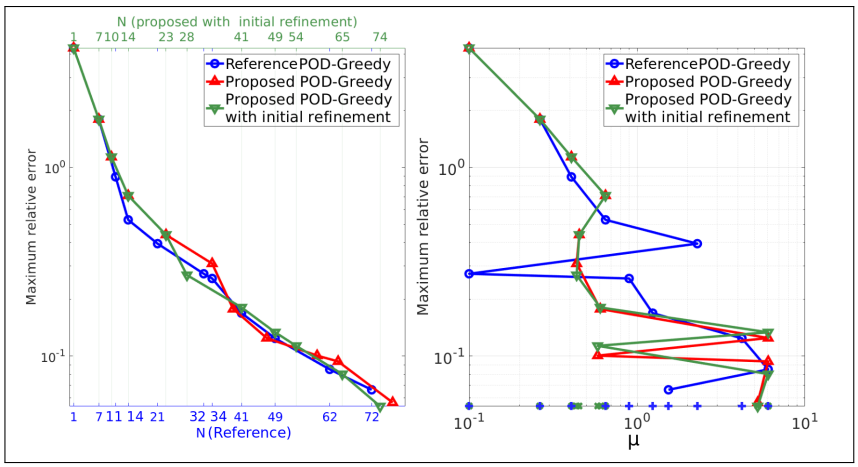
Now speed-up of the ROM is studied. The speed-up is defined as the ratio of the execution time of the ROM over the full order simulation of Matlab Newmark source code and Abaqus. More specifically, the ROM approximation is $\boldsymbol{U}^{\mathrm{r}}(\mu) = \boldsymbol{\Phi}\boldsymbol{\alpha}(\mu)$, while the exact solution is either obtained by solving eq. (1.6) and eq. (1.7) in a step-by-step manner (Matlab Newmark source code), or performing a full Abaqus simulation. Notice that for the Abaqus simulation, GUI is not utilised, instead Matlab is used to call



(a) Initial magic point $\mu = 10^{-1}$

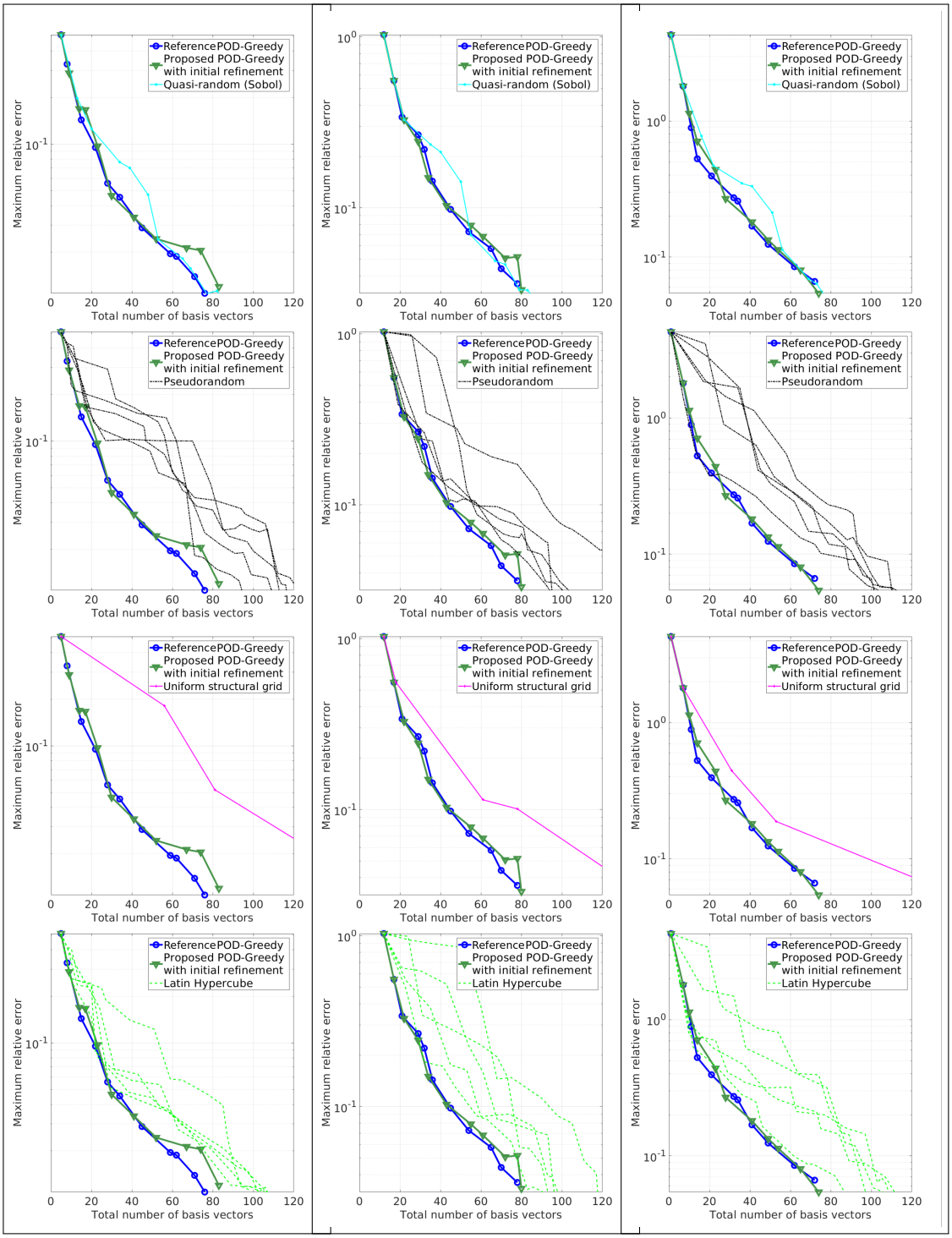


(b) Initial magic point $\mu = 10^{0}$



(c) Initial magic point $\mu = 10^{1}$

Figure 7.3: The maximum error convergence (left) and corresponding magic point locations (right) for experiment 2. Again close convergence are observed between reference and proposed methods. Compare with fig. 7.1, convergence of proposed method with initial refinement improves as a result of fixed error reduction at magic points.



(a) $\mu_1 = 10^{-1}$ (b) $\mu_1 = 10^{0}$ (c) $\mu_1 = 10^{1}$

Figure 7.4: Compare the maximum error convergence between proposed POD-Greedy algorithm and statistically based sampling methods for experiment 2. Proposed POD-Greedy algorithm converges more rapidly than most other cases.

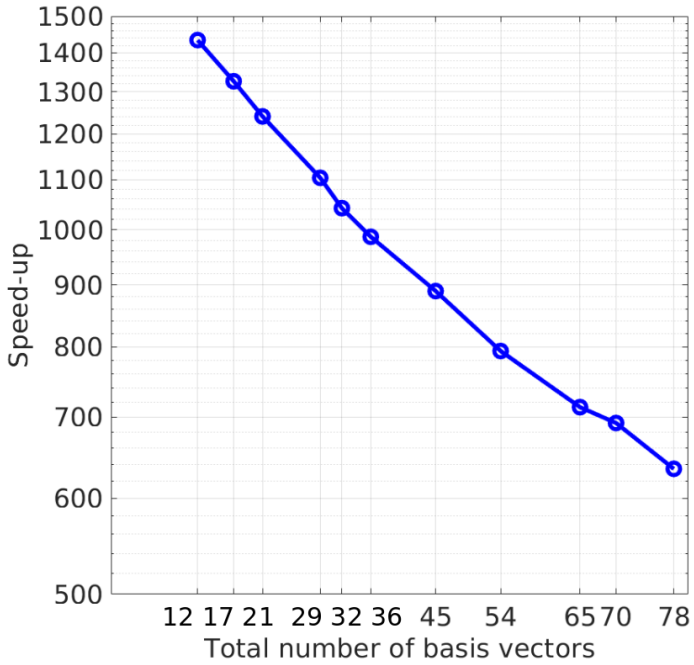


Figure 7.5: Evolution of the speed-up with respect to the number of basis vectors.

Abaqus via the .inp file such that only purely simulation time is being considered. See appendix A for more details. The result can be intuitive: as the number of reduced basis vectors increases, the speed-up and error decrease in a roughly linear manner.

7.1.4 Summary

In this numerical example, the proposed POD-Greedy algorithm gives a satisfying performance in the following aspects: compare with the optimum benchmark *reference POD-Greedy* algorithm, the proposed method 'chases' the reference one quite closely with respect to maximum error convergence; compare with other statistically based sampling methods, the proposed algorithm converges more rapidly than most of them. It can be

noticed that a strong opponent is the quasi-random sampling (Sobol sequence), for some reason it performs better than propose POD-Greedy algorithm in many test cases, which requires future research.

7.2 Feasibility of the proposed POD-Greedy algorithm

The accuracy of the proposed POD-Greedy algorithm has been verified in the previous section. Numerical results show that the proposed algorithm reaches a rapid maximum relative error convergence, that is to say, convergence of the proposed method approaches the *reference POD-Greedy* algorithm. However, in section 7.1, when dealing with a simple model such as the 2D fixed beam, the numerical cost of the proposed algorithm exceeds the reference one, hence is not feasible to be used in real-life applications. The POD-Greedy algorithm is divided into a *basis processing* stage and a *parameter sweep* stage. The *basis processing* stage costs of reference and proposed POD-Greedy algorithm are named as C_{bg}^{st} and C_{bg}^{pr}, and *parameter sweep* stage costs of reference and proposed POD-Greedy algorithm as C_{ps}^{st} and C_{ps}^{pr}, respectively. If comparing algorithm 2 with algorithm 4, it can be seen that under certain circumstances:

- *basis processing* stage of algorithm 2 (step 6-12) may cost less than that of algorithm 4 (step 4-10, 15-26), i.e. $C_{\text{bg}}^{st} < C_{\text{bg}}^{pr}$
- *parameter sweep* stage of algorithm 2 (step 4-5) may cost more than that of algorithm 4 (step 11-14), i.e. $C_{\text{ps}}^{st} > C_{\text{ps}}^{pr}$.

C_{ps}^{st} dominates the numerical cost of algorithm 2 due to iterative computation of exact solutions. C_{ps}^{st} grows exponentially with dimensional increase of parameters. In contrast, algorithm 4 has a smaller N_{exact}, i.e. requires less computation of exact solutions (main cost is C_{bg}^{pr} in step 4), in fact C_{bg}^{pr} is a function of N_i, N and N_j.

Therefore the following prediction is made and proved in this numerical experiment: when dealing with a large model and a large training set, $C^{st}_{\text{bg}} + C^{st}_{\text{ps}} > C^{pr}_{\text{bg}} + C^{pr}_{\text{ps}}$, i.e. the proposed POD-Greedy algorithm is more efficient than the reference in terms of execution time, thus is feasible to be utilised in real-life applications.

7.2.1 The 3D I beam model

In order to prove the above predictions, a new model is required. Compared with the 2D fixed beam model, this new one should possess larger number of DoFs and time steps, and possibly more parameters. Moreover for each parameter, the domain needs to be further discretised to create a larger training set. However in this case the large training set may not be able to be evaluated by the reference POD-Greedy algorithm due to prohibitive computational cost, thus a coarse grid is drawn from the large one to work as a validation training set. Hence the 3D I beam model is proposed as follows:

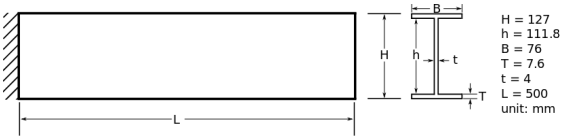
Geometry of the I beam model is chosen from British Steel Universal Beam Sections (see [1], serial size 127×76×13) as an example problem to verify feasibility of the proposed POD-Greedy algorithm. The I beam consists of 2 flanges and 1 web. One end of the beam is fully fixed, results in a cantilever beam model. A dynamic force is applied on the other end pointing to the $-y$ direction, which is defined as a Gaussian function as shown in fig. 7.6b. For time integration, the total time is set to $T = 99$s and time step length is set to $\Delta t = 1$s, result in 100 time steps. Rayleigh damping is applied to the model, where the damping matrix is obtained by affine expansion: $\boldsymbol{C}_j := b\boldsymbol{K}_j$, $\boldsymbol{K}_j$ is the j^{th} affine stiffness operator. Density of the I beam is fixed to 0.01. Parameters of the web are the varying parameters μ in this case, i.e. the parameter values of flanges remain unchanged. The parameter domain $\mathcal{P} := [10^{-1}, 10^{1}] \times [10^{-1}, 10^{1}]$. Both parameters are fixed to 1 in the flanges. The spacial quantity of interest is defined as the displacement at the beam end, and time quantity of interest is step $45 - 55$, i.e. $\boldsymbol{T}_{qoi} = [45, 55]s$.

Again the *weak nonintrusive technique* is utilised in this experiment: the model geom-

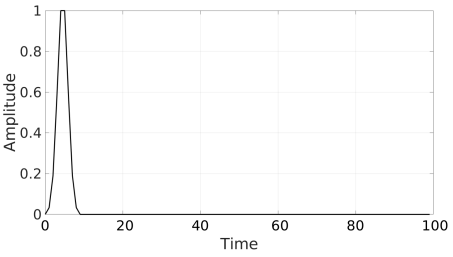
etry is constructed in Abaqus 6.14. The multi-dof dynamic model is moderately meshed in Abaqus with 4-node tetrahedral elements (C3D4 element provided by Abaqus), results in 1978 nodes and 5909 elements. Number of DoF $\mathcal{N} = 5934$. The web consists of 837 nodes and 2417 elements. The affined system matrices (mass, damping and stiffness matrices) are obtained by importing the Abaqus .MTX files, then affine parameterisation is performed to obtain the parameter-dependent system matrices. However, in order to accelerate the process, not Abaqus but rather Matlab Newmark source code is used to obtain the exact solutions for the simple beam model (see appendix A.2 for the *weak nonintrusive technique*).

For proposed POD-Greedy algorithm, the parameter domain $\mathcal{P}$ is discretised into a uniform grid (the training set) $\mathcal{P}^{\text{train}}$ with $\{\mu_i, b_j\}_{i,j=1}^{65,65}$, hence consists of $N_{\text{train}} = 65 \times 65 = 4225$ samples. This parametric domain discretisation is chosen to show that the proposed POD-Greedy algorithm is capable of evaluating a very large training set. This large training set can no longer be evaluated in the *reference POD-Greedy* algorithm due to substantial computational time, hence a coarse training set $\mathcal{P}_{co}^{\text{train}} \subset \mathcal{P}^{\text{train}}$ with $N_{\text{train}} = 9 \times 9$ samples is extracted to be used as a validation set. Both reference and proposed methods utilise $\mathcal{P}_{co}^{\text{train}}$ to compare the maximum relative error convergence and verify the accuracy of the latter method. For testing purpose, a fixed basis enrichment is set as follows: $N^{\text{init}} = 2$ and $N^{\text{add}} = 2$, $N_G = 20$ greedy iterations are performed, result in $N = 40$ basis vectors in total. To ensure generality, 2 cases are tested by choosing different initial magic point: $[\mu, b] = [10^{-1}, 10^{-1}]$ and $[\mu, b] = [10^{1}, 10^{1}]$ (one at a time), i.e. the first and last sample of $\mathcal{P}^{\text{train}}$ (see fig. 7.7).

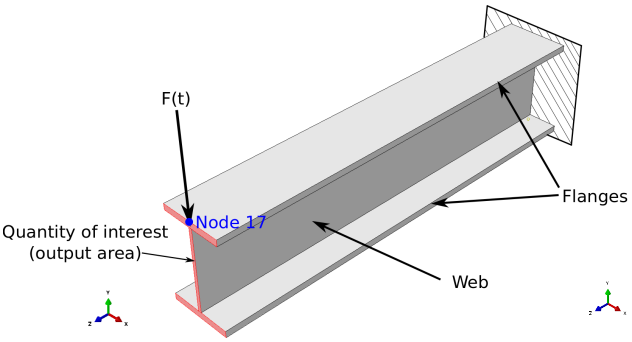
Experiment settings For convergence comparison, the experiment is performed as follows: first the proposed method with $\mathcal{P}^{\text{train}}$ (4225 samples) is performed for 20 Greedy iterations, once it's finished the reduced basis $\mathbf{\Phi}_{pro}$ and magic point set $\mathcal{P}_{pro}^{\mathcal{M}}$ are collected. Then reference method with $\mathcal{P}_{co}^{\text{train}}$ is performed to obtain its maximum relative error convergence (see red curves in fig. 7.10). In order to evaluate performance of the output



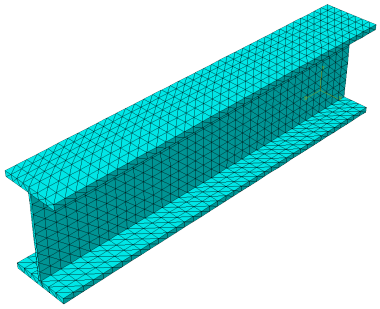
(a) Geometry



(b) Force



(c) Quantity of interest and boundary conditions



(d) Mesh generated in Abaqus

Figure 7.6: The 3D I beam model(using geometry from British Steel). The I beam consists of two flanges and one web. The model is meshed in Abaqus, results in 1978 nodes and 5909 tetrahedral elements. A dynamic force is applied at the centre node (node 17) of top of the free end. The output of interest is the displacement of the free end section.

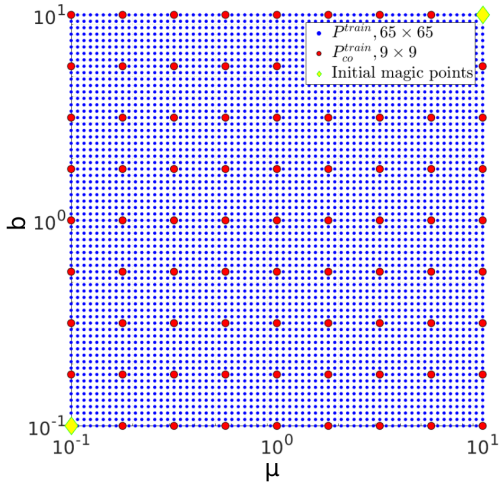


Figure 7.7: The fine training set $\mathcal{P}^{\text{train}}$ being evaluated in proposed method and prediction, and the coarse training set $\mathcal{P}_{co}^{\text{train}}$ being evaluated in the reference POD-Greedy algorithm. Two corner vertices (yellow diamonds) are chosen as test cases to verify accuracy of the proposed POD-Greed algorithm.

reduced basis $\mathbf{\Phi}_{pro}$, its vectors are utilised to construct 20 error response surfaces in a Greedy manner and find corresponding maximum error convergence (blue curves in fig. 7.10). This output convergence is compared with the red curve. Finally statically based sampling approaches are performed to compute error convergence with similar settings, except this time $\mathcal{P}^{M}$ is generated *a priori*.

7.2.2 Numerical results

Execution time First the numerical cost between the reference and proposed methods is compared in fig. 7.8, which is measured by the execution time obtained from Matlab profile function. According to Matlab profiler output, 1620 exact solutions are solved in reference POD-Greedy algorithm which requires 4396.699 seconds, while total execution time is T_{ref} = 4475.864 seconds.

Now execution time of the two test cases for the proposed method is investigated. For case 1 ($[\mu, b] = [10^{-1}, 10^{-1}]$): 2880 exact solutions are solved which costs 3628.533 seconds, while total time cost is $T_{\text{pro}} = 67136.947$ seconds. Number of exact solutions fits our calculation as number of pre-computed responses required in proposed method equals to $N_{\text{exact}} = N_i \times N \times N_j \times 2 = 9 \times 40 \times 4 \times 2 = 2880$. One may notice that it takes less time for the proposed method to produce same number of exact solutions as reference method. The reason is that proposed method allows us to evaluate only max time quantity of interest, which is 55 time steps in this case; while residual-based reference method requires to evaluate the entire 100 time steps. For case 2 ($[\mu, b] = [10^{1}, 10^{1}]$): $h-$refinement are performed twice to reduce the "*error in the error*" *indicator* below the pre-set tolerance, results in 19 interpolation samples. Consequently, the computation becomes more sophisticated and the cost increases. Number of exact solutions $N_{\text{exact}} = N_i \times N \times N_j \times 2 = 19 \times 40 \times 4 \times 2 = 6080$, costs 7504.856 seconds to evaluate. Total execution time is 76063.020 seconds for case 2. The $h-$refinement results in accurate prediction of magic points for Greedy iterations 2-5 (first magic point is manually set to be the same), which is reflected in the maximum error convergence, see fig. 7.10b.

Although the fine sample domain (65×65) cannot be evaluated with the reference method, one may still predict its execution time (see prediction 2 of fig. 7.8): number of exact solutions require to be computed equals to $N_{\text{exact}} = 65 \times 65 \times 20 = 84500$, therefore POD-Greedy *parameter sweep* stage would cost $T_{\text{pre}} = \frac{4396.699}{1620} \times 84500 = 229333.991$ seconds, which is roughly 3.416 times more than test case 1 and 3.015 times more than test case 2. From algorithm 2 it can be seen that increasing parametric domain size leads to trivial increase of *basis processing* cost, thus here it is assumed the prediction possesses similar *basis processing* execution time as the test. It is also interesting to see if given same execution time as test 2, what domain discretisation would be achieved by reference method. Result shows that only a 38×38 grid might be evaluated, which is much smaller than $\mathcal{P}^{\text{train}}$ in test 2 (see prediction 1 of fig. 7.8).

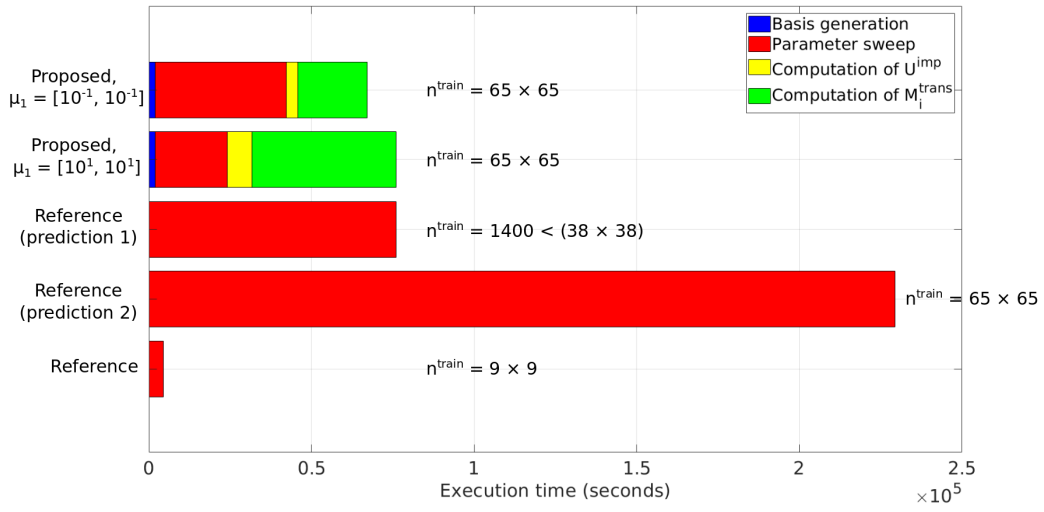
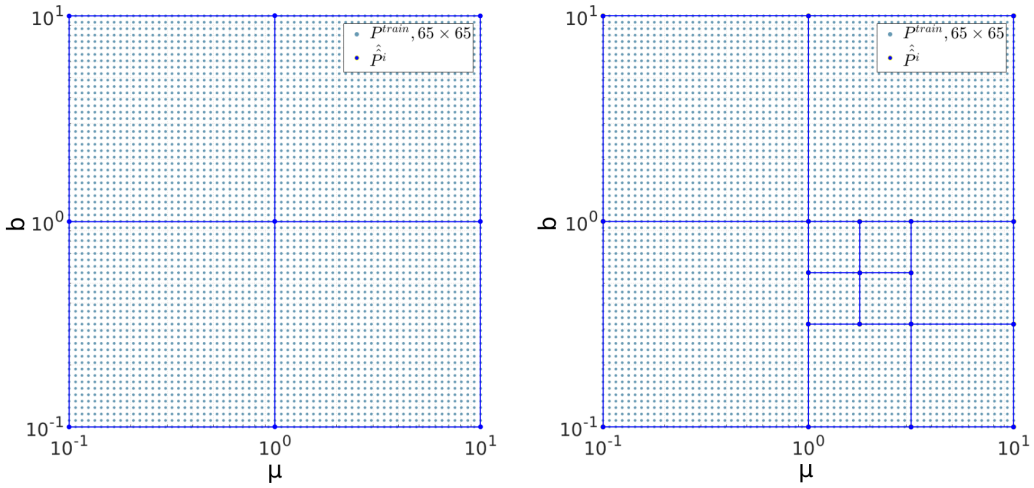


Figure 7.8: Schematic representation of POD-Greedy methods time cost. Prediction 1 shows that with the same execution time, reference method is only capable of evaluating a 38×38 grid, while the proposed method evaluates a 65×65 grid. Prediction 2 shows that if evaluating a 65×65 grid, reference method would cost 3.416 times more than test case 1. Cost of the extra 2 major components in proposed method is highlighted. In the reference method, when dealing with a large training set, *parameter sweep* requires a enormous number of exact solutions being solved, which is the main cost of the entire method. The proposed method achieves a major reduction of the *parameter sweep*, which allows the users to evaluate large training sets. As a trade-off, it requires to compute a number of exact solutions (impulse responses $\boldsymbol{U}^{imp}$) and the *displacement vector product matrix* $\mathsf{M}_i^{\mathbf{trans}}$. However even with the trade-off, proposed method still achieves a much less expensive cost than the reference one when the same large training set is evaluated.

Accuracy Accuracy of the proposed algorithm are again verified in this test: compare maximum relative error convergences of reference with proposed POD-Greedy algorithm over the coarse training set $\mathcal{P}_{co}^{\text{train}}$ for 20 Greedy iterations, and statistically based sampling approaches (pseudorandom sampling, Halton sequence, Sobol sequence, Latin hypercube sampling) are shown alongside. Selection of the statistically based samples follows the same principle as discussed in section 2.2.4. Convergence results are presented in fig. 7.10, which show good agreement between reference and proposed methods. Both results show that proposed algorithm follows closely with the reference method, while other methods converges much slower. By observing magic point distributions in fig. 7.11, it can be seen that for reference and proposed methods, all points fall in $b = 10^{-1}$, i.e. the low damp region. This fits the observation in section 2.2.5. Results of many research show that the magic point distribution is not necessarily uniform in the parameter domain. For example, in [51, 53], Young's modulus and damping coefficient are parameters and output magic points are distributed in low damp regions; in [50], the diffusion constant of subdomains are the parameters and magic points distribute along domain borders; [75] perform Greedy procedure with a 3-parameter example problem and the output magic points show a relatively uniform distribution. Based on existing research and our experiment results, it can be seen that simply using statistically based sampling methods does not predict the magic point distribution accurately in many cases, which is likely to result in a slow maximum error convergence.

The first 10 dynamic modes (reduced basis vectors) generated by the first 5 Greedy iterations for initial magic point $[\mu, b] = [10^{-1}, 10^{-1}]$ are proposed in fig. 7.12 and fig. 7.13. For this case, reference and proposed POD-Greedy algorithm generate 2 identical modes. The y-displacement are plotted on the beam model as colour fields. Suitable scale factors are applied to enhance visibility.

Speed-up Now speed-up of the ROM is studied based on the I beam model. The speed-up is studied in 2 parts: (a) speed-up of recovering solution using reduced order model



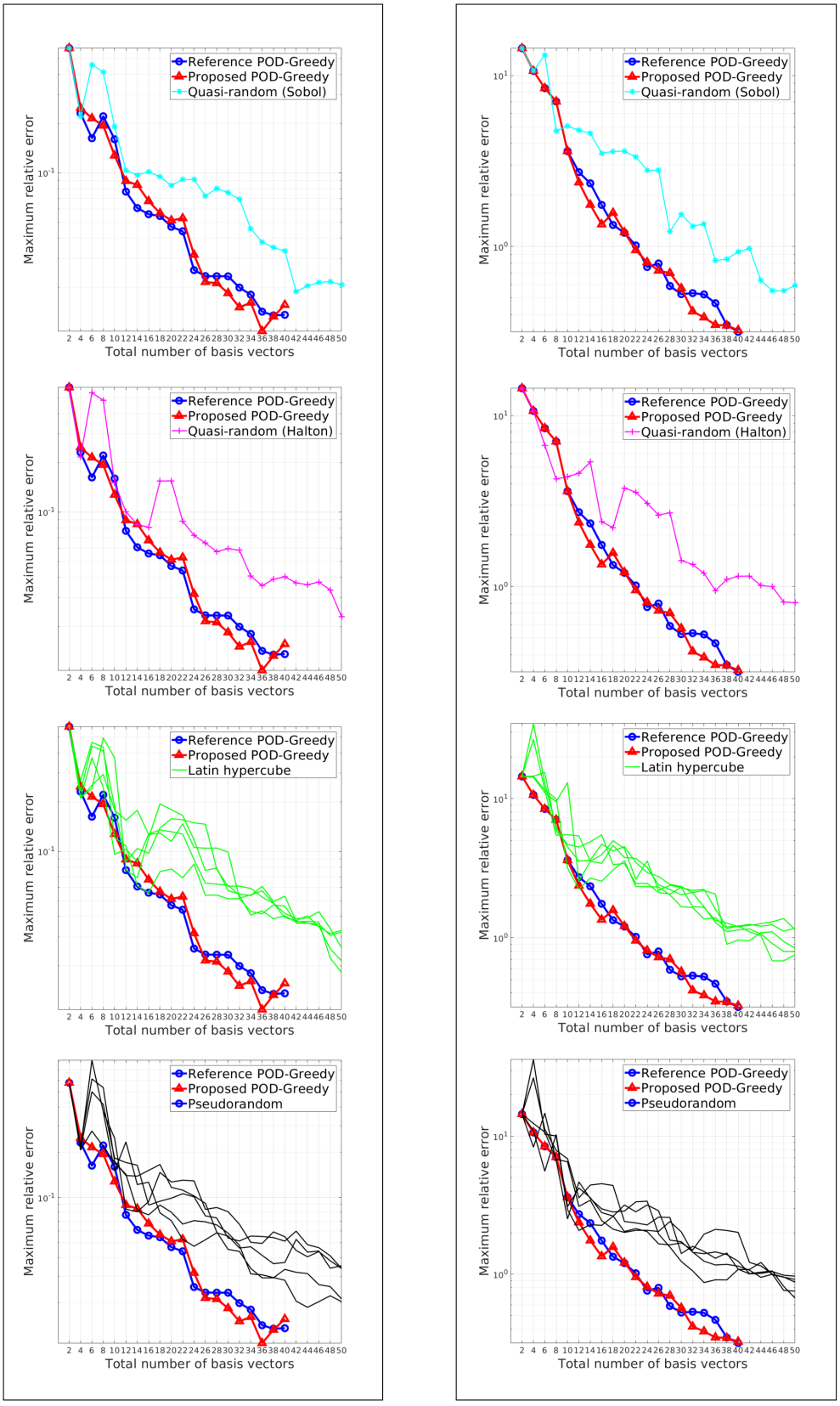
(a) $\hat{\hat{\mathcal{P}}}^i$ for test case 1.

(b) Refined $\hat{\hat{\mathcal{P}}}^i$ for test case 2.

Figure 7.9: Interpolation domains for the two test cases. Case 2 performs 2 local refinements. The grids are partitioned for better visibility.

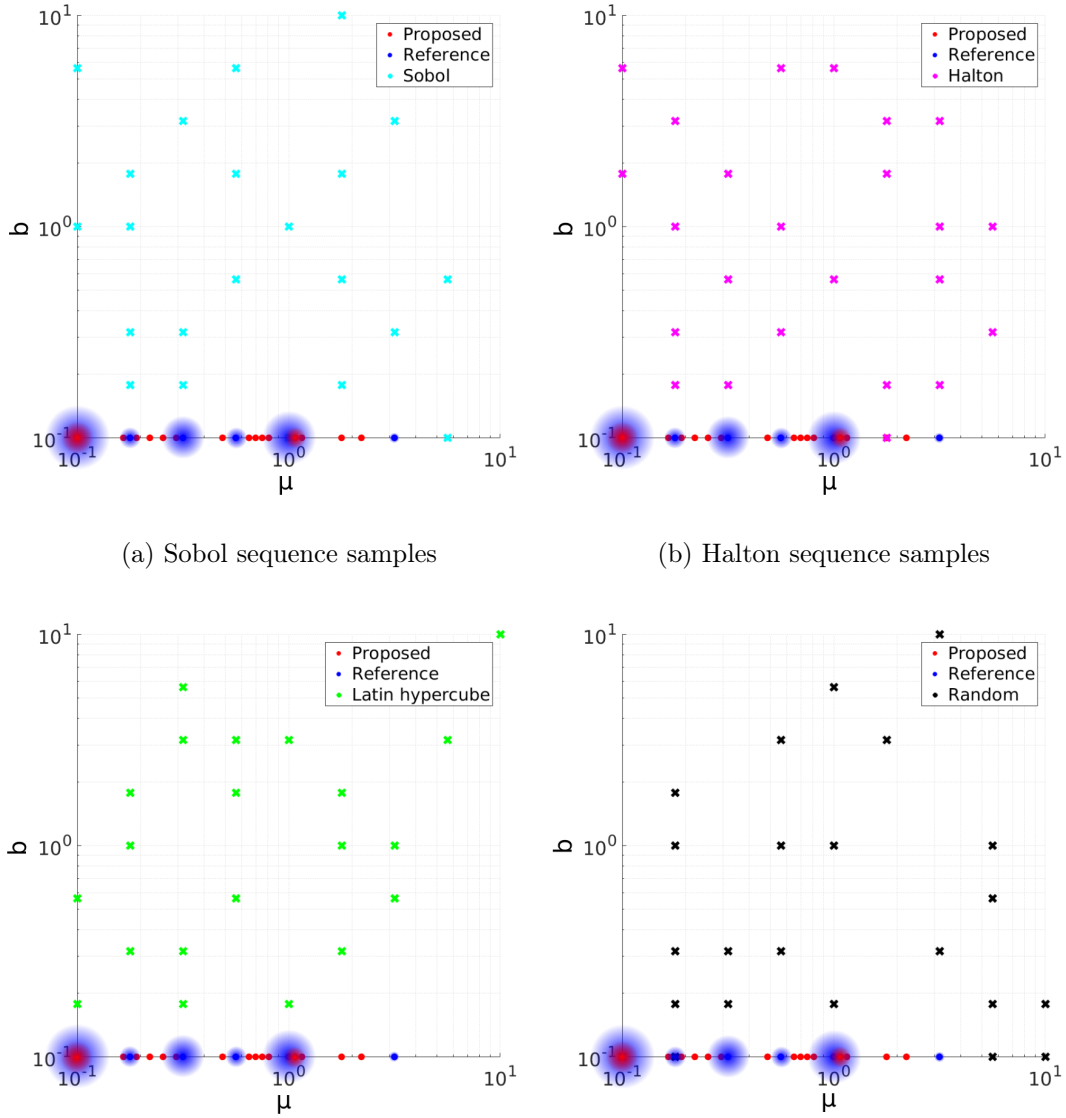
over computing full trajectory, (b) speed-up of proposed method over other sampling methods when reaching same level of error. For (a), similar setting as section 7.1.3 is applied. Again as the number of reduced basis vectors increases, the speed-up decrease in a roughly linear manner. However compare with fig. 7.5, it can be seen that the speed-up for I beam model increases while the same number of basis vectors being used. In (b), maximum 40% of speed-up is achieved.

Time-domain displacements The time-domain displacements in fig. 7.15 for different parametric values after 20 Greedy iterations are presented. The amplitude is the mean displacement of the I beam free end under dynamic load. Each plot shows exact solution (black curve), approximation constructed using reference (blue curve) and proposed POD-Greedy (red curve) output reduced bases, respectively. Results suggest good agreement between exact solution and the two approximations for all time steps, which fits our expectation as the maximum relative error drops below 2% for both approxima-



(a) Initial magic point $[\mu, b] = [10^{-1}, 10^{-1}]$ (b) Initial magic point $[\mu, b] = [10^{1}, 10^{1}]$

Figure 7.10: Compare the maximum error convergence between reference, proposed POD-Greedy algorithm and statistically based sampling methods. Reference and proposed methods show close, fast convergences which well exceed other methods.



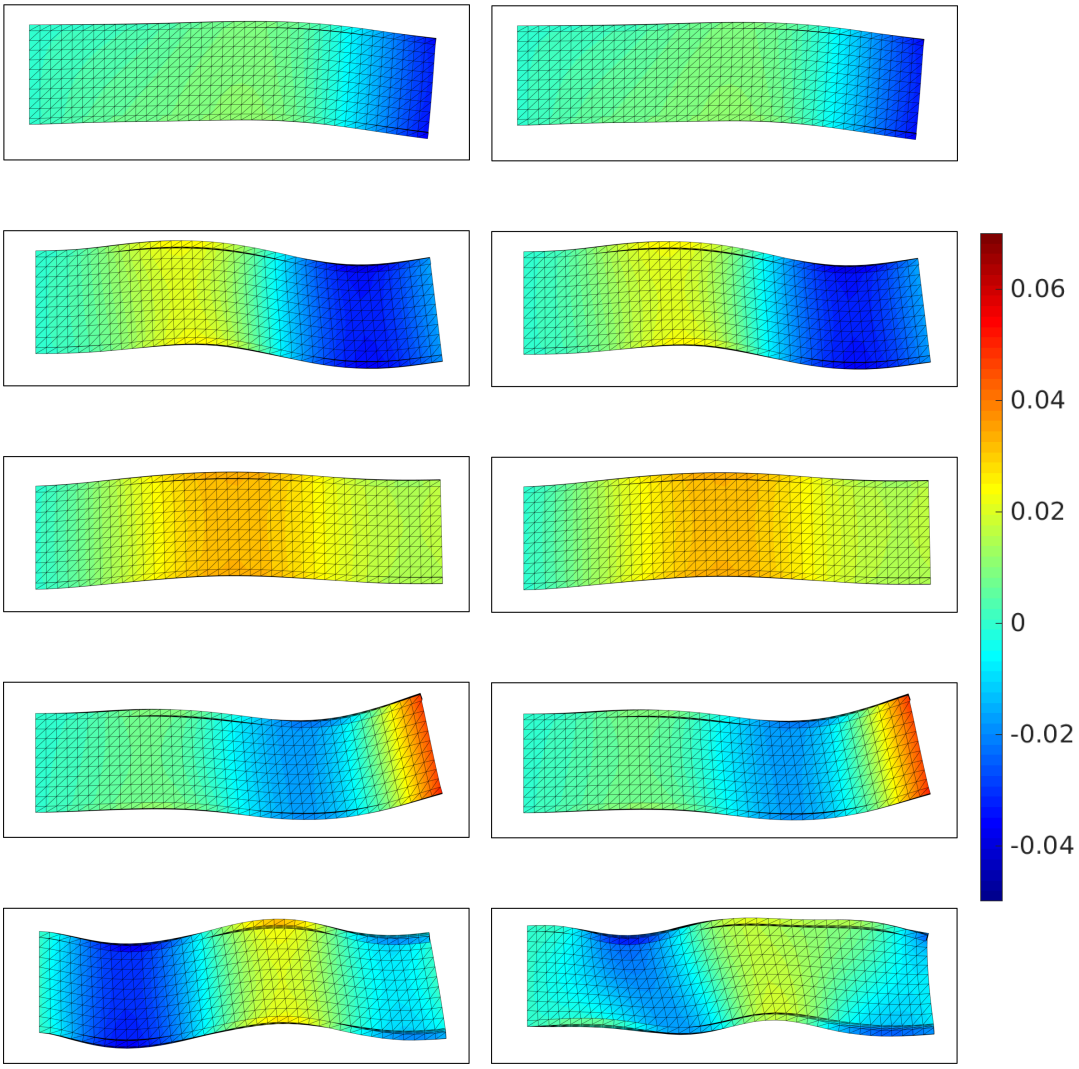
(a) Sobol sequence samples

(b) Halton sequence samples

(c) Latin hypercube samples

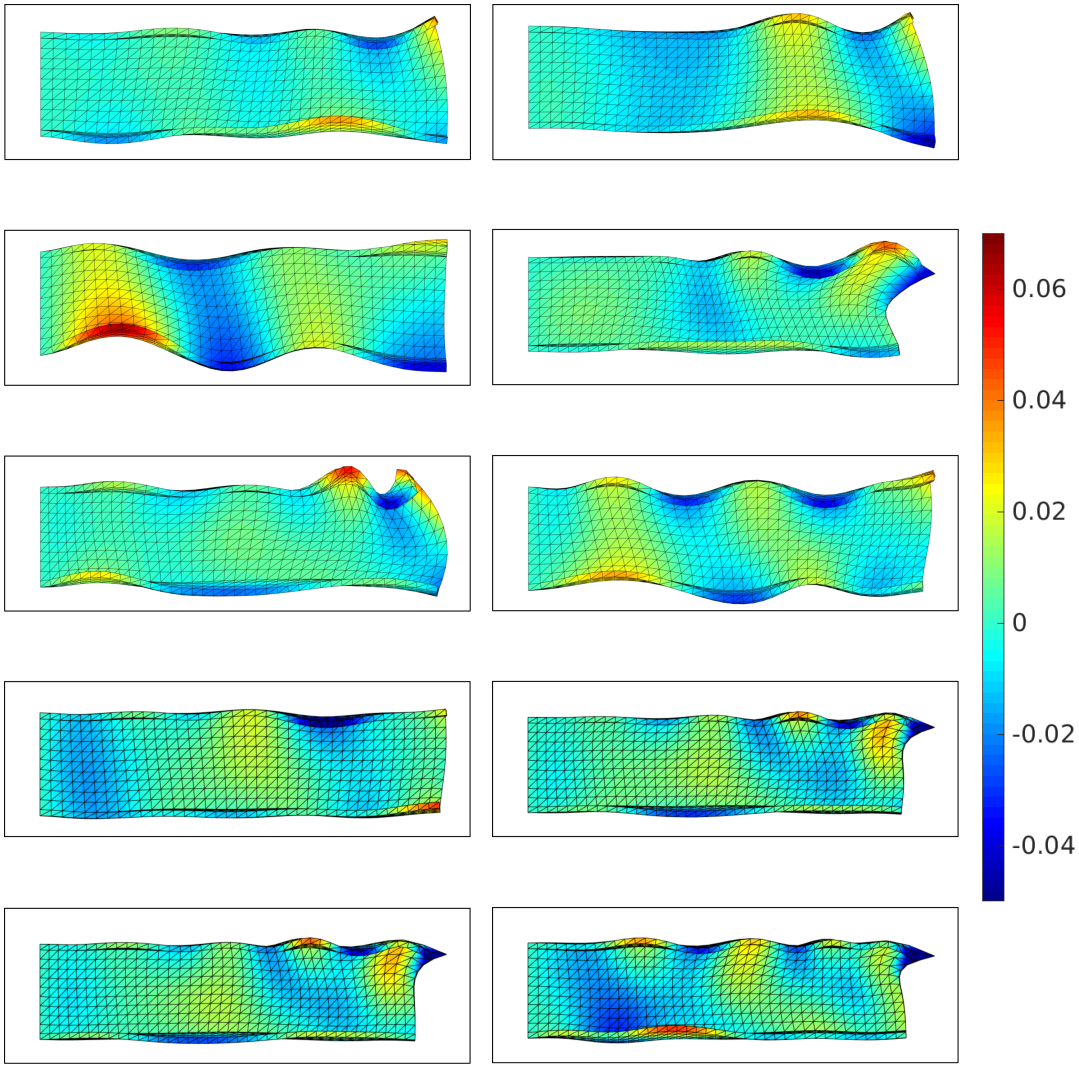
(d) Pseudorandom samples

Figure 7.11: Compare locations of the 20 output samples of reference, proposed POD-Greedy algorithms and statistically based methods. Some samples are chosen at the same point thus gradient size is used to indicate weight of each sample. Notice that for reference method, the points are sampled from the 9×9 grid $\mathcal{P}_{co}^{\text{train}}$, while a 65×65 training set $\mathcal{P}^{\text{train}}$ is used to collect samples for proposed method.



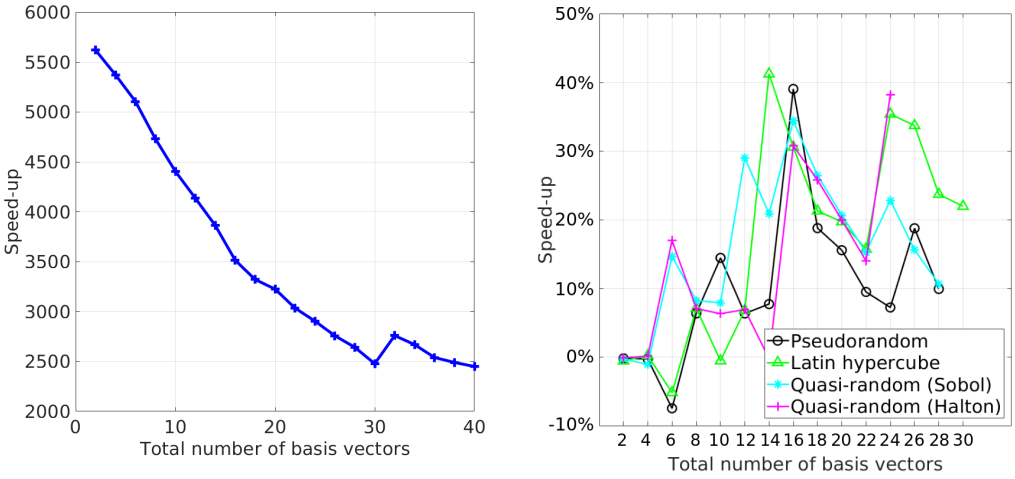
(a) Reference: mode 1-5 (b) Proposed: mode 1-5

Figure 7.12: First 5 modes for reference and proposed POD-Greedy algorithms, $\mu_1 = (E, b) = [10^{-1}, 10^{-1}]$. The colour field is y-displacement.



(a) Reference: mode 6-10 (b) Proposed: mode 6-10

Figure 7.13: Mode 6-10 for reference and proposed POD-Greedy algorithm outputs, $\mu_1 = (E, b) = [10^{-1}, 10^{-1}]$. The colour field is y-displacement. Many modes are similar, which is a possible cause of close convergence between reference and proposed algorithms.



(a) Speed-up of reduced order model over full Newmark solution.

(b) Speed-up of proposed method over statistically based sampling methods when reaching same level of error.

Figure 7.14: Speed-up results for test case 1. In (b), 1 case from pseudorandom and Latin hypercube results are presented. Combine results of fig. 7.10a and (a) it can be seen that with less than 2% of maximum error, the approximation achieves over 2000 times of speed-up over solving the full trajectory. Notice y-axis of (a) is normal scale while (b) uses percentage scale.

tions. Another observation is that as μ decreases and b increases, the mean displacements change from overdamped to underdamped.

7.2.3 Summary

In this section from evaluating the I beam model, three remarks can be made: (i) compare with reference method, the proposed method costs much less time to evaluate a large training set; (ii) in terms of maximum error convergence, the proposed method approaches the reference one, and converges much more rapidly than statistically based sampling approaches; (iii) approximations obtained from the reduced basis are in good agreements with exact solutions. They show that the proposed method is feasible, and can be utilised as a possible solution for real-life applications.

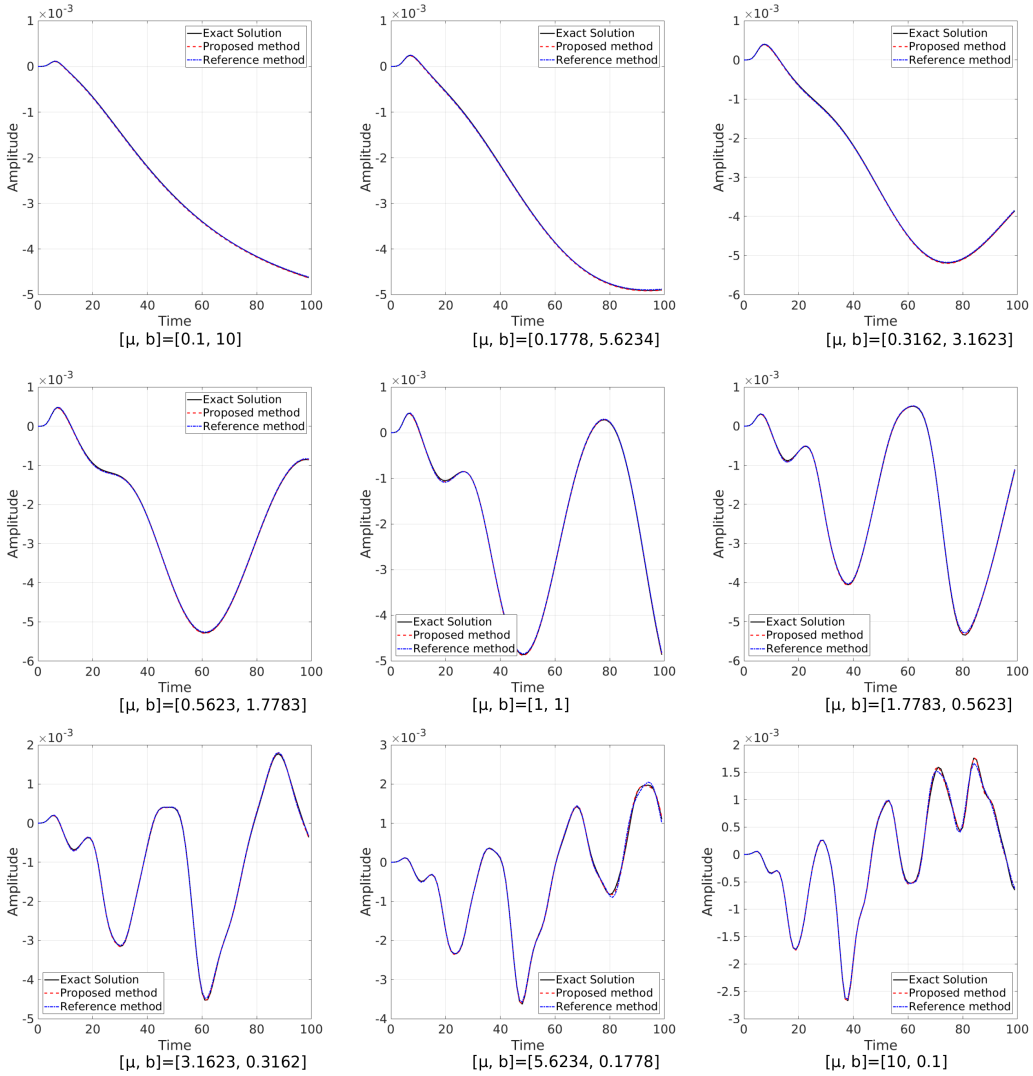


Figure 7.15: Mean displacements of the free end of the I beam under dynamic load for different parametric values. Exact solution and approximations show good agreements.

Chapter 8

Conclusions, Discussion and Future work

8.1 Conclusion

In this thesis, a new POD-Greedy algorithm for elasto-dynamic problems is implemented, which enables users to evaluate a large training set without losing numerical efficiency and accuracy. As a response to section 1.2, the following objectives are met:

1. In chapter 2, numerical examples are provided to demonstrate the Greedy procedure and POD-Greedy algorithms, along with comparisons with statistically based sampling approaches. Results explain the optimality of Greedy procedure and POD-Greedy algorithm, which provide assistance to users to understand the theoretical foundation of this thesis.

2. Based on standard POD-Greedy algorithm and driven by the goal of evaluating a large training set, a new error indicator is proposed in chapter 4, which is derived from the full space-time Newmark representation presented in chapter 3.

The new error indicator is able to process a large training set by interpolating the *dynamic operator* inverse. However direct interpolation of the *dynamic operator* inverse is too expensive due to the large matrix size, therefore the operation is decomposed into solving a set of impulse response problems to form a practical approach. Then a discrete Duhamel's integral is performed to superpose the impulse responses to obtain the approximated displacement error, which is the quantity of interest. Up until now a preliminary numerical experiment is used to investigate the effect of the new error indicator. This is compare with using residual as indicator, see section 4.6 for details.

Computation of the new error indicator is then further accelerated by (i) compressing the impulse responses and only shifting right singular vectors section 4.7; (ii) performing POD on the collection of reduced variable vectors to fix size of the component being interpolated section 4.8. These improvements are of vital importance as it enables rapid interpolation, hence the evaluation of large training set becomes possible.

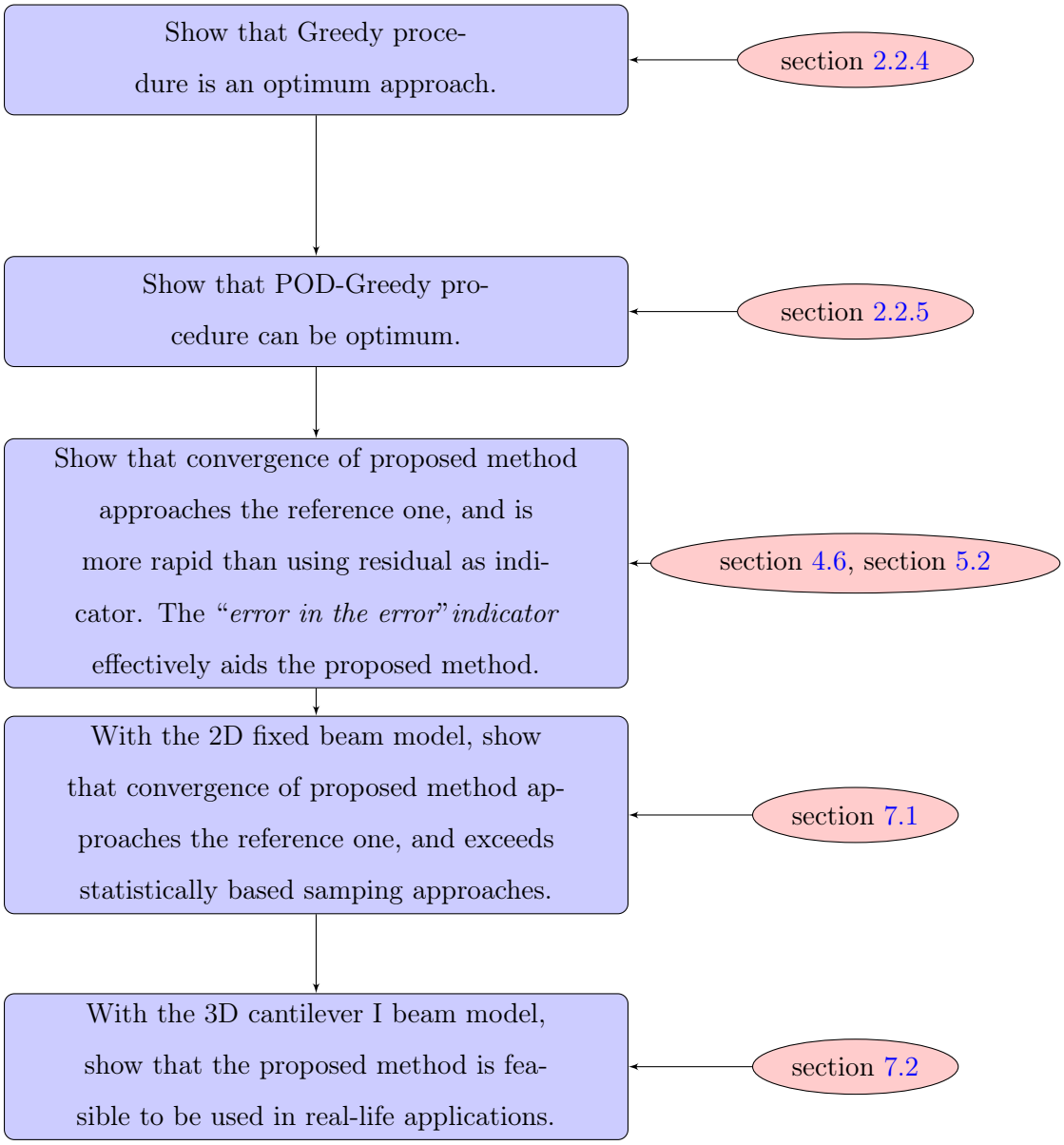
3. A new "*error in the error*" *indicator* is proposed in chapter 5 to focus on providing a criterion of local $h-$ refinement. Using a fixed grid as interpolation samples might be insufficient in terms of accuracy, thus the grid could be refined locally to improve it. The "*error in the error*" *indicator* guides the algorithm to choose when to perform Greedy iterations and when to refine the grid by setting up a pre-defined tolerance. Numerical results are presented in section 5.2 to show the rationale behind this indicator.

4. Once all essential components are proposed, they are combined and the proposed POD-Greedy algorithm is presented. The algorithm utilises (i) the new displacement error indicator, (ii) the "*error in the error*" *indicator*, such that a large training set can be evaluated. A set of illustrative experiments are provided in chapter 7: first a small-scale 2D fixed beam model is used to prove that the algorithm is accurate by showing that convergence of the proposed POD-Greedy algorithm

approaches the reference one, and is more rapid than statically based sampling methods; then a 3D I beam model is used to show that the proposed algorithm is feasible to be used in real-life applications, more specifically, compare with reference algorithm, the proposed one costs much less time to evaluate a same large training set without losing accuracy.

5. A nonintrusive Abaqus/Matlab code coupling technique is presented in appendix A, which can be seamlessly integrated in ROM to utilise Abaqus as an external solver, so that no source code is required during solving for exact solutions. This technique builds a bridge between commercial FE software and source code, allows users to isolate and analyse the problem by only modifying parametric values in some simple text files without developing and validating their own codes.

A simple yet clear sequential introduction of the numerical results is given with the following flow chart:



8.2 Discussion

In many ways this thesis has tried to prove that the proposed POD-Greedy algorithm is accurate and feasible. It is also interesting and important to reflect the advantages and disadvantages of it. In this section, the rationale behind the good performance of

the proposed method is discussed, and possible improvement to be pursued are pointed out.

- For each individual iteration, cost of the reference POD-Greedy algorithm is a function of N_{train}. N_{train} suffers curse of dimensionality, i.e. large number of parameters or highly discretised parametric domain results in substantial N_{train}, hence may lead to prohibitive computational cost.

  The proposed POD-Greedy algorithm alleviate this issue to enable evaluation of large training sets. As shown in section 7.2.2, the proposed method is efficient because it (i) performs fast interpolation; (ii) computes a limited number of exact solutions. For each Greedy iteration, this number is $N_{\text{exact}} = N_i \times N \times N_j \times 2$ for proposed method, while $N_{\text{exact}} = N_{\text{train}}$ for reference method. Hence it can be seen that the proposed method pays off if $N_i \times N \times N_j \times 2 \ll N_{\text{train}}$. In many cases this is easy to be achieved if the parametric domain is moderately or highly discretised. However if N_{train} is small one may use the reference method directly.

- Another component which affects cost of the proposed method is the computation of the displacement vector product matrix $\mathsf{M}_i^{\mathbf{trans}}$. Dimension of $\mathsf{M}_i^{\mathbf{trans}}$ equals to $NN_jN_t+1 \times NN_jN_t+1$ and N_i such matrices need to be computed, therefore the cost increases as number of Greedy iterations grows. The proposed method does not pay off if a very large basis is generated, or a large number of time steps requires to be analysed. However dimension of $\mathsf{M}_i^{\mathbf{trans}}$ is irrelevant to number of DoFs, thus the proposed method should be used when dealing with a large model.

- The proposed method benefits the ROM community by providing a solution for fast training set evaluation, however, this benefit is achieved with a trade-off of memory cost due to interpolation. The impulse responses are treated with SVD hence only singular vectors are stored, which is trivial. However the many displacement vector product matrices required to be stored would result in high memory cost. Possible solutions include memory increase, or developing optimisation approaches

to alleviate this issue.

8.3 Future Work

Many aspects of this work may require further investigations, which are concluded by providing some suggestions for future development.

As discussed in section 8.2, for the proposed POD-Greedy algorithm, our first future work is to eliminate this time effect in the displacement vector product matrix $\mathsf{M}_i^{\mathbf{trans}}$. The size of interpolated component $\mathsf{M}_i^{\mathbf{trans,r}}$ has already been successfully fixed in the Greedy *parameter sweep*, which is done by projecting $\mathsf{M}_i^{\mathbf{trans}}$ in a low-dimensional subspace. However number of time steps still affects computational cost of $\mathsf{M}_i^{\mathbf{trans}}$. The proposed method will be much improved if the time effect can be terminated.

Moreover, as Greedy iterations increase, number of reduced basis vectors also increases which leads to dimension increase of $\mathsf{M}_i^{\mathbf{trans}}$. Another factor is number of parameters and affined terms, if this is large, computations of $\mathsf{M}_i^{\mathbf{trans}}$ is also challenging. However for the reference method, number of parameters and affined terms is also a deciding factor due to curse of dimensionality. Fixing this will contribute to the entire ROM community.

Another future work is to improve the "*error in the error*" *indicator*. Current implementation requires to compute approximated error and perform piecewise linear interpolation over two parameter sample grids, which can be computationally intensive. The first possible improvement is: perform proposed method and only evaluate $\hat{\boldsymbol{e}}$ over $\hat{\boldsymbol{\mathcal{P}}}^i$, once the reduced basis is obtained after a Greedy iteration, set up a sparse sample grid and evaluate the error only over these grid points. If the distance exceed the pre-set tolerance, locally refine $\hat{\boldsymbol{\mathcal{P}}}^i$ in the block which possesses largest error (apply same principle as described in section 5.1). The second improvement is: instead of piecewise linear interpolation, piecewise constant interpolation might be performed to reduce the cost. However

this needs to be verified as the accuracy of interpolation decreases. Suitable combination of the above 2 approaches should reduce cost of the "*error in the error*" *indicator* as well as the proposed method.

Finally, the linear interpolation is hardly satisfying. The possibility of higher-order interpolations has been investigated, such as quadratic interpolation. However numerical instability were observed in the past experiments. In the future, other interpolation polynomials might be explored such as splines to ensure positivity of the polynomials.

Appendix A

Nonintrusive Code Coupling Technique for Abaqus/Matlab

Most of the POD frameworks utilise intrusive methods based on FE source codes, which is implemented by the authors or adapted from open-source codes. For example, POD-based reduced order modelling software package *RBmatlab* is available from http://www.ians.uni-stuttgart.de/MoRePaS/, for PGD-based codes, see [24]. However, these coding works usually requires the detailed knowledge of Finite Element Method from the author, thus is not the best choice in many cases. In order to circumvent these obstacles, nonintrusive approaches which do not require knowledge of the governing equations or source codes are implemented. Moreover, commercial FE software such as Abaqus or ANSYS are widely applied in industrial applications. Commercial FE software has the following benefits: (i) the FE problem is solved in a black box, which only requires definition of parameters, boundary conditions, etc; (ii) reliability of the commercial software is tested by real-world applications, thus can be trusted; (iii) with long-term development and good maintenance, commercial software is able to be used to solve complex problems, while personal-developed source codes lack of such abilities. Motivated by the above, a nonintrusive approach is implemented, which builds a bridge

between parametric-ROM and commercial software Abaqus, such that the parametric exact solutions are obtained from Abaqus and the output data are imported into Matlab POD-Greedy code for further analysis. Abaqus provides many nonintrusive approaches using scripts, such as (i) define .psf file which is a Python script; (ii) record the entire operation as a macro; (iii) use plain-text .inp file; . Author has compared these approaches and find that (i) and (ii) require knowledge of Python, which can be difficult for users, (ii) is also cumbersome as unnecessary operations are also recorded as the macro simply records all operations. The third approach is user-friendly as it is a plain-text file which uses nature language, all parameters are easy to be read and modified, no Python knowledge is required.

A.1 Preparation

A.1.1 Installation of Abaqus on Ubuntu

In this section a step-by-step guide to install Abaqus on Ubuntu is presented. All the following operations are performed with root permissions.

- install the necessary libraries:

```
sudo apt-get update

sudo apt-get install csh

sudo apt-get install libjpeg62

sudo apt-get install libstdc++5
```

- create directories:

```
sudo mkdir ~/abaqus

sudo mkdir ~/abaqustemp

sudo mkdir ~/abaqusworks
```

```
sudo mkdir /media/virtualCD
```

- download the iso file: ABAQUS_6.14-1_x64_Win_Linux.iso
- mount the iso file:

  ```
  sudo mount -o loop /path/to/Abaqus/ABAQUS_6.14-1_x64_Win_Linux.iso /media/virtualCD/
  ```

- change directory to the upper level of the mounted .iso file location: `cd /media`
- run installation file: `sudo ./virtualCD/setup`
- follow the GUI to finish the installation, when prompted to locate the scratch directory, use:

  ```
  ~/abaqustemp
  ```

- run Abaqus by typing the following command in Terminal:

  ```
  /abaqus/Commands/./abq6141 cae -mesa
  ```

  if the command window is transparent, quit and use the following command to start Abaqus:

  ```
  env XLIB_SKIP_ARGB_VISUALS=1 ~/abaqus/6.14-1/code/bin/abq6141 cae -mesa
  ```

A.1.2 Typical procedure of an Abaqus finite element analysis

Many tutorials can be found on the internet to demonstrate the procedure of Finite Element analysis in Abaqus. The settings of the analysis rely heavily on the variation of materials, geometries, assembly of parts, boundary conditions and loads, types of mesh, etc. The general procedure can be summarised as following modules:

Step 1: Start Abaqus/CAE

Step 2: Create parts

Step 3: Create materials

Step 4: Create and assign section properties

Step 5: Assemble the model

Step 6: Define analysis steps

Step 7: Apply boundary conditions and loads

Step 8: Mesh the model

Step 9: Create the job and submit

Step 10: Post-processing

A.1.3 Read nodal coordinates and element connectivities

The nonintrusive Matlab/Abaqus code coupling technique brings us the convenience of exporting information directly from Abaqus. In this section the aim is to export geometrical information, i.e. nodal coordinates and connectivities of structure and inclusions. These information can be read from the Abaqus input file and exported into Matlab to be utilised. In order to read these information automatically, some naming rules need to be followed during creation of model in Abaqus. If taking the fixed beam model in section 2.2.4 as an example: the aim is to read nodal coordinates and element connectivity from the input file. Nodal coordinates of the 2D beam mesh are denoted by $x-y$ coordinates in the .inp file:

```
*Node
1, 73.75, 10.
2, 17.3999996, 5.4000001
3, 66.8499985, 6.55000019
4, 43.2209625, 9.78882408
```

```
5, 11.6499996, 12.3000002
......
```

in this case element type is defined as CPS3: 3-node linear triangles. Connectivity is shown in the following demo, where the first column denotes element number:

```
*Element, type=CPS3
1, 400, 1, 14
2, 400, 34, 1
3, 402, 14, 15
4, 401, 33, 34
5, 425, 403, 422
......
```

therefore in order to read these information into Matlab, one first needs to locate nodal information start and end rows, in this case they are `*Node` and `*Element, type=CPS3`. Then locate element information of start and end rows needs to be located, in this case they are `*Element, type=CPS3` and `*Nset, nset=Set-1`. After obtaining the number of the above rows, the information between can be read by Matlab for further use. Special geometrical information are represented by sets, for instance the following set denotes node number of the inclusion Ω_1:

```
*Nset, nset=Set-1
5, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125
128, 129, 130, 131, 132, 133, 134, 629, 630, 631, 632, 633, 634, 635
638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651
......
```

connectivities of Ω_1 are not completely represented in the input file, however it can be

revealed from the corresponding node set. In case of multiple inclusions, different node set will be created, one then read `*Nset, nset=Set-n` with a loop, and find related node sets. Node sets needs to ne named carefully, best with continuous integers, such that any confusions in the reading process are avoided.

After acquiring nodal coordinates and element connectivities, the mesh can be plotted with Matlab, see fig. A.1. Methods `readINPconsFixie` provides a solution to read the constraint information from .inp file, `readINPgeoMultiInc` reads the geometric information from .inp file, see GitHub repository https://github.com/thinkvantagedu/Matlab.

A.2 The *weak nonintrusive technique*

Extraction of the affined system matrices An important assumption in Model Order Reduction methods is that the system matrices naturally admit an affine form. By admitting this feature, the system matrices can be pre-computed in an computationally intensive "*offline*" phase once and for all. In the "*online*" phase, the system matrices are simply assembled by a linear combination of the pre-computed components with coefficients. Once the model is built, material properties are properly defined and mesh is generated, Abaqus provides us the functionality of exporting system matrices, thus the computational cost is greatly saved without the need of matrix assembly. The procedures of obtaining system matrices require modification of the .inp file: after creating the job, select 'Write Input', an input file with extension '.inp' will be created in the Abaqus directory. Use a text editor to open the Abaqus input file, find the dashed line between definition of Material properties and Steps. Add the following content to the row after the dashed line:

```
*STEP,
name=exportmatrix
```

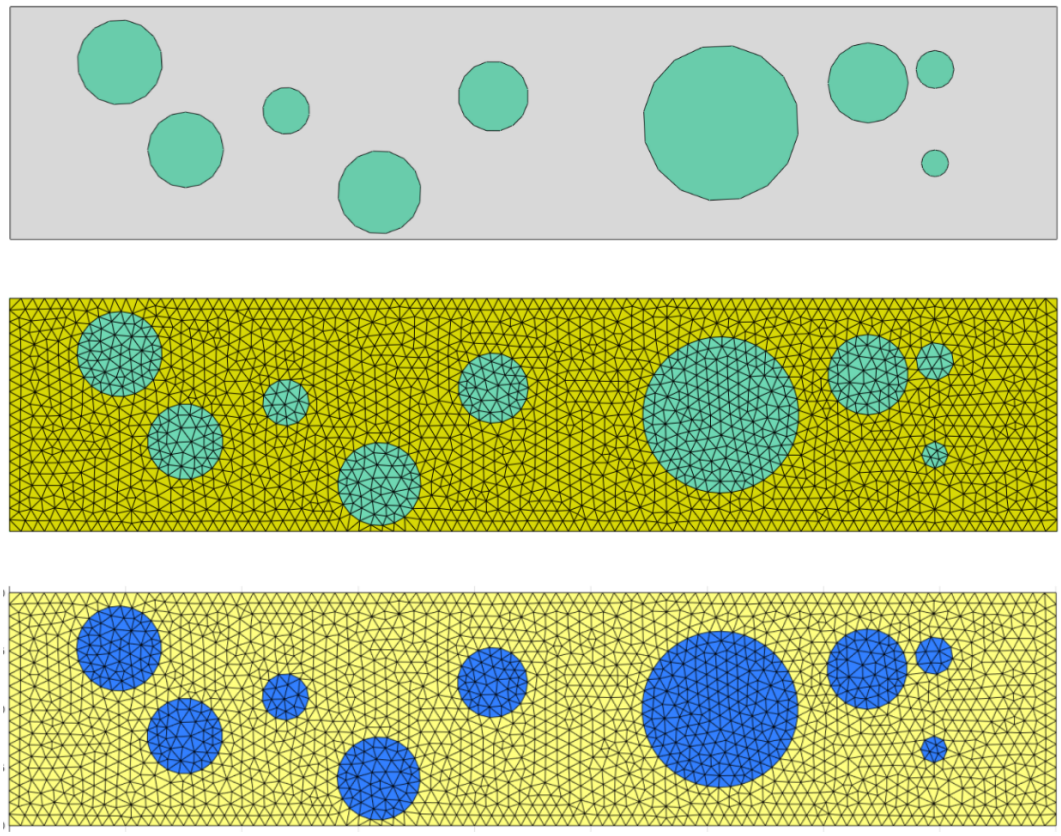


Figure A.1: The parametric beam model (top) and mesh generated in Abaqus (middle) and by Matlab (bottom), notice the Matlab mesh is generated by reading the input file from Abaqus

```
*MATRIX GENERATE, STIFFNESS,
MASS
*MATRIX OUTPUT, STIFFNESS, MASS, FORMAT=MATRIX
INPUT
*END
STEP
** ----------------------------------------
```

after editing the input file, switch back to Abaqus and create job again, this time select 'Input file' from the drop down menu of 'Source'. Find the edited input file and submit the job. After the analysis there will be '.mtx' types of plain-text files in the same directory, which contains information of the mass and stiffness matrices. There is no need for the job to be successfully finished, even if it's aborted, the '.mtx' files will still be exported. Methods '`readMasMTX2DOF`' and '`readStiMTX2DOFBCMod`' read and import the mass and stiffness matrices respectively, see GitHub repository https://github.com/thinkvantagedu/Matlab.

An important premise of exporting the affine system matrices from Abaqus is the correct definitions of material parameters. Here Young's modulus of the fixed beam is taken as an example: define the system stiffness matrix in an affined form

$$\boldsymbol{K}(\mu_i, \mu_s) = \mu_i \boldsymbol{K}_i + \mu_s \boldsymbol{K}_s \tag{A.1}$$

hence $\boldsymbol{K}_i$ and $\boldsymbol{K}_s$ need to be extracted separately. In order to do this, one may define $(\mu_i, \mu_s) = (1, 0)$ for $\boldsymbol{K}_i$ and $(\mu_i, \mu_s) = (0, 1)$ for $\boldsymbol{K}_s$. Since Abaqus does not allow us to define $\mu = 0$, a value which close to machine precision might be chosen instead. In our case μ is set to be 10^{-36} to be as close to 0 as possible. Once $\boldsymbol{K}_i$ and $\boldsymbol{K}_s$ are extracted successfully, they can be further utilised in the POD-Greedy algorithm. The above process reduces the numerical complexity by generating the system matrices by Abaqus. Users can choose to import these system matrices and solve the parametric problems

using source code. Since this approach still requires source code to solve the problem, this method is named as the *weak nonintrusive technique*. Figure A.2 demonstrates the process (take the fix beam model as an example with 1 inclusion and 1 structure).

A.3 The *strong nonintrusive technique*

.inp file structure The .inp file is a type of plain-text files generated by Abaqus, which records all information being input by the user. The Abaqus job can be created directly from the model, or indirectly from the .inp file. Being a type of file specially created by Abaqus, the key words which defines the modules is aligned sequentially, and clearly marked with double asterisks. Following is a minimum example:

```
*Heading
** Job name:  l9h2SingleInc Model name:  Model-1
** Generated by:  Abaqus/CAE 6.12-4
......
** PARTS
......
** ASSEMBLY
......
** Section:  Section-S
......
*Nset, nset=Set-af, instance=beam-1
......
*Elset, elset=Set-lc, instance=beam-1, generate
......
** MATERIALS
......
```

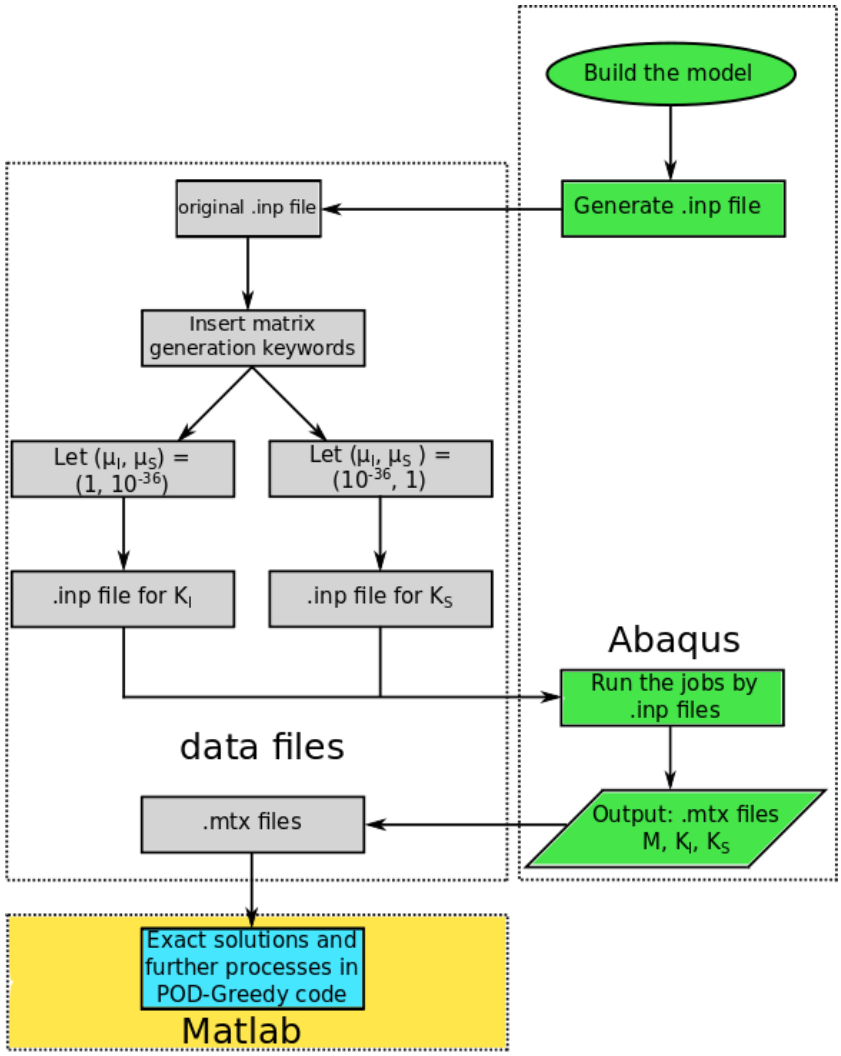


Figure A.2: Flowchart of the *weak nonintrusive technique*, the yellow block indicates that Matlab provides exact solutions.

```
** ----------------------------------------
*STEP, name=exportmatrix
*MATRIX GENERATE, STIFFNESS, MASS
*MATRIX OUTPUT, STIFFNESS, MASS, FORMAT=MATRIX INPUT
*END STEP
** ----------------------------------------
** STEP: Step-af
......
** BOUNDARY CONDITIONS
** Name:  BC-lc Type:  Displacement/Rotation
......
** Name:  BC-rc Type:  Displacement/Rotation
......
** LOADS
** Name:  Load-af Type:  Concentrated force
......
** OUTPUT REQUESTS
......
** FIELD OUTPUT: F-Output-1
......
** HISTORY OUTPUT: H-Output-1
......
```

The material properties are defined in the material module, which is defined as following:

```
   ** MATERIALS
*Material, name=Material-I1
*Density
0.01,
```

```
*Elastic
1, 0.3
```

the values under `*Elastic` are Young's modulus E and Poisson's ratio ν, respectively. Modification of these values allows us to evaluate the parametric model. More specifically, a scalar string $[E, \nu]$ is generated in Matlab and replace the original values in the .inp file, then a new .inp file is generated to be the new Abaqus job. Abaqsu provides the following code to allow users to run Abaqus in Matlab without invoking GUI:

```
system(noGUI job=beamModel inp=beamModel.inp interactive ask_delete=OFF')
```

once the job is finished, a data file with extension .dat contains output responses, what one needs is to read the data file with .dat extension into Matlab to be further processed. Notice that the above process does not involve any operations from the source code, the entire solution process is conducted in Abaqus. Therefore this method is named as the *strong nonintrusive technique*. Flow chart fig. A.3 demonstrates the process.

A.3.1 Numerical example

The *strong nonintrusive technique* is demonstrated with the fixed beam model shown in fig. 2.4. The Young's modulus are set to be $(E_I, E_S) = (10^{-1}, 10^{1})$. Numerical results show completely matched space-time responses between the strong Matlab/Abaqus nonintrusive technique and Matlab source code, see fig. A.4.

Hilber-Hughes-Taylor (HHT) method [39, 87] is used in Abaqus as the time integration scheme for dynamic problems. In the above example where damping is 0, the parameter of the HHT operator needs to be adjusted. Three parameters are involved in the HHT operator: α, β and γ, where Abaqus only allows users to modify α, with the other parameters being adjusted automatically to $\beta = \frac{1}{4} \times (1 - \alpha)^2$ and $\gamma = \frac{1}{2} - \alpha$. In order to satisfy the no damping condition (energy preserving) hence make HHT method equivalent to Newmark method, if let $\alpha = 0$, which can be modified in the STEP module

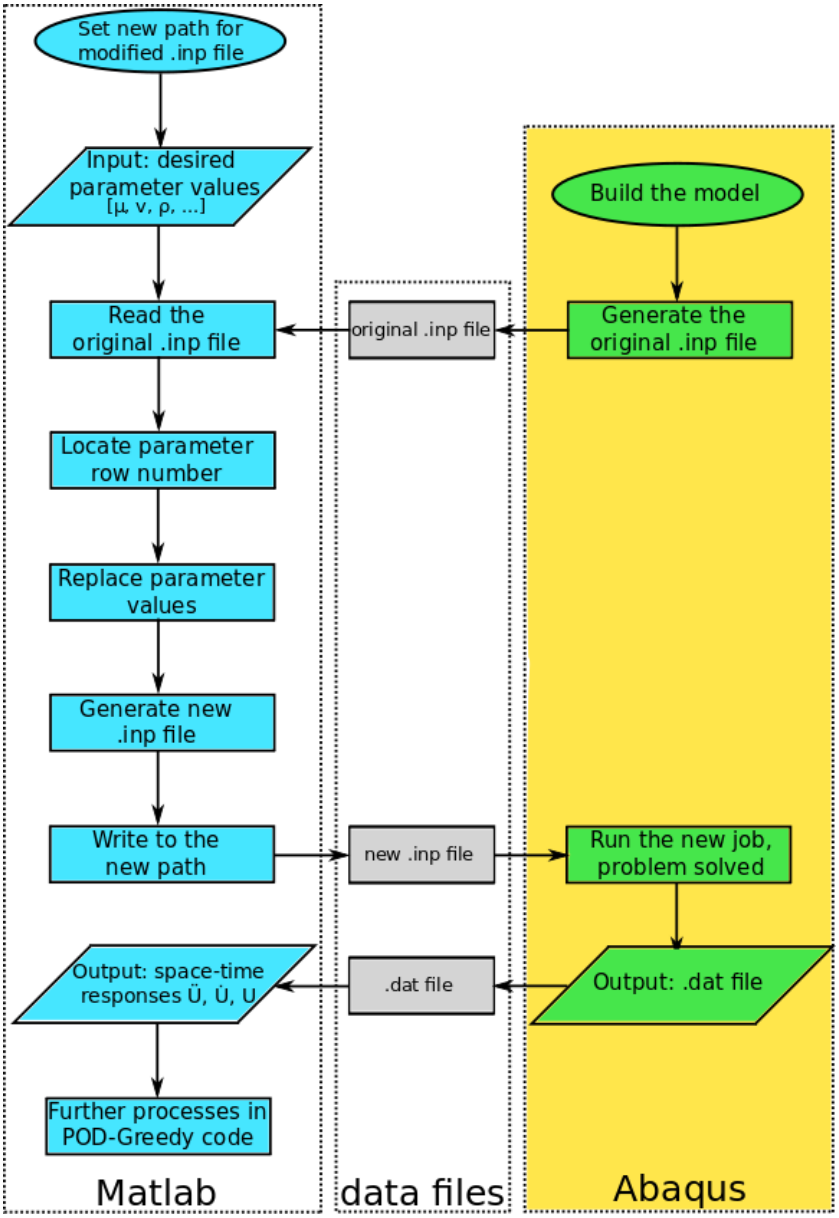


Figure A.3: Flowchart of *strong nonintrusive technique*, the yellow block indicates that Abaqus provides exact solutions.

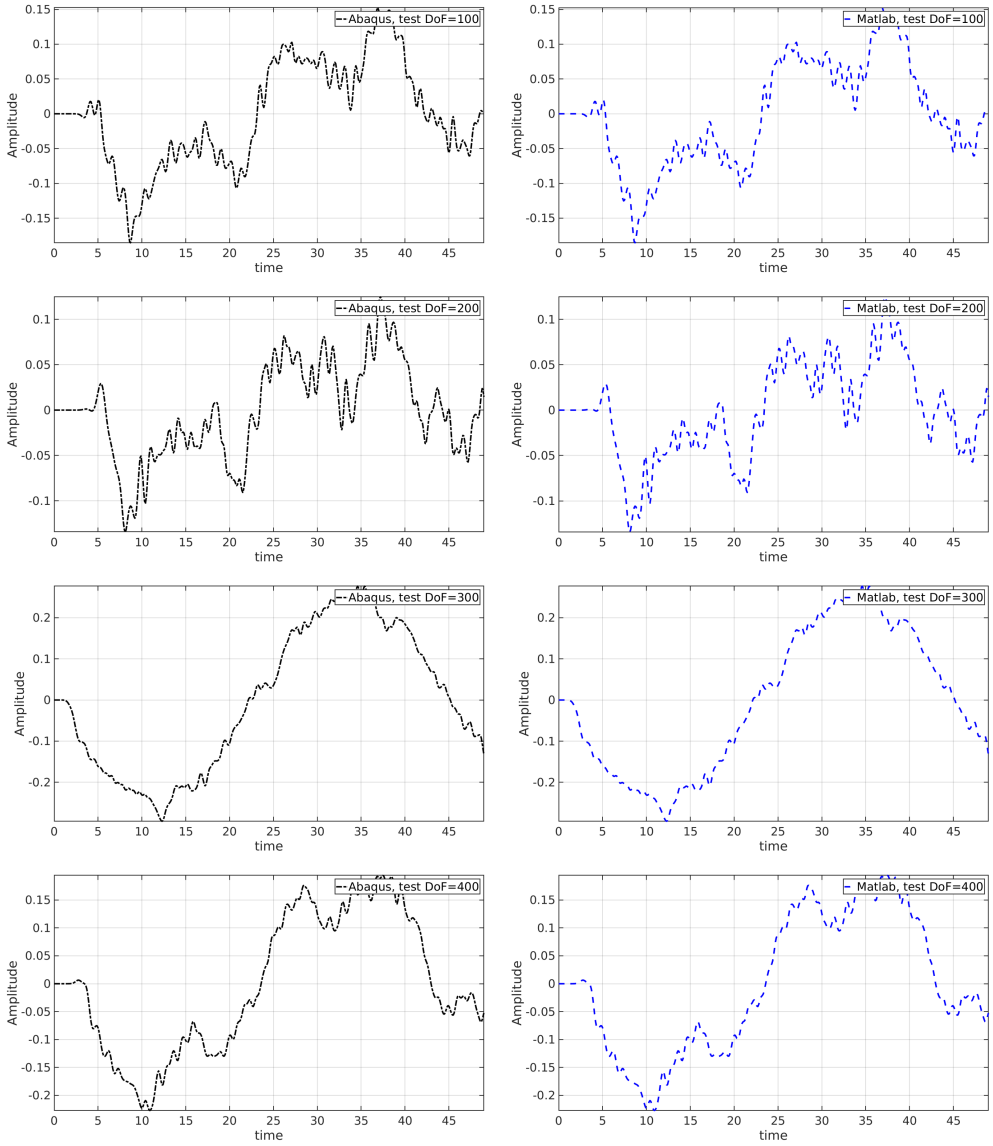


Figure A.4: Compare outputs between the nonintrusive technique and Matlab source code, figures showing DoF 100, 200, 300, 400 of the 2D fixed beam model.

of the .inp file. The modification is as follows

```
** STEP: Step-af
*Step, name=Step-af, inc = 10000
*Dynamic,alpha=0.,direct
0.1, 9.9
```

A.4 Modification of force as a parameter

In Greedy procedure and the POD-Greedy algorithm, the error-residual relation eq. (2.24) and eq. (2.25) are applied to compute the error. Residual appears in a form of external force. Another case is the computation of the impulse responses in section 4.4.1, where many impulses are generated and applied on the system. In these cases, the application of the automated nonintrusive technique requires special treatment to define the forces in the .inp file. The residual force in hyperbolic problem is chosen as an example.

Definition of force in the .inp file contains 3 parts: (i) the set of nodes where force is applied; (ii) the amplitude; (iii) the force step; (iv) load information. The residual is different from the point force as it is applied at each node of the structure. As a result, one needs to define the time-history of the residual force individually for each node. In the original .inp file, the point force is defined by the following commands

```
*Nset, nset=Set-af, instance=beam-1
9,
......
*Amplitude, name=Amp-af
0, 0, 0.1, 0.0332, 0.2, 0.1894, 0.3, 0.
```

```
......
** STEP: Step-af
*Step, name=Step-af, inc = 10000
*Dynamic,alpha=0.,direct
0.1, 9.9
......
** Name:  Load-af Type:  Concentrated force
*Cload, amplitude=Amp-af
Set-af, 2, -1.
```

which need to be modified for the residual case. The new .inp file defines the residual individually at each node as follows

```
*Nset, nset=Set-af1, instance=beam-1
1
*Nset, nset=Set-af2, instance=beam-1
2
......
*Amplitude, name=Amp-af1
......
*Amplitude, name=Amp-af2
......
** STEP: Step-af
*Step, name=Step-af, inc = 10000
*Dynamic,alpha=0.,direct
0.1, 9.9
......
** Name:  Load-af Type:  Concentrated force
```

```
*Cload, amplitude=Amp-af1
Set-af1, 1, 1
*Cload, amplitude=Amp-af2
Set-af1, 2, 1
......
```

Bibliography

[1] British steel universal beam section sizes. https://britishsteel.co.uk/media/40515/british-steel-universal-beams-ub-datasheet.pdf. Accessed: 2018-09-19.

[2] Sondipon Adhikari and Abhishek Kundu. A reduced spectral projection method for stochastic finite element analysis. In *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t*, page 1846, 2011.

[3] Sondipon Adhikari and A Srikantha Phani. RayleighâĂŹs classical damping revisited, 2004.

[4] Christophe Audouze, Florian De Vuyst, and Prasanth B Nair. Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations. *Numerical Methods for Partial Differential Equations*, 29(5):1587–1628, 2013.

[5] I Babuška and BQ Guo. The h, p and h-p version of the finite element method; basis theory and applications. *Advances in Engineering Software*, 15(3):159–174, 1992.

[6] Maxime Barrault, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667–672, 2004.

[7] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[8] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics*, 25(1):539–575, 1993.

[9] Matthew S Bonney and Matthew RW Brake. Determining reduced order models for optimal stochastic reduced order models. *Sandia National Laboratories, Albuquerque, NM, Technical Report No. SAND2015-6896*, 2015.

[10] Piotr Breitkopf, Hakim Naceur, Alain Rassineux, and Pierre Villon. Moving least squares response surface approximation: Formulation and metal forming applications. *Computers & Structures*, 83(17-18):1411–1428, 2005.

[11] Martin D Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.

[12] Tan Bui-Thanh, Karen Willcox, and Omar Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM Journal on Scientific Computing*, 30(6):3270–3288, 2008.

[13] N Cagniart, R Crisovan, Y Maday, and R Abgrall. Model Order Reduction for Hyperbolic Problems: a new framework. working paper or preprint, August 2017.

[14] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. *International Journal for Numerical Methods in Engineering*, 86(2):155–181, 2011.

[15] Anindya Chatterjee. An introduction to the proper orthogonal decomposition. *Current science*, 78(7):808–817, 2000.

[16] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.

[17] Xin Chen, Li Liu, Teng Long, and Zhenjiang Yue. A reduced order aerothermodynamic modeling framework for hypersonic vehicles based on surrogate and pod. *Chinese Journal of Aeronautics*, 28(5):1328–1342, 2015.

[18] Francisco Chinesta, Pierre Ladeveze, and Elías Cueto. A short review on model order reduction based on proper generalized decomposition. *Archives of Computational Methods in Engineering*, 18(4):395–404, 2011.

[19] Seung-Kyum Choi, Ramana V Grandhi, Robert A Canfield, and Chris L Pettit. Polynomial chaos expansion with latin hypercube sampling for estimating response variability. *AIAA journal*, 42(6):1191–1198, 2004.

[20] Anil K Chopra et al. *Dynamics of structures*, volume 3. Prentice Hall New Jersey, 1995.

[21] Paul G Constantine and Qiqi Wang. Residual minimizing model interpolation for parameterized nonlinear dynamical systems. *SIAM Journal on Scientific Computing*, 34(4):A2118–A2144, 2012.

[22] PG Constantine and G Iaccarino. Reduced order models for parameterized hyperbolic conservations laws with shock reconstruction. *Center for Turbulence Research*, 2012.

[23] Peter Craven and Grace Wahba. Smoothing noisy data with spline functions. *Numerische mathematik*, 31(4):377–403, 1978.

[24] Elías Cueto, David González, and Icíar Alfaro. *Proper generalized decompositions: an introduction to computer implementation with Matlab*. Springer, 2016.

[25] Kiran M Desai, Shrikant A Survase, Parag S Saudagar, SS Lele, and Rekha S Singhal. Comparison of artificial neural network (ann) and response surface methodology (rsm) in fermentation media optimization: case study of fermentative production of scleroglucan. *Biochemical Engineering Journal*, 41(3):266–273, 2008.

[26] S Dey, T Mukhopadhyay, and S Adhikari. Metamodel based high-fidelity stochastic analysis of composite laminates: A concise review with critical comparative assessment. *Composite Structures*, 171:227–250, 2017.

[27] Tushar Kanti Dey, Tanmoy Mukhopadhyay, Anupam Chakrabarti, and Umesh Kumar Sharma. Efficient lightweight design of frp bridge deck. *Proceedings of the Institution of Civil Engineers-Structures and Buildings*, 168(10):697–707, 2015.

[28] Markus Dihlmann, Martin Drohmann, and Bernard Haasdonk. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning. *Proc. of ADMOS*, 2011:64, 2011.

[29] Martin Drohmann, Bernard Haasdonk, and Mario Ohlberger. Adaptive reduced basis methods for nonlinear convection–diffusion equations. In *Finite Volumes for Complex Applications VI Problems & Perspectives*, pages 369–377. Springer, 2011.

[30] Martin Drohmann, Bernard Haasdonk, and Mario Ohlberger. Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation. *SIAM Journal on Scientific Computing*, 34(2):A937–A969, 2012.

[31] Nira Dyn, David Levin, and Samuel Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing*, 7(2):639–659, 1986.

[32] Jens L Eftang, Martin A Grepl, and Anthony T Patera. A posteriori error bounds for the empirical interpolation method. *Comptes Rendus Mathematique*, 348(9-10):575–579, 2010.

[33] Jens L Eftang, David J Knezevic, and Anthony T Patera. An hp certified reduced basis method for parametrized parabolic partial differential equations. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):395–422, 2011.

[34] Jens L Eftang, Anthony T Patera, and Einar M Rønquist. An" hp" certified reduced basis method for parametrized elliptic partial differential equations. *SIAM Journal on Scientific Computing*, 32(6):3170–3200, 2010.

[35] M Fares, Jan S Hesthaven, Yvon Maday, and Benjamin Stamm. The reduced basis method for the electric field integral equation. *Journal of Computational Physics*, 230(14):5532–5555, 2011.

[36] Charbel Farhat, Philippe Geuzaine, and Gregory Brown. Application of a three-field nonlinear fluid–structure formulation to the prediction of the aeroelastic parameters of an f-16 fighter. *Computers & Fluids*, 32(1):3–29, 2003.

[37] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[38] Felix Fritzen, Bernard Haasdonk, David Ryckelynck, and Sebastian Schöps. An algorithmic comparison of the hyper-reduction and the discrete empirical interpolation method for a nonlinear thermal problem. *Mathematical and computational applications*, 23(1):8, 2018.

[39] Henri Gavin. Numerical integration for structural dynamics. *Department of Civil and Environmental Engineering, Duke University, Durham, NC*, 2001.

[40] Michel Géradin and Daniel J Rixen. *Mechanical vibrations: theory and application to structural dynamics*. John Wiley & Sons, 2014.

[41] Abdollahi Ghaffari, H Abdollahi, MR Khoshayand, I Soltani Bozchalooi, A Dadgar, and M Rafiee-Tehrani. Performance comparison of neural network training algorithms in modeling of bimodal drug delivery. *International journal of pharmaceutics*, 327(1-2):126–138, 2006.

[42] Olivier Goury, David Amsallem, Stéphane Pierre Alain Bordas, Wing Kam Liu, and Pierre 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(2):213–234, 2016.

[43] Anthony Gravouil and Alain Combescure. Multi-time-step and two-scale domain decomposition method for non-linear structural dynamics. *International Journal for Numerical Methods in Engineering*, 58(10):1545–1569, 2003.

[44] Martin A Grepl and Anthony T Patera. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 39(1):157–181, 2005.

[45] Bernard Haasdonk. Reduced basis methods for parametrized pdes–a tutorial introduction for stationary and instationary problems. *Model reduction and approximation: theory and algorithms*, 15:65, 2017.

[46] Bernard Haasdonk, Markus Dihlmann, and Mario Ohlberger. A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):423–442, 2011.

[47] Bernard Haasdonk and Mario Ohlberger. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(2):277–302, 2008.

[48] Rolland L Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8):1905–1915, 1971.

[49] CE Heaney, AG Buchan, CC Pain, and S Jewer. A reduced order model for criticality problems in reactor physics varyingcontrol rod settings. In *Proceedings of the 24th UK Conference of the Association for Computational Mechanics in Engineering*.

[50] Jan S Hesthaven, Benjamin Stamm, and Shun Zhang. Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation

and reduced basis methodsâĹŮ. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(1):259–283, 2014.

[51] Khac Chi Hoang, Pierre Kerfriden, and Stéphane Bordas. A goal-oriented reduced basis method for the wave equation in inverse analysis. *arXiv preprint arXiv:1305.3519*, 2013.

[52] Khac Chi Hoang, Pierre Kerfriden, and Stéphane Bordas. Space-time goal-oriented reduced basis approximation for linear wave equation. *arXiv preprint arXiv:1305.3528*, 2013.

[53] Khac Chi Hoang, Pierre Kerfriden, BC Khoo, and Stephane Pierre Alain Bordas. An efficient goal-oriented sampling strategy using reduced basis method for parametrized elastodynamic problems. *Numerical Methods for Partial Differential Equations*, 31(2):575–608, 2015.

[54] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[55] John D Jakeman, Richard Archibald, and Dongbin Xiu. Characterization of discontinuities in high-dimensional stochastic problems on adaptive sparse grids. *Journal of Computational Physics*, 230(10):3977–3997, 2011.

[56] Stefan Jakobsson and Olivier Amoignon. Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization. *Computers & Fluids*, 36(6):1119–1136, 2007.

[57] Soo-Chang Kang, Hyun-Moo Koh, and Jinkyo F Choo. An efficient response surface method using moving least squares approximation for structural reliability analysis. *Probabilistic Engineering Mechanics*, 25(4):365–371, 2010.

[58] Pierre Kerfriden, Pierre Gosselet, Sondipon Adhikari, and Stephane Pierre-Alain Bordas. Bridging proper orthogonal decomposition methods and augmented newton–krylov algorithms: an adaptive model order reduction for highly nonlinear

mechanical problems. *Computer Methods in Applied Mechanics and Engineering*, 200(5):850–866, 2011.

[59] Pierre Kerfriden, Olivier Goury, Timon Rabczuk, and Stephane Pierre-Alain Bordas. A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics. *Computer methods in applied mechanics and engineering*, 256:169–188, 2013.

[60] Pierre Kerfriden, Juan-José Ródenas, and SP-A Bordas. Certification of projection-based reduced order modelling in computational homogenisation by the constitutive relation error. *International Journal for Numerical Methods in Engineering*, 97(6):395–422, 2014.

[61] André I Khuri and Siuli Mukhopadhyay. Response surface methodology. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):128–149, 2010.

[62] David J Knezevic and Anthony T Patera. A certified reduced basis method for the fokker–planck equation of dilute polymeric fluids: Fene dumbbells in extensional flow. *SIAM Journal on Scientific Computing*, 32(2):793–817, 2010.

[63] A Kundu and S Adhikari. Dynamic analysis of stochastic structural systems using frequency adaptive spectral functions. *Probabilistic Engineering Mechanics*, 39:23–38, 2015.

[64] A Kundu, S Adhikari, and MI Friswell. Transient response analysis of randomly parametrized finite element systems based on approximate balanced reduction. *Computer Methods in Applied Mechanics and Engineering*, 285:542–570, 2015.

[65] A Kundu, FA DiazDelaO, S Adhikari, and MI Friswell. A hybrid spectral and metamodeling approach for the stochastic finite element analysis of structural dynamic systems. *Computer Methods in Applied Mechanics and Engineering*, 270:201–219, 2014.

[66] Abhishek Kundu and Sondipon Adhikari. Transient response of structural dynamic systems with parametric uncertainty. *Journal of Engineering Mechanics*, 140(2):315–331, 2013.

[67] Karl Kunisch and Stefan Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical analysis*, 40(2):492–515, 2002.

[68] Karl Kunisch and Stefan Volkwein. Optimal snapshot location for computing pod basis functions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 44(3):509–529, 2010.

[69] Peter Lancaster and Kes Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158, 1981.

[70] Genyuan Li, Maxim Artamonov, Herschel Rabitz, Sheng-wei Wang, Panos G Georgopoulos, and Metin Demiralp. High-dimensional model representations generated from low order termsâĂŤlp-rs-hdmr. *Journal of computational chemistry*, 24(5):647–656, 2003.

[71] Genyuan Li, Jishan Hu, Sheng-Wei Wang, Panos G Georgopoulos, Jacqueline Schoendorf, and Herschel Rabitz. Random sampling-high dimensional model representation (rs-hdmr) and orthogonality of its different order component functions. *The Journal of Physical Chemistry A*, 110(7):2474–2485, 2006.

[72] Genyuan Li and Herschel Rabitz. General formulation of hdmr component functions with independent and correlated variables. *Journal of Mathematical Chemistry*, 50(1):99–130, 2012.

[73] Genyuan Li, Sheng-Wei Wang, Herschel Rabitz, Sookyun Wang, and Peter Jaffé. Global uncertainty assessments by high dimensional model representations (hdmr). *Chemical Engineering Science*, 57(21):4445–4460, 2002.

[74] Z Lin, D Xiao, F Fang, CC Pain, and Ionel M Navon. Non-intrusive reduced order modelling with least squares fitting on a sparse grid. *International Journal for Numerical Methods in Fluids*, 83(3):291–306, 2017.

[75] GR Liu, Khin Zaw, and YY Wang. Rapid inverse parameter estimation using reduced-basis approximation with asymptotic error estimation. *Computer Methods in Applied Mechanics and Engineering*, 197(45):3898–3910, 2008.

[76] Yvon Maday, Ngoc Cuong Nguyen, Anthony T Patera, and George SH Pau. A general, multipurpose interpolation procedure: the magic points. 2007.

[77] Avik Mahata, Tanmoy Mukhopadhyay, and Sondipon Adhikari. A polynomial chaos expansion based molecular dynamics study for probabilistic strength analysis of nano-twinned copper. *Materials Research Express*, 3(3):036501, 2016.

[78] Laura Mainini and K Willcox. Surrogate modeling approach to support real-time structural assessment and decision making. *AIAA Journal*, 53(6):1612–1626, 2015.

[79] Balaraman Manohar and Soundar Divakar. An artificial neural network analysis of porcine pancreas lipase catalysed esterification of anthranilic acid with methanol. *Process Biochemistry*, 40(10):3372–3376, 2005.

[80] Marcus Meyer and Hermann G Matthies. Efficient model reduction in non-linear dynamics using the karhunen-loeve expansion and dual-weighted-residual methods. *Computational Mechanics*, 31(1-2):179–191, 2003.

[81] Marc P Mignolet, Adam Przekop, Stephen A Rizzi, and S Michael Spottswood. A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures. *Journal of Sound and Vibration*, 332(10):2437–2460, 2013.

[82] David Modesto, Sergio Zlotnik, and Antonio Huerta. Proper generalized decomposition for parameterized helmholtz problems in heterogeneous and unbounded domains: Application to harbor agitation. *Computer Methods in Applied Mechanics and Engineering*, 295:127–149, 2015.

[83] T Mukhopadhyay, A Mahata, S Dey, and S Adhikari. Probabilistic analysis and design of hcp nanowires: An efficient surrogate based molecular dynamics simulation approach. *Journal of Materials Science & Technology*, 32(12):1345–1351, 2016.

[84] Tanmoy Mukhopadhyay. A multivariate adaptive regression splines based damage identification methodology for web core composite bridges including the effect of noise. *Journal of Sandwich Structures & Materials*, 20(7):885–903, 2018.

[85] Raymond H Myers, Douglas C Montgomery, et al. *Response surface methodology: process and product optimization using designed experiments*, volume 3. Wiley New York, 1995.

[86] S Naskar, T Mukhopadhyay, S Sriramula, and S Adhikari. Stochastic natural frequency analysis of damaged thin-walled laminated composite beams with uncertainty in micromechanical properties. *Composite Structures*, 160:312–334, 2017.

[87] Dan Negrut, Rajiv Rampalli, Gisli Ottarsson, and Anthony Sajdak. On an implementation of the hilber-hughes-taylor method in the context of index 3 differential-algebraic equations of multibody dynamics (detc2005-85096). *Journal of computational and nonlinear dynamics*, 2(1):73–85, 2007.

[88] Anthony Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199(23):1603–1626, 2010.

[89] Peter J. Olver. On multivariate interpolation. Technical report, IN STUD. APPL. MATH, 2004.

[90] VK Pareek, MP Brungs, AA Adesina, and Raj Sharma. Artificial neural network modeling of a multiphase photodegradation system. *Journal of Photochemistry and photobiology A: Chemistry*, 149(1-3):139–146, 2002.

[91] Anthony T Patera, Gianluigi Rozza, et al. Reduced basis approximation and a posteriori error estimation for parametrized partial differential equations, 2007.

[92] Benjamin Peherstorfer, Daniel Butnaru, Karen Willcox, and Hans-Joachim Bungartz. Localized discrete empirical interpolation method. *SIAM Journal on Scientific Computing*, 36(1):A168–A192, 2014.

[93] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

[94] René Pinnau. Model reduction via proper orthogonal decomposition. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 95–109. Springer, 2008.

[95] Christophe PrudâĂŹHomme, Dimitrios V Rovas, Karen Veroy, Luc Machiels, Yvon Maday, Anthony T Patera, and Gabriel Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124(1):70–80, 2002.

[96] SE Pryse, S Adhikari, and A Kundu. Sample-based and sample-aggregated based galerkin projection schemes for structural dynamics. *Probabilistic Engineering Mechanics*, 54:118–130, 2018.

[97] SE Pryse, A Kundu, and S Adhikari. Projection methods for stochastic dynamic systems: A frequency domain approach. *Computer Methods in Applied Mechanics and Engineering*, 338:412–439, 2018.

[98] Alfio Quarteroni, Gianluigi Rozza, and Andrea Manzoni. Certified reduced basis approximation for parametrized partial differential equations and applications. *Journal of Mathematics in Industry*, 1(1):1, 2011.

[99] Annika Radermacher and Stefanie Reese. Model reduction in elastoplasticity: proper orthogonal decomposition combined with adaptive sub-structuring. *Computational Mechanics*, 54(3):677–687, 2014.

[100] Kamron Saniee. A simple expression for multivariate lagrange interpolation.

[101] Sugata Sen. Reduced-basis approximation and a posteriori error estimation for many-parameter heat conduction problems. *Numerical Heat Transfer, Part B: Fundamentals*, 54(5):369–389, 2008.

[102] AA Shah, WW Xing, and V Triantafyllidis. Reduced-order modelling of parameter-dependent, linear and nonlinear dynamic partial differential equation models. *Proc. R. Soc. A*, 473(2200):20160809, 2017.

[103] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of applied mathematics*, 45(3):561–571, 1987.

[104] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[105] Karsten Urban, Stefan Volkwein, and Oliver Zeeb. Greedy sampling using nonlinear optimization. In *Reduced Order Methods for modeling and computational reduction*, pages 137–157. Springer, 2014.

[106] Karen Veroy, Christophe PrudâĂŹHomme, Dimitrios V Rovas, and Anthony T Patera. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. In *Proceedings of the 16th AIAA computational fluid dynamics conference*, volume 3847, pages 23–26. Orlando, FL, 2003.

[107] Stefan Volkwein. Model reduction using proper orthogonal decomposition. *Lecture Notes, Institute of Mathematics and Scientific Computing, University of Graz. see http://www. uni-graz. at/imawww/volkwein/POD. pdf*, 2011.

[108] S Walton, O Hassan, and K Morgan. Reduced order modelling for unsteady fluid flow using proper orthogonal decomposition and radial basis functions. *Applied Mathematical Modelling*, 37(20-21):8930–8945, 2013.

[109] SW Wang, H Levy, G Li, and H Rabitz. Fully equivalent operational models for

atmospheric chemical kinetics within global chemistry-transport models. *Journal of Geophysical Research: Atmospheres*, 104(D23):30417–30426, 1999.

[110] Xiaoou Wang, Yingying Liu, and EK Antonsson. Fitting functions to data in high dimensional design space. In *Proceedings of DETC*, volume 99, page 1999. Citeseer, 1999.

[111] Norbert Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

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

[113] Jeroen AS Witteveen and Hester Bijl. Modeling arbitrary uncertainties using gram-schmidt polynomial chaos. In *44th AIAA aerospace sciences meeting and exhibit*, page 896, 2006.

[114] Jeroen AS Witteveen and Gianluca Iaccarino. Simplex stochastic collocation with random sampling and extrapolation for nonhypercube probability spaces. *SIAM Journal on Scientific Computing*, 34(2):A814–A838, 2012.

[115] Liang Xia, Balaji Raghavan, Piotr Breitkopf, and Weihong Zhang. A pod/pgd reduction approach for an efficient parameterization of data-driven material microstructure models. *Computer Methods in Materials Science*, 13(2):219–225, 2013.

[116] D Xiao, F Fang, AG Buchan, CC Pain, IM Navon, and A Muggeridge. Non-intrusive reduced order modelling of the navier–stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 293:522–541, 2015.

[117] D Xiao, F Fang, C Pain, and G Hu. Non-intrusive reduced-order modelling of the navier–stokes equations based on rbf interpolation. *International Journal for Numerical Methods in Fluids*, 79(11):580–595, 2015.

[118] Byeng D Youn and Kyung K Choi. A new response surface methodology for reliability-based design optimization. *Computers & structures*, 82(2-3):241–256, 2004.

[119] Olivier Zahm, Marie Billaud-Friess, and Anthony Nouy. Projection-based model order reduction methods for the estimation of vector-valued variables of interest. *SIAM Journal on Scientific Computing*, 39(4):A1647–A1674, 2017.

[120] Yan Zhang and Nikolaos V Sahinidis. Uncertainty quantification in co2 sequestration using surrogate models from polynomial chaos expansion. *Industrial & Engineering Chemistry Research*, 52(9):3121–3132, 2012.

[121] Xi Zou, Michele Conti, Pedro Díez, and Ferdinando Auricchio. A nonintrusive proper generalized decomposition scheme with application in biomechanics. *International Journal for Numerical Methods in Engineering*, 113(2):230–251, 2018.