Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Olav Milian Schmitt Gran

Reduced Order Modeling Techniques for Non-Affine Problems in Solid Mechanics

Master's thesis in Industrial Mathematics
Supervisor: Trond Kvamsdal
Co-supervisor: Eivind Fonn, Kjetil A. Johannessen
June 2022

NTNU
Norwegian University of Science and Technology

Olav Milian Schmitt Gran

Reduced Order Modeling Techniques for Non-Affine Problems in Solid Mechanics

Master's thesis in Industrial Mathematics
Supervisor: Trond Kvamsdal
Co-supervisor: Eivind Fonn, Kjetil A. Johannessen
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

To enable the full power of the offline-online concept in Reduced Order Modelling (ROM) the parametrized problem must be affine under variation of the parameters. However, many relevant problems, in particular those involving parametrized geometry, are not affine. The present master project aims to develop techniques for handling such cases applied to solid mechanics. This is done through the development and testing of the *Matrix Least Squares* technique, where *Matrix Least Squares* refers to a least square fitting problem for matrices.

In the case studies, we study three different cases of geometry deformations on a rectangle using the *plane stress* problem of *Constant body force in 2D*. Here we also restrict ourselves to the cases where there is some prescribed displacement on parts of the boundary. (i) The body force, (ii) the prescribed displacement, and (iii) the prescribed traction, on the other parts of the boundary, are independent of the material parametrization parameters we choose, i.e. the domain and thereby its boundary depend on the geometric parametrization parameters we choose. This leads us to testing and observing the effect of the *Matrix Least Squares* technique on affine and non-affine problems.

The conclusion from this thesis is that the developed *Matrix Least Square* technique performs well when we have enough snapshots and are well within the maximum valid geometry parameter range for the Taylor expansion. This is because when using it on a problem with over 10 000 degrees of freedom we observed a computational speedup of order 600 and the technique splits the matrices of an affine problem as desired. We also conclude that the overall objective of the thesis is achieved since we first thoroughly studied the basic theory behind the reduced basis methods using finite elements, presented the *Matrix Least Square* technique, and applied the theory to our restricted cases of the *Linear Elasticity Equations*. Then secondly, we built a solver and tested it, and then finally, we used it to test the *Matrix Least Square* technique and studied a simple numerical example using the technique for different non-affine geometries. For future work, we suggest to investigate the use of sparse snapshot generation, the extension to the 3D case, even more complex geometry and other partial differential equations.

Sammendrag

For å aktivere den fulle kraften til offline-online konseptet i Redusert Ordens Modellering (ROM), må det parametriserte problemet være affint under variasjon av parameterne. Imidlertid er mange relevante problemer, spesielt de som involverer parametrisert geometri, ikke affine. Denne masteroppgaven har som mål å utvikle teknikker for håndtering av slike problemer anvendt på solidmekanikk. Dette gjøres gjennom utvikling og testing av *Matrise Minste Kvadraters*-teknikken, der *Matrise Minste Kvadraters* refererer til et minste kvadraters tilpasningsproblem for matriser.

I casestudiene studerer vi tre forskjellige tilfeller av geometrideformasjoner på et rektangel ved å bruke *planspenningstilfelle*-problemet med *Konstant Volumkraft i 2D*. Her begrenser vi oss også til de tilfellene hvor det finnes foreskrevet forskyvning på deler av randen. (i) Volumkraften, (ii) den foreskrevne forskyvningen og (iii) den foreskrevne trekkraften, på de andre delene av randen, er uavhengige av de materielle parametriseringsparametrene vi velger, dvs. domenet og dermed dets rand avhenger av de geometriske parametriseringsparametere vi velger. Dette fører oss til å teste og observere effekten av *Matrise Minste Kvadraters*-teknikken på affine og ikke-affine problemer.

Konklusjonen fra denne oppgaven er at den utviklede *Matrise Minste Kvadraters*-teknikken fungerer godt når vi har nok øyeblikksbilder og er godt innenfor det maksimale gyldige geometriparameterrommet for Taylor-utviklingen. Dette er fordi når vi brukte det på et problem med over 10 000 frihetsgrader, observerte vi en beregningsmessig tidsbesparelse av størrelsesorden 600 og teknikken deler opp matrisene til et affint problem som ønsket. Vi konkluderer også med at det overordnede målet med oppgaven er oppnådd siden vi først grundig studerte den grunnleggende teorien bak redusert basis modellering ved bruk av endelige elementer, presenterte *Matrise Minste Kvadraters*-teknikken og anvendte teorien på våre begrensede tilfeller av de *Lineære Elastisitetsligningene*. For det andre bygget vi en løser og testet den, og til slutt bruke vi den til å teste *Matrise Minste Kvadraters*-teknikken og studerte et enkelt numerisk eksempel ved å bruke teknikken for forskjellige ikke-affine geometrier. For fremtidig arbeid foreslår vi å undersøke bruken av sparsom øyeblikksbildegenerering, utvidelsen til 3D, enda kompleksere geometri og andre partielle differensiallikninger.

Acknowledgements

I would like to thank my supervisors Trond Kvamsdal, Eivind Fonn and Kjetil A. Johannessen, who have been guiding and helping me though this Master thesis for the past months. Especially I would like to thank Fonn for writing a script for disk-storage of matrices and vectors, and an implementation of the *Matrix Least Squares* algorithm. This script helped me greatly. However, all my supervisors deserve a special thanks for keeping me on track with this thesis. Next, I would like to thank my parents for proofreading the thesis and tips.

Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

dofs — Degrees of freedom

FDM — Finite difference method

FEM — Finite element method

FVM — Finite volume method

HF — High-fidelity

MLS — Matrix Least Square

PCA — Principal component analysis

PDE — Partial differential equation

POD — Proper orthogonal decomposition

RB — Reduced basis

ROM — Reduced order model

SVD — Singular value decomposition

Chapter 1

Introduction

1.1 Background

In several engineering disciplines, partial differential equations (PDEs) are used to describe physical and simulated problems. The problems may be as simple as for instance the heat transfer in an object or the airflow around a stationary object. However, the problems may also be as complicated as adding physical realism to any digital twin [2]. In common for these physical problems is that the governing equation, i.e. the PDEs, often are driven by the laws of nature. Due to complexities, the equations need to be solved numerically on computers quite often. Therefore it is of interest to solve these problems accurately and if possible efficiently.

Conventional methods for solving PDEs include well-established techniques such as Finite Volume Methods (FVM), Finite Difference Methods (FDM) and Finite Element Methods (FEM). Common to all these methods is that they give rise to huge systems to accurately model physical problems, often the number of degrees of freedom will be in the millions or billions [3]. Models arising from such physical problems are usually classified as high-fidelity models.

This makes the computation of the high-fidelity models quite demanding, even in our time of constantly increasing available computational power, accompanied by the progressive improvement of algorithms for solving large linear systems [1]. This is at odds with the ever increasing demand for real-time solutions, which is particularly relevant in optimization, control systems, inverse and inference problems, and uncertainty quantification [3]. Common for many of these solutions is that the PDE in question is parametrized by a suitably small number of input parameters. We usually call these PDEs *parametrized PDEs*.

This leads to the need for reduction. The main motivation for the reduced basis (RB) or reduced-order (ROM) methods. As stated in [1] the RB methods represent a remarkable instance of reduced-order modelling techniques. This is because they exploit the parametric dependence, i.e. the affinity, of the PDE solution. They do so by combining a handful of high-fidelity solutions computed for a set of parameter values. They build a basis for approximating the general solution function space in a low-cost and low-dimensional way, by decreasing the degrees of freedom. This leads to a great computational speed up, but on the cost of some level of accuracy [2]. However, many relevant problems, in particular those involving parametrized geometry, are not affine. This means that when the problem can not be represented affinely, we need an approximated and affine representation. In the case with geometric parameters, i.e. parametrized geometry, this can lead to significantly more complicated representations as seen in [3].

As mentioned in both [1] and [2] the RB methods are divided into two stages, the offline and

online stages. The offline stage is run only once, and the online stage is run once for each problem instance to solve. Here we off-load as much work as possible from the online to the offline stage, making the cost per online execution very small. In other words, this means that the online stage is cheap and computationally fast, since the slow offline stage has been computed beforehand and is considered "free" [2].

1.2 Objectives and Research Topic

The tasks for this thesis was given as:

> To enable the full power of the offline-online concept in Reduced Order Modelling (ROM) the parametrized problem must be affine under variation of the parameters. However, many relevant problems, in particular those involving parametrized geometry, are not affine. The present master project aims to develop techniques for handling such cases applied to solid mechanics.

1.3 Research Approach

In this thesis will study geometry deformations whiles solving the *Linear Elasticity Equations*. We observe that previous techniques, such as the one presented in [3], are based on mapping to a reference domain and finding approximated and affine representations of the Jacobian of the mentioned mapping and its inverse. However, in contrast to previous techniques we find approximated and affine representations after assembling the high-fidelity systems for different choices of our parameters. This is done by treating each of the high-fidelity system matrices and vectors as a snapshot. Choosing a basis for our parameters based on the approximated and affine representations of the Jacobian and its inverse. This leads to a fitting problem, which we call *Matrix Least Squares*. The result of the *Matrix Least Squares* fit is a set of matrices and vectors representing an approximated and affine interpolation of the high-fidelity systems. Therefore the aim of the present master thesis becomes to construct, present and test the *Matrix Least Squares* technique.

1.4 Working Method and Report Structure

As seen above, several fields where reduced-order models can be applied, and several approaches to construct these models exist. Especially for today's age the use in digital twins is interesting. However, in many cases, in particular those involving parametrized geometry, the problems are not affine. Therefore the emphasis of this thesis is to construct, present and test the *Matrix Least Squares* technique. This is done by using the Galerkin reduced basis method for the *Linear Elasticity Equations*.

To do this we start by presenting the relevant FEM and ROM theory in chapter 2, before continuing with the construction of the *Matrix Least Squares* method and the *Linear Elasticity Equations* with a mapping to the reference domain. Here we also restrict ourselves to the cases where there is some prescribed displacement on parts of the boundary. (i) The body force, (ii) the prescribed displacement, and (iii) the prescribed traction, on the other parts of the boundary, are independent of the material parametrization parameters we choose, i.e. the domain and thereby its boundary depend on the geometric parametrization parameters we choose.

In chapter 3 start with an introduction to our numerical case, before preforming the Patch Test. Continuing with chapter 3 we determine a basis for the *Matrix Least Squares* method, i.e. the *Matrix Least Squares Functions*. We end chapter 3 by presenting the problem of constant body

force in 2D and discussing some numerical results, testing and observing the effect of the *Matrix Least Squares* technique.

Finally, in chapter 4 we give a summary of the main results, and present our conclusion and suggestions for further work.

The Python code building the solver, the Python code used in this thesis and the log files can be found via the doi-link [4].

Finally, I would like to note that parts of this chapter and multiple sections in the theory chapter 2 are reused and partly rewritten or edited from my Specialization Project.

Chapter 2

Theory

Since we will be solving the *linear elasticity equations* we in first section of this chapter present some relevant theory for solving parametrized partial differential equations. Following *Reduced Basis Methods for Partial Differential Equations* by Quarteroni, Manzoni and Negri (QMN2016) [1] we then will be presenting the Galerkin finite element method and the construction of a reduced-order method based on the Galerkin high-fidelity approximation using the Proper Orthogonal Decomposition (POD) method. Next, since all the previous five sections assume that we have an affine problem we present the idea and theory behind the Matrix Least Squares method. Lastly we apply the presented theory to the *Linear Elasticity Equations* by presenting the strong and weak formulation, the mapping to the reference domain and the resulting algebraic systems.

2.1 Parametrized Partial Differential Equations

As described in chapter 1 of QMN2016 [1], a *parametrized partial differential equation* (parametrized PDE) is a partial differential equation depending on some set of parameters. Let us denote the input parameters by a vector, $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 & \cdots & \mu_p \end{bmatrix}^\top$, which is the input parameters belonging to the parameter space $\mathscr{P} \in \mathbb{R}^p$. The parameter space $\mathscr{P}$ is a closed and bounded subset in the Euclidean space $\mathbb{R}^p, p \geq 1$. Then the exact solution of the parametrized PDE given by the input parameters $\boldsymbol{\mu}$ can be written as $u(\boldsymbol{\mu})$, where $u : \mathscr{P} \to \mathbb{V}$ is the map mapping any $\boldsymbol{\mu} \in \mathscr{P}$ to the solution $u(\mu)$ belonging to a suitable function space $\mathbb{V}$.

2.1.1 Strong Formulation

The strong formulation of a parametrized PDE is defined in section 3.1 of QMN2016 [1];

> Let us denote by $\Omega \subset \mathbb{R}^d, d = 1, 2, 3$ denote the reference domain, $\mathbb{V} = \mathbb{V}(\Omega)$ a suitable Hilbert space, $\mathbb{V}'$ its dual. For every $\boldsymbol{\mu} \in \mathscr{P}$ let $L(\boldsymbol{\mu}) : \mathbb{V} \to \mathbb{V}'$ denotes [sic] a second-order differential operator and $f(\boldsymbol{\mu}) : \mathbb{V} \to \mathbb{R}$ a linear and continuous form on $\mathbb{V}$, that is an element of $\mathbb{V}'$. In abstract from, the parametrized problem we focus on can be written as follows:
>
> given $\boldsymbol{\mu} \in \mathscr{P}$, find the solution $u(\boldsymbol{\mu}) \in \mathbb{V}$ of

$$L(\boldsymbol{\mu})u(\boldsymbol{\mu}) = f(\boldsymbol{\mu}) \quad \text{in} \quad \mathbb{V}'. \tag{2.1}$$

2.1.2 Weak Formulation

For the sake of construction and numerical approximation the weak formulation of problem (2.1) as stated in section 3.1 of QMN2016 [1] is introduced;

given $\boldsymbol{\mu} \in \mathbb{P}$, find $u(\boldsymbol{\mu}) \in \mathbb{V}$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) \quad \forall v \in \mathbb{V}, \tag{2.2}$$

where the parametrized bilinear form $a(\cdot, \cdot; \boldsymbol{\mu}) : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ is obtained from $L(\boldsymbol{\mu})$,

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) =_{\mathbb{V}'} \langle L(\boldsymbol{\mu})u, v\rangle_{\mathbb{V}} \, \forall u, v \in \mathbb{V}, \tag{2.3}$$

and encodes the differential operator. The linear form $f(.; \boldsymbol{\mu}) : \mathbb{V} \to \mathbb{R}$ denotes

$$f(v; \boldsymbol{\mu}) =_{\mathbb{V}'} \langle f(\boldsymbol{\mu}), v\rangle_{\mathbb{V}} \, . \tag{2.4}$$

As stated in chapter 5 in QMN2016 [1] the set of all solutions is called the solution manifold

$$\mathscr{M} = \{u(\boldsymbol{\mu}) \in \mathbb{V} : \boldsymbol{\mu} \in \mathscr{P}\} \subset \mathbb{V}. \tag{2.5}$$

For more details in deriving the weak formulation we refer the interested readers to *Numerical Models for Differential Problems* by Quarteroni (Q2009) [5] and the book by Tröltzsch [6].

Remark 2.1. *It is possible to find a solution to the problem* (2.1) *through the weak formulation even if the solution u is not twice differentiable, as seen in section 3.2.1 in Q2009 [5]. With the example modelling the equilibrium configuration of an elastic string, where u is its vertical displacement, and the string is fixed in both ends and pulled with a force at the middle of the string, forming a V-shape. Here a solution is possible because the weak formulation weakens the differentiability requirements, moving from a second-order differential problem to a first order differential problem.*

2.1.3 Well-posedness of the Weak Formulation

The well-posedness of the weak formulation (2.2) can in general be established by similar arguments as in the Lax-Milgram Lemma and Neĉas Theorem, both found in chapter 2 of QMN2016 [1]. Knowing this we present first present Lax-Milgram Lemma from section 2.2.1 of QMN2016;

Lemma 2.1 (Lax-Milgram)**.** *Let $\mathbb{V}$ be a Hilbert space, $a : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ a continuous, strongly coercive bilinear form on $\mathbb{V} \times \mathbb{V}$, and $\mathbb{V} \to \mathbb{R}$ a bounded linear functional on $\mathbb{V}$. Then, the abstract variational problem (P_1) has a unique solution and it satisfies the stability estimate*

$$\|u\|_{\mathbb{V}} \leq \frac{1}{\alpha}\|f\|_{\mathbb{V}'}. \tag{2.6}$$

Here $\alpha > 0$ is the coercivity constant of $a(\cdot, \cdot)$ that exists when $a(\cdot, \cdot)$ is strongly coercive, i.e.

$$a(v, v) \geq \alpha\|v\|^2_{\mathbb{V}} \quad \forall v \in \mathbb{V}. \tag{2.7}$$

Next, we present the general well-posedness statement from section 3.1 of QMN2016;

Assuming that $a(\cdot,\cdot;\boldsymbol{\mu}) : \mathbb{V}\times\mathbb{V} \to \mathbb{R}$ is continues over $\mathbb{V}\times\mathbb{V}$ for any $\boldsymbol{\mu}\in\mathscr{P}$, i.e there exists a constant $\bar{\gamma} > 0$ such that

$$\gamma(\boldsymbol{\mu}) = \sup_{v\in\mathbb{V}}\sup_{w\in\mathbb{V}} \frac{a(v,w;\boldsymbol{\mu})}{\|v\|_{\mathbb{V}}\|w\|_{\mathbb{V}}} < \overline{\gamma} \quad \forall\boldsymbol{\mu}\in\mathscr{P}. \tag{2.8}$$

Since $f(\boldsymbol{\mu}) \in \mathbb{V}'$ for any $\boldsymbol{\mu}\in\mathscr{P}$, also $f(.,\boldsymbol{\mu})$ is a continuous from, i.e there exists a constant $\bar{\gamma}_F > 0$ such that

$$\gamma_F(\boldsymbol{\mu}) = \sup_{w\in\mathbb{V}} \frac{f(w;\boldsymbol{\mu})}{\|w\|_{\mathbb{V}}} < \overline{\gamma}_F \quad \forall\boldsymbol{\mu}\in\mathscr{P}. \tag{2.9}$$

Here $\gamma(\boldsymbol{\mu})$ and $\gamma_F(\boldsymbol{\mu})$ represent the continuity factors of $a(\cdot,\cdot;\boldsymbol{\mu})$ and $f(.,\boldsymbol{\mu})$. For stability, we assume that there exist a constant $\beta_0 > 0$ such that for each $\boldsymbol{\mu}\in\mathscr{P}$,

$$\beta(\boldsymbol{\mu}) = \inf_{v\in\mathbb{V}}\sup_{w\in\mathbb{V}} \frac{a(v,w;\boldsymbol{\mu})}{\|v\|_{\mathbb{V}}\|w\|_{\mathbb{V}}} \geq \beta_0, \tag{2.10}$$

$$\inf_{w\in\mathbb{V}}\sup_{v\in\mathbb{V}} \frac{a(v,w;\boldsymbol{\mu})}{\|v\|_{\mathbb{V}}\|w\|_{\mathbb{V}}} > 0. \tag{2.11}$$

Where we call $\beta(\boldsymbol{\mu})$ the inf-sup stability factor and we say that $a(\cdot,\cdot;\boldsymbol{\mu})$ is *inf-sup* stable.

Provided that the continuity properties and the stability properties are verified, problem (2.2) admits a unique solution thanks to Neĉas Theorem. Furthermore, the following stability estimate holds for all $\boldsymbol{\mu}\in\mathscr{P}$

$$\|u(\boldsymbol{\mu})\|_{\mathbb{V}} \leq \frac{1}{\beta(\boldsymbol{\mu})}\|f(.;\boldsymbol{\mu})\|_{\mathbb{V}'} \leq \frac{1}{\beta_0}\|f(.;\boldsymbol{\mu})\|_{\mathbb{V}'} \tag{2.12}$$

A proof of the Lax-Milgram Lemma above can be seen in e.g section 3.4.1 of [7].

Remark 2.2. *Note that the abstract variational problem (P_1) not stated here is equal to the weak formulation in* (2.2) *if we remove the dependence on* $\boldsymbol{\mu}$.

Furthermore, from the general well-posedness statement above we see that the continuity properties (2.8) and (2.9) establish the continuity of $a(\cdot,\cdot;\boldsymbol{\mu})$ and $f(.;\boldsymbol{\mu})$, and (2.10) establishes the weak coercivity of $a(\cdot,\cdot;\boldsymbol{\mu})$. With regards to the coercivity of $a(\cdot,\cdot;\boldsymbol{\mu})$ we state the first remark in section 3.1 of QMN2016 [1];

Remark 2.3. *A particular case where the assumptions in* (2.10)- (2.11) *are verified is when, for each* $\boldsymbol{\mu}\in\mathscr{P}$, *there exists* $\alpha_0 > 0$ *such that*

$$\alpha(\boldsymbol{\mu}) = \inf_{v\in\mathbb{V}} \frac{a(v,v;\boldsymbol{\mu})}{\|v\|^2_{\mathbb{V}}} \geq \alpha_0. \tag{2.13}$$

In this case $a(\cdot,\cdot;\boldsymbol{\mu})$ *is coercive and* $\alpha(\boldsymbol{\mu})$ *is the coercive factor.*

2.1.4 Sobolev Spaces

In section 2.1.2 the weak formulation was introduced. For the numerical analysis of the linear elasticity equations in section 2.7 we need that the solution $u(\boldsymbol{\mu})$ and the test function v lie in a certain space for the weak formulation to hold. In QMN2016 [1] this space is introduced as the *Sobolev Spaces* $\mathbb{H}^1(\Omega)$ and $\mathbb{H}^1_{\Gamma_{\mathrm{D}}}(\Omega)$, where Γ_{D} is the Dirichlet part of the boundary Γ of our domain Ω. To familiarize the reader with the notion of these spaces we refer to [8, 5], and state the definitions (for $k = 1$) from section 2.4 in Q2009 [5];

Definition 2.1. *Let Ω be an open set of $\mathbb{R}^d$. We call the Sobolev space of order $k = 1$ on Ω the space formed by the totality of functions of $L^2(\Omega)$ whose (distributional) derivatives up to the first order ($k = 1$) belong to $L^2(\Omega)$:*

$$\mathbb{H}^1(\Omega) = \left\{ f \in L^2(\Omega) : Df \in L^2(\Omega) \right\}. \tag{2.14}$$

Using the definition above we define $\mathbb{H}^1_{\Gamma_D}(\Omega)$ as

$$\mathbb{H}^1_{\Gamma_D}(\Omega) = \left\{ f \in \mathbb{H}^1(\Omega) : f|_{\Gamma_D} = 0 \right\}. \tag{2.15}$$

The Sobolev spaces $\mathbb{H}^1(\Omega)$ are Hilbert spaces with respect to the following inner product

$$(f, g)_{\mathbb{H}^1(\Omega)} = \int_\Omega fg \, d\Omega + \int_\Omega \nabla f \cdot \nabla g \, d\Omega \tag{2.16}$$

which induces the $\mathbb{H}^1$-norm

$$\|f\|_{\mathbb{H}^1(\Omega)} = \sqrt{(f, g)_{\mathbb{H}^1(\Omega)}}. \tag{2.17}$$

Finally, we define the $\mathbb{H}^1$-seminorm

$$|f|_{\mathbb{H}^1(\Omega)} = \|\nabla f\|_{L^2(\Omega)} \tag{2.18}$$

2.1.5 The Energy Norm

Following the previous section we also want to define the problem dependent *energy norm*. We do this through section 2.2.1 in QMN2016 [1];

When the bilinear form $a(\cdot, \cdot; \boldsymbol{\mu})$ is symmetric — that is, $a(v, w; \boldsymbol{\mu}) = a(w, v; \boldsymbol{\mu}) \quad \forall v, w \in \mathbb{V}$ — it defines an inner product over $\mathbb{V}$

$$(v, w)_a = a(v, w) \tag{2.19}$$

and the corresponding norm

$$\|.\|_a = \sqrt{a(.,.)} \tag{2.20}$$

induced by this scalar product is called *energy norm*.

Concerning when the *energy norm* is defined, we make the following remark;

Remark 2.4. *The energy norm is only defined when the bilinear for $a(\cdot, \cdot; \boldsymbol{\mu})$ is symmetric positive definite, i.e it is symmetric as described in the quote above and*

$$a(v, v; \boldsymbol{\mu}) > 0 \quad \forall v \in \mathbb{V} \setminus \{0\}. \tag{2.21}$$

2.2 The Galerkin Finite Element Method

In this section we present the details on deriving a linear system from the *Galerkin problem* as this sets the foundation for the *Galerkin reduced basis* problem which we construction from the Galerkin high-fidelity approximation.

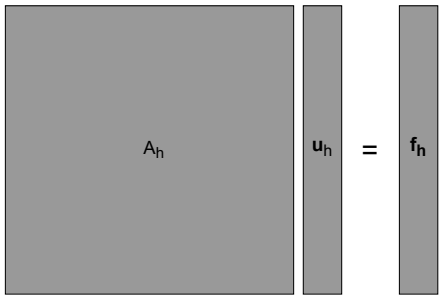


Figure 2.1. The linear system in equation (2.25) with $A_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}, \boldsymbol{u}_h(\boldsymbol{\mu}) \in \mathbb{R}_h^N$ and $\boldsymbol{f}_(\boldsymbol{\mu}) \in \mathbb{R}_h^N$

2.2.1 Galerkin High-fidelity Approximation

To discretize problem (2.2) we introduce the approximation space $\mathbb{V}_h \subset \mathbb{V}$ where we seek the weak solution u_h, which we also called the high-fidelity solution. This leads to the Galerkin high-fidelity approximation of the weak formulation defined in section 3.2 of QMN2016 [1];

find $u_h(\boldsymbol{\mu}) \in \mathbb{V}_h$ such that

$$a(u_h(\boldsymbol{\mu}), v_h; \boldsymbol{\mu}) = f(v_h; \boldsymbol{\mu}) \quad \forall v_h \in \mathbb{V}_h. \tag{2.22}$$

Remark 2.5. *Note here that, the subscript h is related to the grid size of the high-fidelity system and as noted above $\mathbb{V}_h$ is some finite-dimensional subspace of $\mathbb{V}$.*

Now following section 4.1 in Q2009 [5] we denote by $N_h = \dim(\mathbb{V}_h)$ and $\{\varphi_i\}_{i=1}^{N_h}$ the dimension and basis for $\mathbb{V}_h$. We can now write u_h in therms of the basis

$$u_h(\boldsymbol{\mu}) = u_h(\boldsymbol{x}; \boldsymbol{\mu}) = \sum_{j=1}^{N_h} u_h^{(j)}(\boldsymbol{\mu}) \varphi_j(\boldsymbol{x}) \tag{2.23}$$

where $\boldsymbol{u_h}(\boldsymbol{\mu}) = \begin{bmatrix} u_h^{(1)}(\boldsymbol{\mu}) & \dots & u_h^{(N_h)}(\boldsymbol{\mu}) \end{bmatrix}^\top$ holds the coefficients associated with the degrees of freedom of $u_h(\boldsymbol{\mu})$. Now since equation (2.22) holds for all $\forall v_h \in \mathbb{V}_h$ we can write the test function as $v_h(\boldsymbol{x}) = \varphi_i(\boldsymbol{x})$. Then the Galerkin problem becomes

$$\begin{aligned} a\left(\sum_{j=1}^{N_h} u_h^{(j)}(\boldsymbol{\mu}) \varphi_j(\boldsymbol{x}), \varphi_i(\boldsymbol{x}); \boldsymbol{\mu}\right) &= f(\varphi_i(\boldsymbol{x}); \boldsymbol{\mu}) \\ \sum_{j=1}^{N_h} a\left(u_h^{(j)}(\boldsymbol{\mu}) \varphi_j(\boldsymbol{x}), \varphi_i(\boldsymbol{x}); \boldsymbol{\mu}\right) &= f(\varphi_i(\boldsymbol{x}); \boldsymbol{\mu}) \end{aligned} \tag{2.24}$$

which must hold for all $i = 1, \dots, N_h$. Meaning that the Galerkin high-fidelity approximation is equivalent to solving the linear system

$$A_h(\boldsymbol{\mu}) \boldsymbol{u}_h(\boldsymbol{\mu}) = \boldsymbol{f}_h(\boldsymbol{\mu}) \tag{2.25}$$

where $A_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times N_h}$ is the stiffness matrix and $\boldsymbol{f}_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$ is the right-hand side load vector both depending on depending on $\boldsymbol{\mu}$, and with elements

$$\begin{aligned} (A_h(\boldsymbol{\mu}))_{ij} &= a(\varphi_j, \varphi_i; \boldsymbol{\mu}) \, 1 \leq i, j \leq N_h \\ (\boldsymbol{f}_h(\boldsymbol{\mu}))_i &= f(\varphi_i; \boldsymbol{\mu}) \, 1 \leq i \leq N_h \end{aligned} \tag{2.26}$$

One can see the linear system of (2.25) in figure 2.1.

From section 5.1 in QMN2016 [1] we also state the definition of the discrete solution manifold as

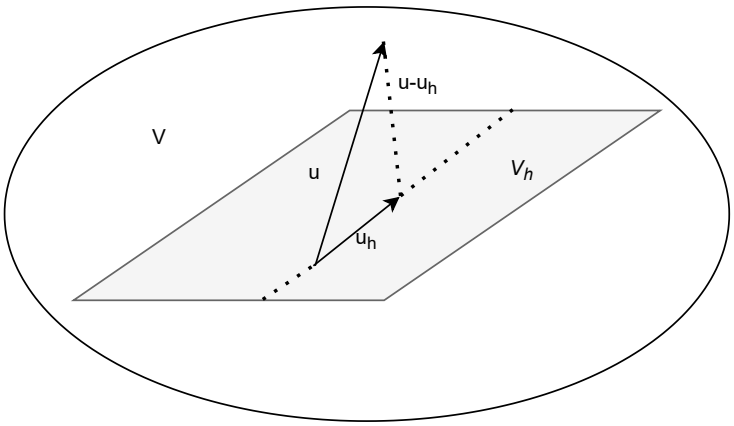


Figure 2.2. Geometric interpretation of the Galerkin orthogonality

$$\mathscr{M}_h = \{u_h(\boldsymbol{\mu}) \in \mathbb{V}_h : \boldsymbol{\mu} \in \mathscr{P}\} \subset \mathbb{V}_h, \tag{2.27}$$

where obviously it is a subset of the exact solution manifold, i.e. $\mathscr{M}_h \subset \mathscr{M}$. Furthermore, we assume that choosing the discretization fine enough, or equivalently by choosing h small enough, we can approximate $\mathbb{V}$ by $\mathbb{V}_h$ and $\mathscr{M}$ by $\mathscr{M}_h$ within an acceptable approximation error.

2.2.2 Galerkin Orthogonality

By fixing $\boldsymbol{\mu} \in \mathscr{P}$ such that $u(\boldsymbol{\mu}) = u, u_h(\boldsymbol{\mu}) = u_h, a(\cdot, \cdot; \boldsymbol{\mu}) = a(\cdot, \cdot)$ and $f(.; \boldsymbol{\mu}) = f(.)$. We now present the Galerkin orthogonality as stated in Lemma 4.1 of Q2009 [5];

Lemma 2.2 (Galerkin Orthogonality)**.** *The solution of the Galerkin Method satisfies*

$$a(u - u_h, v_h) = 0 \quad \forall v_h \in V_h \tag{2.28}$$

Proof. Since $\mathbb{V}_h \subset \mathbb{V}$, the exact solution satisfies the weak problem (2.2) for each element $v = v_h \in \mathbb{V}$, hence we have

$$a(u, v_h) = f(v_h) \quad \forall v_h \in \mathbb{V}. \tag{2.29}$$

By subtracting side by side (2.22) from (2.29), we obtain

$$a(u, v_h) - a(u_h, v_h) = 0 \quad \forall v_h \in \mathbb{V}, \tag{2.30}$$

from which, thanks to bilinearity of the form $a(\cdot, \cdot)$, the claim follows. □

A figure of the geometric interpretation of the Galerkin orthogonality is given in figure 2.2.

Remark 2.6. *Because of property* (2.29) *we could also state Cea's lemma, see e.g Lemma A.6 in the book by Hesthaven, Rozza and Stamm (HRS2016) [9].*

2.3 The Linear Lagrange Rectangle Element

The Finite Element solver used in this thesis is based on a commonly used Bilinear Lagrange Element and hence we briefly discuss the topic of the Bilinear Lagrange Rectangle Element. First

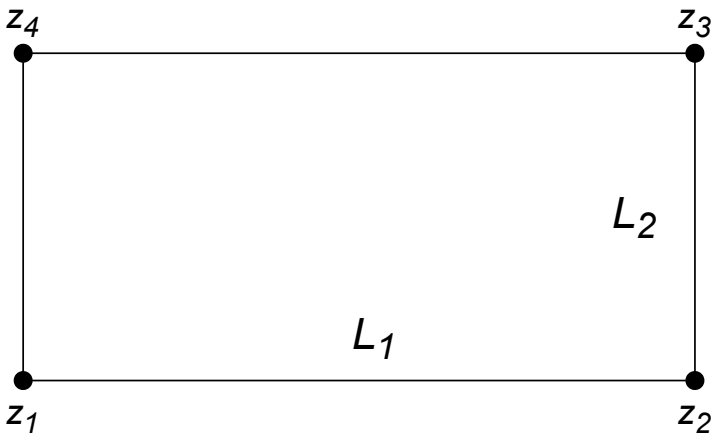


Figure 2.3. Bilinear Lagrange Rectangle Element — Here "•" indicates the nodal value evaluation at the point where the dot is located.

we state the definition of a finite element and two lemmas, then we present the example Bilinear Lagrange Rectangle Element from chapter 3 in *The Mathematical Theory of Finite Element Methods* by Brenner and Scott (BS2008) [10];

Definition 2.2. *Let*

(i) $K \subseteq \mathbb{R}^n$ *be a bounded closed set with nonempty interior and piecewise smooth boundary (the* ***element domain****).*

(ii) $\mathcal{P}$ *be a finite-dimensional space of functions on* K *(the space of* ***shape functions****) and*

(iii) $\mathcal{N} = \{N_1, N_2, \cdots, N_k\}$ *be a basis for* $\mathcal{P}'$ *(the set of* ***nodal variables****).*

Then $(K, \mathcal{P}, \mathcal{N})$ *is called a* ***finite element****.*

Lemma 2.3. *Let* $\mathcal{P}$ *be a* d*-dimensional vector space and let* $\{N_1, N_2, \cdots, N_d\}$ *be a subset of the dual space* $\mathcal{P}'$*. Then the following two statements are equivalent.*

(a) $\{N_1, N_2, \cdots, N_d\}$ *is a basis for* $\mathcal{P}'$*.*

(b) Given $v \in \mathcal{P}$ *with* $N_i(v) = 0$ *for* $i = 1, 2, \cdots, d$*, then* $v \equiv 0$*.*

Lemma 2.4. *Let* P *be a polynomial of degree* $d \geq 1$ *that vanishes on a hyperplane* L*. Then we can write* $P = LQ$*, where* Q *is a polynomilal of degree* $(d-1)$*.*

Example 2.1. *Let* K *be any rectangle,* $\mathcal{P} = \mathcal{Q}_1$*, and* $\mathcal{N}$ *as depicted in Fig. 2.3.*

Suppose that the polynomial $P \in \mathcal{Q}_1$ *vanishes at* z_1, z_2, z_3 *and* z_4*. The restriction of* P *to any side of the rectangle is a first-order polynomial of one variable. Therefore, we can write* $P = cL_1L_2$ *for some constant* c*. But*

$$0 = P(z_4) = cL_1(z_4)L_2(z_4) \Rightarrow c = 0, \tag{2.31}$$

since $L_1(z_4) \neq 0$ *and* $L_2(z_4) \neq 0$*. Thus* $P \equiv 0$*.*

The poof of the lemmas above can be found in chapter 2 of BS2008 [10]. We do here note that in example 2.1, we could write $P = cL_1L_2$ because of lemma 2.4. Furthermore, we verified lemma 2.2(iii) using lemma 2.3(b), i.e we prove that $\mathcal{N}$ determines $\mathcal{Q}_1$.

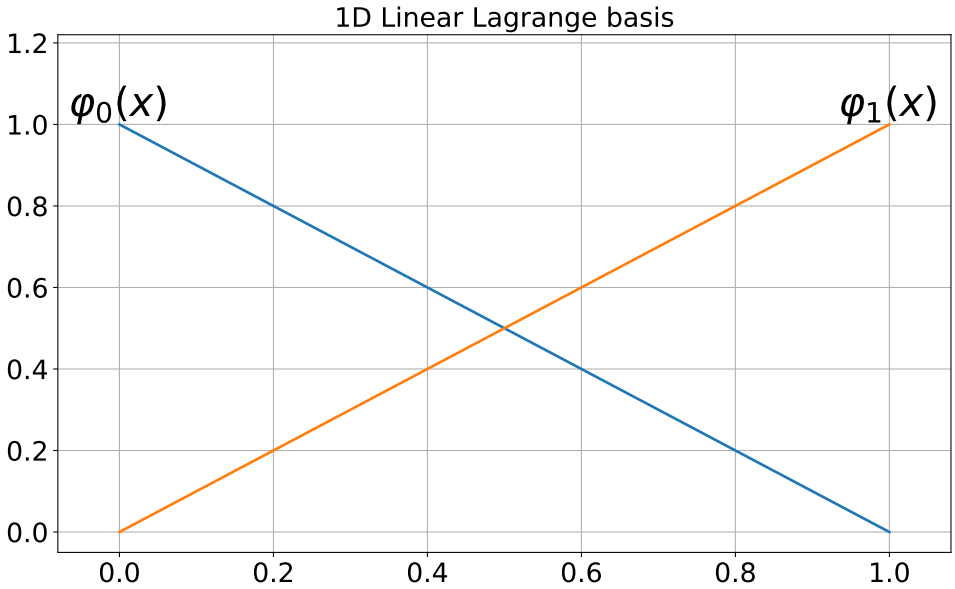


Figure 2.4. The two Lagrange basis functions on the line element $[0, 1]$.

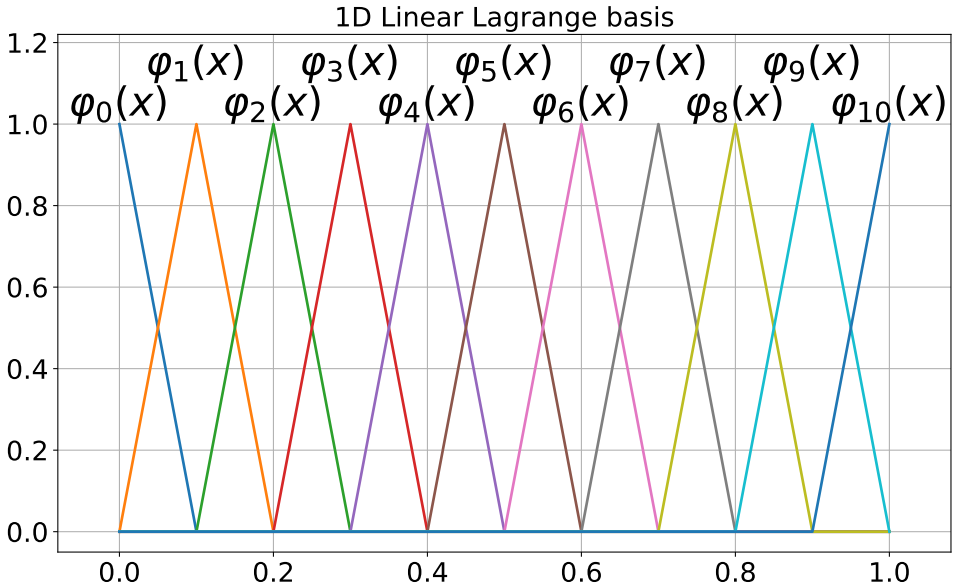


Figure 2.5. The Lagrange basis functions on the line $[0, 1]$ divided into 10 elements.

Now, to present a picture of the basis functions on the lines L_1 and L_2, we show two basis functions on the 1D line element $[0, 1]$ in figure 2.4. Figure 2.5 shows how the basis functions behave over multiple elements in a 1D example.

2.4 Reduced Basis Methods

As mentioned in chapter 3 of both HRS2016 and QMN2016 [9, 1] solving the high-fidelity problem (2.25) for any value of $\boldsymbol{\mu} \in \mathscr{P}$ entails severe computational cost. However, this can be mitigated by introducing a suitable reduced-order approximation. In this section we present the Reduced Basis (RB) Method as a way to approximate the high-fidelity solutions as described above. With the aim of exploiting the $\boldsymbol{\mu}$-dependence of the solution, the following observation is stated in section 3.3 of QMN2016;

> Given the discrete solution set (2.27) of the high-fidelity solutions generated as $\boldsymbol{\mu}$ varies over the parameter domain $\mathscr{P}$, we could expect that any $u_h(\boldsymbol{\mu})$ could be well approximated by linearly combining few elements of $\mathscr{M}_h$. This is true especially when $\mathscr{M}_h$ is low-dimensional and smooth. The idea behind RB methods is to generate an approximate solution to problem (2.25) belonging to a low-dimensional subspace $\mathbb{V}_N \subset \mathbb{V}_h$ of dimension $N \ll N_h$. The smaller N, the cheaper the reduced problem to solve.

For more on the low-dimensionality and smoothness of $\mathscr{M}_h$ we refer to chapter 5 in QMN2016 [1]. Furthermore, continuing from section 3.3 in QMN2016 we state precisely what setting a RB method entails;

1. the construction of a basis of $\mathbb{V}_N$. We start from a set of high-fidelity solutions

$$\{u_h(\boldsymbol{\mu}_1), \cdots, u_h(\boldsymbol{\mu}_N)\}, \tag{2.32}$$

that we call *snapshots*, corresponding to a set of N selected parameters

$$S_N = \{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_N\} \subset \mathscr{P}. \tag{2.33}$$

Then, we generate a set of N functions

$$\{\zeta_1, \cdots, \zeta_N\}, \tag{2.34}$$

called the *reduced basis*, by orthonormalization of the snapshots with respect to a suitable scalar product $(.,.)_N$, that is

$$(\zeta_m, \zeta_k)_N = \delta_{km}, \quad 1 \leq k, m \leq N. \tag{2.35}$$

Typically, $(.,.)_N = (.,.)_{\mathbb{V}}$, the $\mathbb{V}$-scalar product. The functions (2.34) are thefore called *reduced basis functions*, and generate the *reduced basis space*

$$\mathbb{V}_N = \text{span}\{\zeta_1, \cdots, \zeta_N\}. \tag{2.36}$$

The spaces $\{\mathbb{V}_N, N \geq 1\}$ are therfore nested, that is $\mathbb{V}_N \supset \mathbb{V}_{N-1}, N \geq 2$. By construction, the reduced basis functions are no longer solutions of the high- fidelity problem. However,

$$\mathbb{V}_N = \text{span}\{\zeta_1, \cdots, \zeta_N\} = \text{span}\{u_h(\boldsymbol{\mu}_1), \cdots, u_h(\boldsymbol{\mu}_N)\}. \tag{2.37}$$

2. the computation of the RB solution $u_N(\boldsymbol{\mu}) \in \mathbb{V}_N$, expressed as a linear combination of the reduced basis functions,

$$u_N(\boldsymbol{\mu}) = \sum_{m=1}^{N} u_N^{(m)}(\boldsymbol{\mu})\zeta_m \tag{2.38}$$

where $\boldsymbol{u}_N(\boldsymbol{\mu}) = \begin{bmatrix} u_N^{(1)}(\boldsymbol{\mu}) & \dots & u_N^{(N)}(\boldsymbol{\mu}) \end{bmatrix}^\top \in \mathbb{R}^N$ denotes the RB coefficients, also called the generalized coordinates, of $u_N(\boldsymbol{\mu})$ in the reduced basis;

3. the setup of a reduced problem for determining the unknown coefficients $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$.

2.4.1 Galerkin Reduced-order Approximation

A setup for determining the unknown coefficients $\boldsymbol{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ is the Galerkin RB method which in section 3.3.1 of QMN2016 [1] is defined as;

find $u_N(\boldsymbol{\mu}) \in \mathbb{V}_N$ such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = f(v_N; \boldsymbol{\mu}) \quad \forall v_N \in \mathbb{V}_N. \tag{2.39}$$

Remark 2.7. *Note that the well-posedness of the weak formulation above follows from a similar argument as for well-posedness of the weak formulation* (2.2) *in section 2.1.3.*

We now consider the Galerkin case stated in section 3.4.1 of QMN2016 [1];

Inserting (2.38) into (2.39) and then choosing $v_N = \zeta_n, 1 \leq n \leq N$, we obtain a set of N linear algebraic equations

$$\sum_{m=1}^{N} a(\zeta_m, \zeta_m; \boldsymbol{\mu}) u_N^{(m)}(\boldsymbol{\mu}) = f(\zeta_n; \boldsymbol{\mu}), \quad 1 \leq n \leq N. \tag{2.40}$$

We denote by $A_N(\boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$ the matrix with elements $(A_N(\boldsymbol{\mu}))_{nm} = a(\zeta_m, \zeta_m; \boldsymbol{\mu})$ and by $\boldsymbol{f}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$ the vector the vector with components $(\boldsymbol{f}_N(\boldsymbol{\mu}))_n = f(\zeta_n; \boldsymbol{\mu})$. Then, (2.40) is equivalent to the linear system

$$A_N(\boldsymbol{\mu}) \boldsymbol{u}_N(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu}). \tag{2.41}$$

Furthermore, to relate the reduced-order system to the high-fidelity system in section 2.2.1 we state the description from section 3.4.1 of QMN2016 [1];

Since the basis functions ζ_m belong to $\mathbb{V}_h$ we can compute the RB matrices and vectors from the corresponding high-fidelity ones. Indeed, expanding each RB basis function with respect to the basis functions $\{\varphi_i\}_{i=1}^{N_h}$,

$$\zeta_m = \sum_{i=1}^{N_h} \zeta_m^{(i)} \varphi_i, \quad 1 \leq m \leq N, \tag{2.42}$$

we can define the *transformation matrix* $V \in \mathbb{R}^{N_h \times N}$ whose columns contain the coefficients of the RB basis functions in (2.42), that is $V = \begin{bmatrix} \zeta_1 | & \cdots & | \zeta_N \end{bmatrix}$, or equivalently

$$(V)_{im} = \zeta_m^{(i)}, \quad 1 \leq m \leq N, 1 \leq i \leq N_h. \tag{2.43}$$

It follows that, for $1 \leq n, m \leq N$,

$$a_q(\zeta_m, \zeta_n) = \sum_{i=1}^{N_h} \sum_{j=1}^{N_h} \zeta_m^{(j)} a_q(\varphi_j, \varphi_i) \zeta_n^{(i)}, \quad f_q(\zeta_n) = \sum_{i=1}^{N_h} f_q(\varphi_i) \zeta_n^{(i)}. \tag{2.44}$$

Equivalently, in matrix form

$$A_N^{(q)} = V^\top A_h^{(q)} V, \quad \boldsymbol{f}_N^{(q)} = V^\top \boldsymbol{f}_h^{(q)} \tag{2.45}$$

where

$$(A_h^{(q)})_{ij} = a_q(\varphi_j, \varphi_i), \quad (\boldsymbol{f}_h^{(q)})_i = f_q(\varphi_i), \quad 1 \leq n \leq N. \tag{2.46}$$

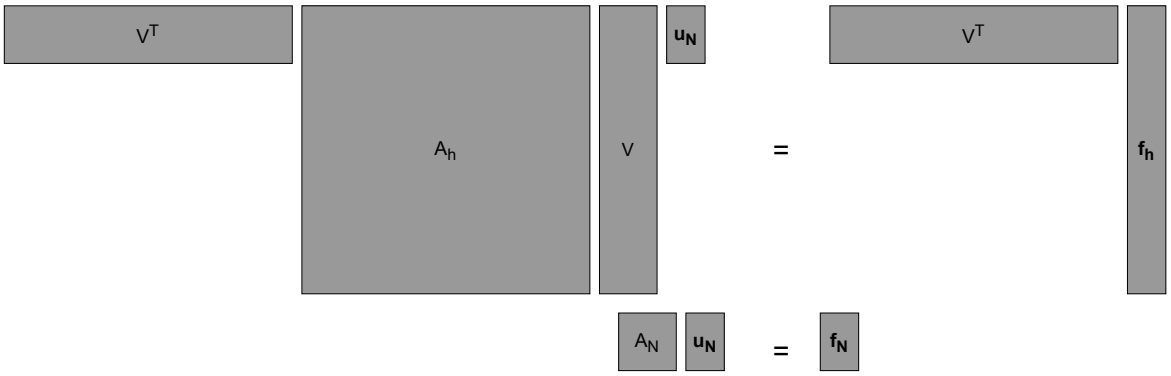


Figure 2.6. The linear systems of equations (2.41) and (2.45) with $A_h = A_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h \times n_h}$, $\boldsymbol{u}_N = \boldsymbol{u}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$, $\boldsymbol{f}_h = \boldsymbol{f}_h(\boldsymbol{\mu}) \in \mathbb{R}^{N_h}$, $V \in\in \mathbb{R}^{N_h \times N}$, $A_N = A_N(\boldsymbol{\mu}) \in \mathbb{R}^{N \times N}$, and $\boldsymbol{f}_N = \boldsymbol{f}_N(\boldsymbol{\mu}) \in \mathbb{R}^N$.

A figure of how the reduced-order system (2.41) relates to the high-fidelity system (2.25) by the *transformation matrix* V introduced above is given in figure 2.6.

Remark 2.8. *In this section we have said nothing about the construction of transformation matrix V. For this we refer to section 2.5.*

2.4.2 The Affine Parametric Dependence Assumption

The observant reader may have noted that the last quoted description of Quarteroni, Manzoni and Negri in section 2.4.1 mentions RB matrices and vectors, and not RB matrix and vector. This is because of the *affine parametric dependence* assumption described in e.g both HRS2016 and QMN2016 [9, 1]. We here provide the introduction to the *affine parametric dependence* assumption from section 3.4.1 of QMN2016;

> Matrix A_N is full, whereas the high-fidelity matrix A_h is (in general) sparse. However, since typically $N \ll N_h$, (2.41) is (in principle) much faster and less computationally intensive to solve than the original high-fidelity linear system (2.25). Unfortunately, the assembly of the reduced matrix $A_N(\boldsymbol{\mu})$ and vector $\boldsymbol{f}_N(\boldsymbol{\mu})$ still involves computations whose complexity depends on N_h.
>
> A key ingredient to overcome this drawback is to make the *affine parametric dependence* assumption. As anticipated in Chap.1, in this case we require both the parametric bilinear form a and the parametric linear form f to be *affine (or separable) with respect to the parameter* $\boldsymbol{\mu}$, that is
>
> $$a(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\boldsymbol{\mu}) a_q(w, v) \quad \forall v, w \in \mathbb{V}, \boldsymbol{\mu} \in \mathscr{P}, \tag{2.47}$$
>
> $$f(v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^{(q)} f_q(v) \quad \forall v \in \mathbb{V}, \boldsymbol{\mu} \in \mathscr{P}. \tag{2.48}$$
>
> Here $\theta_a^{(q)} : \mathscr{P} \to \mathbb{R}, q = 1, \cdots, Q_a$ and $\theta_f^{(q)} : \mathscr{P} \to \mathbb{R}, q = 1, \cdots, Q_f$ are $\boldsymbol{\mu}$-dependent functions, whereas $a_q : \mathbb{V} \times \mathbb{V} \to \mathbb{R}, f_q : \mathbb{V} \to \mathbb{R}$ are $\boldsymbol{\mu}$-independent forms.

This will lead to that the high-fidelity system (2.25) matrix and vector can be written as

$$A_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\boldsymbol{\mu}) A_h^{(q)}, \quad \boldsymbol{f}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^{(q)}(\boldsymbol{\mu}) \boldsymbol{f}_h^{(q)} \tag{2.49}$$

where

$$(A_h^{(q)})_{ij} = a_q(\varphi_j, \varphi_i), \quad (\boldsymbol{f}_h^{(q)})_i = f_q(\varphi_i). \tag{2.50}$$

Similarly for the reduced-order system (2.41) we can write

$$A_N(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \theta_a^{(q)}(\boldsymbol{\mu}) A_N^{(q)}, \quad \boldsymbol{f}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \theta_f^{(q)}(\boldsymbol{\mu}) \boldsymbol{f}_N^{(q)} \tag{2.51}$$

where

$$(A_N^q)_{nm} = a_{(q)}(\zeta_m, \zeta_n), \quad (\boldsymbol{f}_N^{(q)})_i = f_q(\zeta_n). \tag{2.52}$$

2.4.3 Error Computations

Picking up from where we left in section 2.4.1, we now want to look at the error computations between the high-fidelity and RB solution. We denote by $e_h(\boldsymbol{\mu}) = u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu}) \in \mathbb{V}_h$ this error, as defined in section 3.6 of QMN2016 [1], and note that by the triangle inequality we have that

$$\|e_h(\boldsymbol{\mu})\| = \|u(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\| \leq \|u(\boldsymbol{\mu}) - u_h(\boldsymbol{\mu})\| + \|u_h(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})\|. \tag{2.53}$$

Hence if for desired accuracy, the grid parameter h is small enough, i.e. chose h such that $\|u(\boldsymbol{\mu}) - u_h(\boldsymbol{\mu})\| < \varepsilon_{\text{tol}}$, then the accuracy of how good the RB solution approximates the exact solution depends on how good the RB solution approximates the high-fidelity solution.

Now, using (2.38) and (2.42) we can write $u_N(\boldsymbol{\mu})$ as

$$u_N(\boldsymbol{\mu}) = \sum_{m=1}^{N} \sum_{j=1}^{N_h} u_N^{(m)}(\boldsymbol{\mu}) \zeta_m^{(j)} \varphi_j = \sum_{j=1}^{N_h} \sum_{m=1}^{N} u_N^{(m)}(\boldsymbol{\mu}) \zeta_m^{(j)} \varphi_j, \tag{2.54}$$

which together with (2.23)gives $e_h(\boldsymbol{\mu})$ as

$$e_h(\boldsymbol{\mu}) = \sum_{j=1}^{N_h} \left(u_h^{(j)}(\boldsymbol{\mu}) - \sum_{m=1}^{N} \zeta_m^{(j)} u_N^{(m)}(\boldsymbol{\mu}) \right) \varphi_j. \tag{2.55}$$

This leads us to the to definition of the discrete error between the recovered RB solution and high-fidelity solution in section 3.7.1 in QMN2016 [1];

$$\boldsymbol{e}_h(\boldsymbol{\mu}) = \boldsymbol{u}_h(\boldsymbol{\mu}) - V \boldsymbol{u}_N(\boldsymbol{\mu}), \tag{2.56}$$

Now defining the vector $\boldsymbol{\Phi} = \begin{bmatrix} \varphi_1 & \dots & \varphi_{N_h} \end{bmatrix}^\top$ we have $e_h(\boldsymbol{\mu}) = \boldsymbol{\Phi}^\top \boldsymbol{e}_h(\boldsymbol{\mu})$, and can write the X-norms induced by some $\mathbb{V}$-inner product as

$$\begin{aligned} \|e_h\|_X^2 = (e_h, e_h)_X = (\boldsymbol{\Phi}^\top \boldsymbol{e}_h(\boldsymbol{\mu}), \boldsymbol{\Phi}^\top \boldsymbol{e}_h(\boldsymbol{\mu}))_X = \\ \boldsymbol{e}_h(\boldsymbol{\mu})^\top (\boldsymbol{\Phi}, \boldsymbol{\Phi}^\top)_X \boldsymbol{e}_h(\boldsymbol{\mu}) = \boldsymbol{e}_h(\boldsymbol{\mu})^\top X_h \boldsymbol{e}_h(\boldsymbol{\mu}), \end{aligned} \tag{2.57}$$

where $X_h \in \mathbb{R}^{N_h \times N_h}$ is a symmetric positive definite matrix defined by the respective $\mathbb{V}$-scalar product, which in case of the energy norm form section 2.1.5 is $X_h = A_h(\boldsymbol{\mu})$.

2.4.4 The Formal Obtaintion of the Galerkin RB Problem

Inspired by the computations in section 2.4.3 we subtract the RB weak formulation (2.39) from the high-fidelity one (2.2) and we get the error representation defined in section 3.6.1 of QMN2016 [1];

$$a(e_h(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}); v; \boldsymbol{\mu}) \quad \forall v \in \mathbb{V}_h, \tag{2.58}$$

Continuing form the same section we set

$$r(v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}) - a(u_N(\boldsymbol{\mu}), v : \boldsymbol{\mu}) \quad \forall v \in \mathbb{V}_h, \tag{2.59}$$

as the residual, between the RB and high-fidelity solutions. This gives rise to the discrete residual defined in section 4.1.2 of QMN2016;

$$\boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu}) = \boldsymbol{f}_h(\boldsymbol{\mu}) - A_h(\boldsymbol{\mu}) V \boldsymbol{u}_N(\boldsymbol{\mu}), \tag{2.60}$$

Following the mentioned section of QMN2016 we state the lemma providing the main algebraic connections between the RB and high-fidelity Garlerkin approximations and the formal oblation of the Galerkin RB problem (2.41) for a given matrix V of reduced bases;

Lemma 2.5. *The following algebraic relations hold:*

$$A_h(\boldsymbol{\mu}) \boldsymbol{e}_h(\boldsymbol{\mu}) = \boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu}) \tag{2.61}$$

$$V^\top A_h(\boldsymbol{\mu}) \boldsymbol{u}_h(\boldsymbol{\mu}) = \boldsymbol{f}_N(\boldsymbol{\mu}) \tag{2.62}$$

$$V^\top \boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu}) = \boldsymbol{0}. \tag{2.63}$$

In summary, for a given matrix V of reduced bases, the Galerkin RB problem (2.41) can be formally obtained as follows:

Galerkin Reduced Basis (G-RB) problem

1. consider the Galerkin high-fidelity problem (2.25);
2. set $\boldsymbol{u}_h(\boldsymbol{\mu}) = V \boldsymbol{u}_N(\boldsymbol{\mu}) + \boldsymbol{e}_h(\boldsymbol{\mu})$, where $\boldsymbol{u}_N \in \mathbb{R}^N$ has to be determined and the error $\boldsymbol{e}_h$ is the difference between $\boldsymbol{u}_h$ and $V \boldsymbol{u}_N$;
3. left multiply (2.25) by $V^\top$ to obtain

   $$V^\top A_h(\boldsymbol{\mu}) V \boldsymbol{u}_N(\boldsymbol{\mu}) - V^\top \boldsymbol{f}_h(\boldsymbol{\mu}) = -V^\top A_h(\boldsymbol{\mu}) \boldsymbol{e}_h,$$

   that is

   $$V^\top A_h(\boldsymbol{\mu}) V \boldsymbol{u}_N(\boldsymbol{\mu}) - V^\top \boldsymbol{f}_h(\boldsymbol{\mu}) = -V^\top \boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu});$$

4. require $\boldsymbol{u}_N$ to satisfy $V^\top \boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu}) = \boldsymbol{0}$, or equivalently

   $$V^\top A_h(\boldsymbol{\mu}) V \boldsymbol{u}_N(\boldsymbol{\mu}) = V^\top \boldsymbol{f}_h(\boldsymbol{\mu}). \tag{2.64}$$

This coincides with what we found in section 2.4.1, where the relationship between (2.41) and (2.64) is shown in figure 2.6.

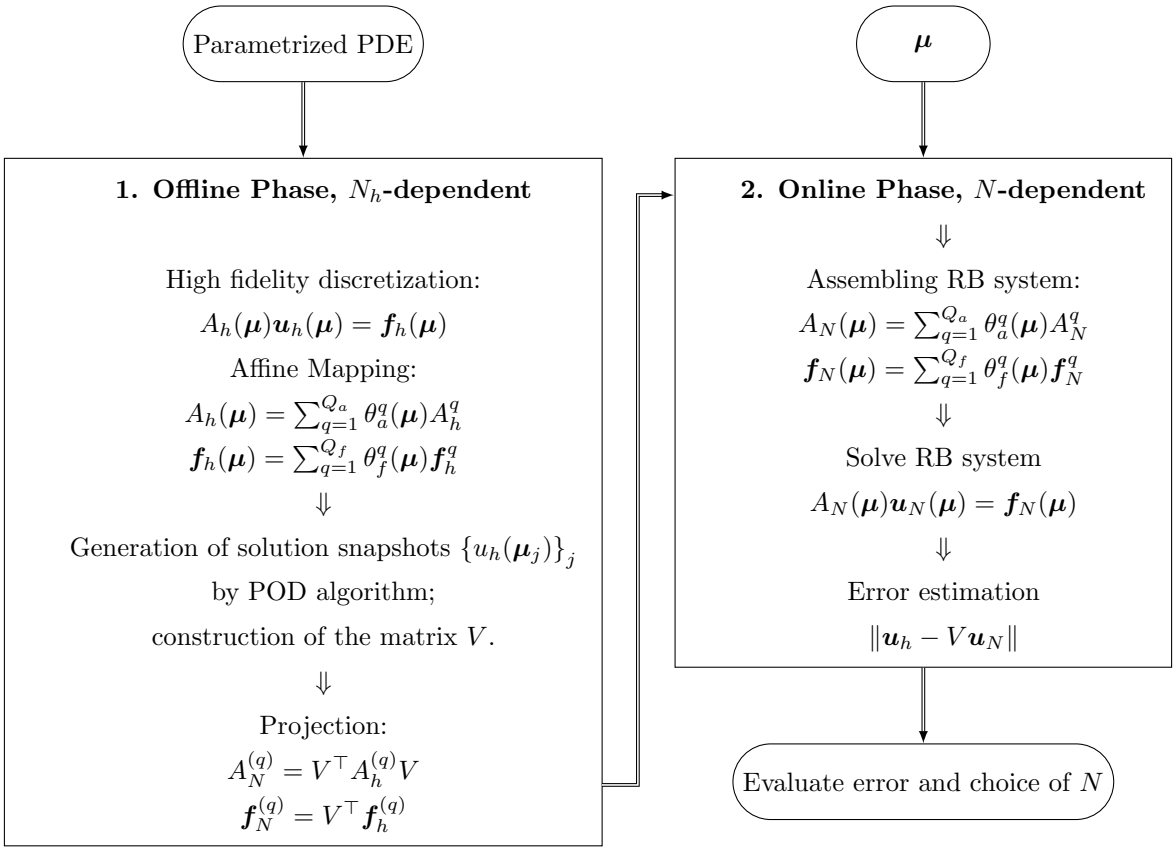


Figure 2.7. The affine reduced basis (RB) workflow at a glance.

2.4.5 The Offline and Online Phases

Going back to section 2.4.2 we see that from a computational standpoint, we can take advantage of the affine parametric dependence property by splitting the assembly of the reduced matrices and vectors in two different phases. In the first phase, we perform offline once and for all, doing the computation of all the N_h-dependent and $\boldsymbol{\mu}$-independent matrices $A_h^{(q)}$ and vectors $\boldsymbol{f}_h^{(q)}$. In the second phase, to be performed online for any given value of $\boldsymbol{\mu} \in \mathscr{P}$, we assemble and solve the RB system. Which has a cost depending only on N, as stated in section 3.5 of QMN2016 [1]. For further reading, see the computational complexity section 3.5 of QMN2016 for the Galerkin RB case.

Figure 2.7, inspired by the figure in chapter 1. of QMN2016 [1], shows the algebraic workflow for the affine RB workflow. The process Proper Orthogonal Decomposition (POD) algorithm for constructing V and choosing N has not been presented yet, but will be introduced in the following and discussed in further detail in section 2.5.

2.5 Proper Orthogonal Decomposition

There are multiple ways to create the reduced basis from which the reduced-order model can be constructed. In QMN2016 [1] both the *Proper Orthogonal Decomposition* (POD) and the Greedy methods are discussed. For this project, POD was chosen, which is discussed in detail in section 6.3 of QMN2016. From that section we note that POD is a numerical technique for compressing and approximating a high-dimensional data set by an orthonormal basis. For the finite element case, this means that the original variables u_h, are transformed into a new set of

uncorrelated variables, called POD modes or principal components, where the first few modes ideally retain most of the energy present in all of the original variables. This means that POD is *Principal Component Analysis*(PCA) used in mechanical engineering. For more on PCA we refer the reader to e.g [11, 12].

2.5.1 Singular Value Decomposition

Before we can apply POD to parametric PDEs the concept of singular value decomposition (SVD) is needed. Therefore we present the singular value decomposition as stated in section 6.1 of QMN2016 [1];

if $A \in \mathbb{R}^{m\times n}$ is a real matrix, there exist two orthogonal matrices.

$$U = \left[\ \boldsymbol{\zeta}_1| \quad \dots \quad |\boldsymbol{\zeta}_m\ \right] \in \mathbb{R}^{m\times m}, \quad Z = \left[\ \boldsymbol{\psi}_1| \quad \dots \quad |\boldsymbol{\psi}_n\ \right] \in \mathbb{R}^{n\times n} \tag{2.65}$$

such that

$$A = U\Sigma Z^\top, \quad \text{with} \quad \Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{m\times n} \tag{2.66}$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, for $p = \min(m, n)$.

The matrix factorization (2.66) is called singular value decomposition (SVD) of A and the numbers $\sigma_i = \sigma_i(A)$ are called singular values of A. $\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_m$ are called left singular vectors of A, whereas $\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_m$ right singular vectors of A, as

$$A\boldsymbol{\psi}_i = \sigma_i\boldsymbol{\zeta}_i, \quad A^T\boldsymbol{\zeta}_j = \sigma_j\boldsymbol{\psi}_j, \quad i, j = 1, \cdots, n.$$

Now, since the singular values of a matrix are related to its norm, we state the definition of the two relevant norms for matrices from section of QMN2016 [1];

$$\|A\|_2 = \sigma_{max} = \max_{i=1,\dots,p} \sigma_i, \quad \|A\|_{\text{F}} = \sqrt{\sum_{i=1}^{p} \sigma_i^2}. \tag{2.67}$$

where the *Frobenius norm* is defined as

$$\|A\|_{\text{F}} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}. \tag{2.68}$$

For the complex version of the SVD, more on matrix norms and a proof for the 2-norm in (2.67) see section 1.9 and 1.11 in [7]. Moreover, if $A \in \mathbb{R}^{m\times n}$ has r positive singular values, then $\operatorname{rank}(A) = r$. As stated in section 6.1.1 in QMN2016 [1] a particular feature of SVD arises;

if $A \in \mathbb{R}^{m\times n}$ has rank equal to r, then it can be written as the sum of r rank-1 matrices

$$A = \sum_{i=1}^{r} \sigma_i\boldsymbol{\zeta}_i\boldsymbol{\psi}_i^\top \tag{2.69}$$

Formula (2.69) is very useful, since it allows us to compute low-rank approximations of a matrix, which is thanks to properties (2.67). This leads to Schmidt-Eckart-Young Theorem and a similar result for the 2-norm in section 6.1.1 of QMN2016 [1];

Theorem 2.1 (Schmidt-Eckart-Young)**.** *Given a matrix $A \in \mathbb{R}^{m\times n}$ of rank r, the matrix*

$$A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{\zeta}_i \boldsymbol{\psi}_i^\top, \quad 0 \leq k \leq r, \tag{2.70}$$

satisfies the optimality property

$$\|A - A_k\|_{\mathrm{F}} = \min_{\substack{B \in \mathbb{R}^{m\times n} \\ \mathrm{rank}(B) \leq k}} \|A - B\|_{\mathrm{F}} = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}. \tag{2.71}$$

A similar result holds by considering the 2-norm instead of the Frobenius norm: for any $0 < k \leq r$, the matrix A_k defined in (2.70) is also such that

$$\|A - A_k\|_2 = \min_{\substack{B \in \mathbb{R}^{m\times n} \\ \mathrm{rank}(B) \leq k}} \|A - B\|_2 = \sigma_{k+1}. \tag{2.72}$$

For a proof of (2.71) Quarteroni, Sacco and Saleri refer to [13], while for the proof of the optimality with respect to the 2-norm they refer to [14, 15]. Using the theorem and result from above we see that there is no better rank-k approximation with respect to the energy of the system.

2.5.2 Orthogonal Projection Operators

We now want to provide a brief note on Orthogonal Projection Operators. To learn more about orthogonal projectors please see e.g section 1.12 in [16]. From (2.35) and (2.43) we see that columns of V are orthonormal with respect to a $\mathbb{V}$-scalar product, i.e $V^\top X_h V = I_N$, where $X_h \in \mathbb{R}^{N_h \times N_h}$ is a symmetric positive definite matrix defined by the respective $\mathbb{V}$-scalar product, which in case of the energy norm is $X_h = A_h(\boldsymbol{\mu})$.

Now, we for simplicity assume that $V^\top V = I_N$ and state proposition 4.1 from section 4.2.4 in QMN2016 [1];

Proposition 2.1. *The following results hold:*

1. *the matrix $P = VV^\top \in \mathbb{R}^{N_h \times N_h}$ is an orthogonal projector from the whole space $\mathbb{R}^{N_h}$ onto the subspace $\mathbb{V}_N$;*
2. *the matrix $I - VV^\top \in \mathbb{R}^{N_h \times N_h}$ is a projector from the whole space $\mathbb{R}^{N_h}$ onto the subspace $\mathbb{V}_N^\perp$, the subspace of $\mathbb{R}^{N_h}$ orthogonal to $\mathbb{V}_N$;*
3. *the residual $\boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu})$ satisfies*

$$P\boldsymbol{r}_h(\boldsymbol{u}_N; \boldsymbol{\mu}) = \boldsymbol{0}, \tag{2.73}$$

that is, it belongs to the orthogonal space $\mathbb{V}_N^\perp$.

Remark 2.9. *Note that the proposition also holds in the case when $V^\top X_h V = I_N$, by defining $Y = X_h^{1/2} V$ because this gives $Y^\top Y = I_N$.*

2.5.3 POD for Parametrized Problems

The starting point for the POD approach is a parameter sample set $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{n_s}\} \subset \mathscr{P}$ for which high-fidelity solutions $\{u_h(\boldsymbol{\mu}_1), \ldots, u_h(\boldsymbol{\mu}_{n_s})\}$ are calculated. Therefore we define the snapshot matrix $S \in \mathbb{R}^{N_h \times n_s}$ as

$$S = \left[\begin{array}{c|ccc} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{n_s} \end{array} \right], \tag{2.74}$$

and utilizing the SVD in section 2.5.1 we get the statement in section 6.3.1 of QMN2016 [1];

> According to (2.66), the SVD decomposition of S reads
>
> $$S = U\Sigma Z^\top, \tag{2.75}$$
>
> where $U = \left[\begin{array}{c|ccc} \boldsymbol{\zeta}_1 & \cdots & \boldsymbol{\zeta}_m \end{array} \right] \in \mathbb{R}^{N_h \times N_h}$ and $Z = \left[\begin{array}{c|ccc} \boldsymbol{\psi}_1 & \cdots & \boldsymbol{\psi}_n \end{array} \right] \in \mathbb{R}^{n_s \times n_s}$ are orthogonal matrices, and $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{N_h \times n_s}$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. Here $r \leq \min(N_h, n_s)$ denotes the rank of S, which is strictly smaller than n_s if the snapshot vectors are not all linearly independent. Then, we can write
>
> $$S\boldsymbol{\psi}_i = \sigma_i \boldsymbol{\zeta}_i, \quad S^T\boldsymbol{\zeta}_i = \sigma_i \boldsymbol{\psi}_i, \quad i = 1, \cdots, r. \tag{2.76}$$
>
> or, equivalently
>
> $$S^\top S\boldsymbol{\psi}_i = \sigma_i^2 \boldsymbol{\psi}_i, \quad SS^T\boldsymbol{\zeta}_i = \sigma_i^2 \boldsymbol{\zeta}_i, \quad i = 1, \cdots, r. \tag{2.77}$$
>
> i.e.$\sigma_i^2\boldsymbol{\zeta}_i, \quad i = 1, \cdots, r$ are the nonzero eigenvalues of the matrix $S^\top S$ (and also of $SS^\top$), listed in nondecreasing order.

Now, by defining the *correlation matrix* $C = S^\top S \in \mathbb{R}^{n_s \times n_s}$ by its elements as

$$C_{ij} = \boldsymbol{u}_i^\top \boldsymbol{u}_j, \quad i, j \leq i, j \leq n_s, \tag{2.78}$$

we provide the next statement in section 6.3.1 QMN2016 [1];

> For any $N \leq n_s$, the POD basis $\mathbb{V} \in \mathbb{R}^{N_h \times N}$ of dimension N is defined as the set of the first N left singular vectors $\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_N$ of U or, equivalently, the set of vectors
>
> $$\boldsymbol{\zeta}_j = \frac{1}{\sigma_j} S\boldsymbol{\psi}_j, \quad 1 \leq j \leq N \tag{2.79}$$
>
> obtained from the first N eigenvectors $\boldsymbol{\psi}_1, \cdots, \boldsymbol{\psi}_N$ of the correlation matrix C.

From this we have $V = \left[\begin{array}{ccc} \boldsymbol{\zeta}_1 & \cdots & \boldsymbol{\zeta}_N \end{array} \right]$ as described in point one of what precisely setting a RB method entails in section 2.4.

Inspired our discussion of orthogonal projectors in by section 2.5.2 and that by construction the POD basis is orthonormal, we define

$$P_{\mathbb{W}}\boldsymbol{x} = \sum_{j=1}^{N} (\boldsymbol{x}, \boldsymbol{w}_j)_2 \boldsymbol{w}_j = WW^\top \boldsymbol{x} \tag{2.80}$$

as a projector onto the subspace spanned by $W = \left[\begin{array}{c|ccc} \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_N \end{array} \right] \in \mathbb{R}^{N_h \times N}$, and state the proposition given in section 6.3.1 QMN2016 [1];

> **Proposition 2.2.** *Let* $\mathscr{V}_N = \left\{ W \in \mathbb{R}^{N_h \times N} : W^\top W = I_N \right\}$ *be the set of all* N*-dimensional orthonormal bases. Then,*
>
> $$\sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - VV^\top \boldsymbol{u}_i \right\|_2^2 = \min_{W \in \mathscr{V}_N} \sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - WW^\top \boldsymbol{u}_i \right\|_2^2 = \sum_{i=N+1}^{r} \sigma_i^2. \tag{2.81}$$

For the proof of the proposition we refer to section 6.3.1 QMN2016, but note that it is by Theorem 2.1 in section 2.5.1. Furthermore, this means that the POD basis minimizes the sum of the squares of the errors between each snapshot vector $\boldsymbol{u}_i$ and its projection onto the subspace spanned by W. It also follows that the error in the POD basis is equal to the sum of the squares of the singular values corresponding to the neglected POD modes. This gives us a suitable criterion to select the minimal POD dimension $N \leq r$ such that the projection error is smaller than a desired tolerance ε_{POD}, as suggested in section 6.3.1 QMN2016 [1];

It is sufficient to choose N as the smallest integer such that

$$I(N) = \frac{\sum_{i=1}^{N} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \geq 1 - \varepsilon_{\text{POD}}^2, \tag{2.82}$$

that is the energy retained by the last $r - N$ modes is equal or smaller than ε_{POD}.

Remark 2.10. *$I(N)$ represents the percentage of energy of the snapshots captured by the first N POD modes, and it is referred to as the relative information content of the POD basis.*

Lastly, we here also state one remark by Quarteroni, Sacco and Saleri in section 6.3.1 QMN2016 [1];

> Computing of the POD basis by solving an eigenvalue problem for the correlation matrix C yields inaccurate results for the modes associated to small singular values. This is due to the roundoff errors introduced while constructing C and the fact that $\kappa(C) = (\kappa(S))^2$. In such cases it is recommended to construct the POD basis by means of stable algorithms for the computation of the SVD.

This is important in algorithm 1, for computing the POD basis described in section 6.3.1 of QMN2016 [1].

Algorithm 1 The algorithm for computing the POD basis described in section 6.3.1 of *Reduced Basis Methods for Partial Differential Equations* by Quarteroni, Manzoni and Negri (QMN2016) [1].

function $V = POD(S, \varepsilon_{\text{POD}})$
  if $n_s \leq N_h$ **then**
    from the correlation matrix $C = S^\top S$
    solve the eigenvalue problem $C\boldsymbol{\psi}_i = \sigma_i^2 \boldsymbol{\psi}_i, \quad i = 1, \cdots, r$
    set $\boldsymbol{\zeta}_i = \frac{1}{\sigma_i} S\boldsymbol{\psi}_i$
  else
    form the matrix $K = SS^\top$
    solve the eigenvalue problem $K\boldsymbol{\zeta}_i = \sigma_i^2 \boldsymbol{\zeta}_i, \quad i = 1, \cdots, r$
  end if
  define N as the minimum integer such that $I(N) \geq 1 - \varepsilon_{\text{POD}}^2$
  $V = \left[\ \boldsymbol{\zeta}_1 | \quad \cdots \quad | \boldsymbol{\zeta}_N \ \right]$
end function

2.5.4 POD with Respect to Energy Inner Product

Since the snapshot functions $u_h(\boldsymbol{\mu}_i)$ belong to the space $\mathbb{V}_h \subset \mathbb{V}$, it becomes natural to seek a POD basis minimizing the norm defined by a inner product of $\mathbb{V}$, which is usually the energy

norm and energy inner product which we defined in section 2.1.5. In particular as stated in section 6.3.2 in QMN2016 [1];

We seek a basis $W \in \mathcal{V}_N^{X_h}$, with

$$\mathcal{V}_N^{X_h} = \left\{ W \in \mathbb{R}^{N_h \times N} : W^\top X_h W = I_N \right\},$$

which minimizes the squares of the X_h-norm of the error between each snapshot vector $\boldsymbol{u}_i$ and its X_h-orthogonal projection onto the subspace spanned by W, i.e.

$$\min_{W \in \mathcal{V}_N^{X_h}} \sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - P_W^{X_h} \boldsymbol{u}_i \right\|_{X_h}^2 . \tag{2.83}$$

Therefore as in section 2.5.3 and our discussion of orthogonal projectors in by section 2.5.2 we define the X_h-orthogonal projector

$$P_W^{X_h} \boldsymbol{x} = \sum_{j=1}^{N} (\boldsymbol{x}, \boldsymbol{w}_j)_{X_h} \boldsymbol{w}_j = W W^\top X_h \boldsymbol{x}, \tag{2.84}$$

remembering that for the energy norm and energy inner product $X_h = A_h(\boldsymbol{\mu})$ as stated in section 2.4.3. Then by (2.84) we have

$$\begin{aligned} &\sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - P_W^{X_h} \boldsymbol{u}_i \right\|_{X_h}^2 = \sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - W W^\top X_h \boldsymbol{u}_i \right\|_{X_h}^2 \\ &= \sum_{i=1}^{n_s} \left\| X_h^{1/2} \boldsymbol{u}_i - X_h^{1/2} W W^\top X_h \boldsymbol{u}_i \right\|_2^2 = \left\| X_h^{1/2} S - X_h^{1/2} W W X_h S \right\|_{\mathrm{F}}^2, \end{aligned} \tag{2.85}$$

which leads us to the proposition deduced from Schmidt-Eckart-Young Theorem 2.1 and Proposition 2.2 stated in section 6.3.2 in QMN2016 [1];

Proposition 2.3. *Let* $S = \begin{bmatrix} \boldsymbol{u}_1 | & \cdots & | \boldsymbol{u}_{n_s} \end{bmatrix} \in \mathbb{R}^{N_h \times n_s}$ *be a given matrix of rank* $r \leq \min(N_h, n_s)$, $X_h \in \mathbb{R}^{N_h \times N_h}$ *a symmetric positive definite matrix,* $\tilde{S} = X_h^{1/2} S$ *and* $\tilde{S} = \tilde{U} \Sigma \tilde{Z}^\top$ *its singular value decomposition, where*

$$\tilde{U} = \begin{bmatrix} \tilde{\boldsymbol{\zeta}}_1 | & \cdots & | \tilde{\boldsymbol{\zeta}}_{N_h} \end{bmatrix} \in \mathbb{R}^{N_h \times N_h}, \quad \tilde{Z} = \begin{bmatrix} \tilde{\boldsymbol{\psi}}_1 | & \cdots & | \tilde{\boldsymbol{\psi}}_{n_s} \end{bmatrix} \in \mathbb{R}^{n_s \times n_s} \tag{2.86}$$

are orthogonal matrices and $\Sigma = \mathrm{diag}(\sigma_1, \cdots, \sigma_r) \in \mathbb{R}^{N_h \times n_s}$ *with* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$. *Then, for* $N \leq r$, *the POD basis* $V = \begin{bmatrix} X_h^{-1/2} \tilde{\boldsymbol{\zeta}}_1 | & \cdots & | X_h^{-1/2} \tilde{\boldsymbol{\zeta}}_N \end{bmatrix}$ *is such that*

$$\sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - V V^\top X_h \boldsymbol{u}_i \right\|_{X_h}^2 = \min_{W \in \mathcal{V}_N^{X_h}} \sum_{i=1}^{n_s} \left\| \boldsymbol{u}_i - W W^\top X_h \boldsymbol{u}_i \right\|_{X_h}^2 = \sum_{i=N+1}^{r} \sigma_i^2. \tag{2.87}$$

Now, since for $n_s < N_h$ we have

$$\tilde{S}^\top \tilde{S} \tilde{\boldsymbol{\psi}}_i = \sigma_i^2 \tilde{\boldsymbol{\psi}}_i, \quad i = 1, \cdots, r, \tag{2.88}$$

we can obtain the POD basis without forming the matrix $X_h^{1/2}$, as described in section 6.3.2 in QMN2016 [1];

We first compute the correlation matrix $\tilde{C} = \tilde{S}^\top \tilde{S} = S^\top X_h S$ and its first N eigenvectors $\tilde{\boldsymbol{\psi}}_1, \cdots, \tilde{\boldsymbol{\psi}}_N$. Then, we define the POD basis as $V = \begin{bmatrix} X_h^{-1/2}\boldsymbol{\zeta}_1 | & \cdots & | X_h^{-1/2}\boldsymbol{\zeta}_N \end{bmatrix}$, where

$$\boldsymbol{\zeta}_i = X_h^{-1/2}\tilde{\boldsymbol{\zeta}}_i = X_h^{-1/2}\frac{1}{\sigma_i}\tilde{S}\tilde{\boldsymbol{\psi}}_i = \frac{1}{\sigma_i}S\tilde{\boldsymbol{\psi}}_i. \tag{2.89}$$

Using this we present algorithm 2, for computing the POD basis with respect to the X_h norm in section 6.3.2 in QMN2016 [1].

Remark 2.11. *For algorithm 2, we make the same note as in section 2.5.3 about the computation of the POD basis. i.e that solving the eigenvalue problem for the correlation matrix $\tilde{C}$ yields inaccurate results for the modes associated with small singular values.*

Algorithm 2 The algorithm for computing the POD basis with respect to the X_h norm in section 6.3.2 of *Reduced Basis Methods for Partial Differential Equations* by Quarteroni, Manzoni and Negri (QMN2016) [1].

function $V = POD(S, X_h, \varepsilon_{\text{POD}})$
  if $n_s \leq N_h$ **then**
    from the correlation matrix $\tilde{C} = S^\top X_h S$
    solve the eigenvalue problem $\tilde{C}\tilde{\boldsymbol{\psi}}_i = \sigma_i^2\tilde{\boldsymbol{\psi}}_i, \quad i = 1, \cdots, r$
    set $\boldsymbol{\zeta}_i = \frac{1}{\sigma_i}S\tilde{\boldsymbol{\psi}}_i$
  else
    form the matrix $\tilde{K} = X_h^{1/2}SS^\top X_h^{1/2}$
    solve the eigenvalue problem $\tilde{K}\tilde{\boldsymbol{\zeta}}_i = \sigma_i^2\tilde{\boldsymbol{\zeta}}_i, \quad i = 1, \cdots, r$ set $\boldsymbol{\zeta}_i = X_h^{-1/2}\tilde{\boldsymbol{\zeta}}_i$
  end if
  define N as the minimum integer such that $I(N) \geq 1 - \varepsilon_{\text{POD}}^2$
  $V = \begin{bmatrix} X_h^{-1/2}\boldsymbol{\zeta}_1 | & \cdots & | X_h^{-1/2}\boldsymbol{\zeta}_N \end{bmatrix}$
end function

2.6 Matrix Least Squares

In section 2.4.2 we introduced the affine parametric dependence assumption that gives rise to the affine notation of the high-fidelity system (2.49),

$$A_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_a}\theta_a^{(q)}(\boldsymbol{\mu})A_h^{(q)}, \quad \boldsymbol{f}_h(\boldsymbol{\mu}) = \sum_{q=1}^{Q_f}\theta_f^{(q)}(\boldsymbol{\mu})\boldsymbol{f}_h^{(q)}. \tag{2.90}$$

However, this only holds if the problem is affine. If the problem is not affine we do not have this splitting, i.e.

$$A_h(\boldsymbol{\mu}) \neq \sum_{q=1}^{Q_a}\theta_a^{(q)}(\boldsymbol{\mu})A_h^{(q)}, \quad \boldsymbol{f}_h(\boldsymbol{\mu}) \neq \sum_{q=1}^{Q_f}\theta_f^{(q)}(\boldsymbol{\mu})\boldsymbol{f}_h^{(q)}. \tag{2.91}$$

Since the affine mapping does not exists, both the projection and the assembly of the reduced-order system will fail, i.e. the workflow in figure 2.7 breaks down.

In this section we present our suggested technique to for solving non-affine problems where we do not have an affine mapping. Our technique consists of turning the "mapping" around to get

a *fitting problem*, i.e.

$$\begin{aligned}\sum_{q=0}^{Q} g_q(\boldsymbol{\mu}_k) A_h^{(q)} &:= A_h(\boldsymbol{\mu}_k), \quad 1 \leq k \leq n, \\ \sum_{q=0}^{Q} g_q(\boldsymbol{\mu}_k) \boldsymbol{f}_h^{(q)} &:= \boldsymbol{f}_h(\boldsymbol{\mu}_k), \quad 1 \leq k \leq n,\end{aligned} \tag{2.92}$$

where q starts at 0 because we by $g_0(\boldsymbol{\mu})$ denote the constant function 1. The goal with this is to find an approximate affine decomposition for the matrices $A(\boldsymbol{\mu}_k)$ and vectors $\boldsymbol{f}_h(\boldsymbol{\mu}_k)$ to be used in reduced order methods.

2.6.1 The Matrix Least Square Problem

Generalizing the *fitting problem* (2.92) we get the matrix problem

$$\sum_{q=0}^{Q} g_q(\boldsymbol{\mu}_k) A_q := A(\boldsymbol{\mu}_k), \quad 1 \leq k \leq n, \tag{2.93}$$

where $A(\boldsymbol{\mu}_k)$ and A_q are matrices of size $N \times m$ and the functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ are assumed known.

2.6.2 A Simple Matrix Least Square Problem

To get an introduction, we look at the simple case where $Q = 3$, the vector of parameter $\boldsymbol{\mu}$ only has the component μ and the functions $\{g_q(\boldsymbol{\mu}) = \mu^q\}_{q=0}^{Q}$. This gives us the problem

$$A_0 + \mu_k A_1 + \mu_k^2 A_2 + \mu_k^3 A_3 := A(\mu_k), \quad 1 \leq k \leq n, \tag{2.94}$$

where $A(\boldsymbol{\mu}_k)$ and A_q are matrices of size $N \times N$.

To solve this by least squares we start by defining the matrix $M \in \mathbb{R}^{n \times 4}$ as follows

$$M = \begin{bmatrix} 1 & \mu_1 & \mu_1^2 & \mu_1^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \mu_n & \mu_n^2 & \mu_n^3 \end{bmatrix}. \tag{2.95}$$

Next we define the block matrices

$$X = \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} \in \mathbb{R}^{4N \times N}, \quad B = \begin{bmatrix} A(\mu_1) \\ \vdots \\ A(\mu_n) \end{bmatrix} \in \mathbb{R}^{nN \times N}. \tag{2.96}$$

From here we see two methods, either mapping the matrices to vectors and minimizing the 2-norm or working with the matrices we have and minimizing the Frobenius norm. In the following sections we look at both methods and see that they give the same result.

2.6.2.1 Method 1 — Mapping to vectors.

For this method we start by mapping the matrices in X and B from (2.96) to vectors giving us

$$x = \text{vec}(X) = \begin{bmatrix} \text{vec}(A_0) \\ \text{vec}(A_1) \\ \text{vec}(A_2) \\ \text{vec}(A_3) \end{bmatrix} \in \mathbb{R}^{4N^2}, \quad b = \text{vec}(B) = \begin{bmatrix} \text{vec}(A(\mu_1)) \\ \vdots \\ \text{vec}(A(\mu_n)) \end{bmatrix} \in \mathbb{R}^{nN^2}. \tag{2.97}$$

Next up we define the matrix $\bar{M} = M \otimes I_{N^2} \in \mathbb{R}^{nN^2 \times 4N^2}$ to set up the minimization problem

$$\min_x \|b - Mx\|_2^2 . \tag{2.98}$$

This minimization problem has the known solution of

$$\hat{x} = \left(\bar{M}^\top \bar{M}\right)^{-1} \bar{M}^\top b. \tag{2.99}$$

Next note that by the rules for the Kronecker product we have

$$\begin{aligned} \bar{M}^\top &= M^\top \otimes I_{N^2} \\ \bar{M}^\top \bar{M} &= \left(M^\top \otimes I_{N^2}\right)(M \otimes I_{N^2}) = \left(M^\top M\right) \otimes I_{N^2} \\ \left(\bar{M}^\top \bar{M}\right)^{-1} &= \left(M^\top M\right)^{-1} \otimes I_{N^2} \\ \left(\bar{M}^\top \bar{M}\right)^{-1} \bar{M}^\top &= \left(\left(M^\top M\right)^{-1} \otimes I_{N^2}\right)\left(M^\top \otimes I_{N^2}\right) = \left(\left(M^\top M\right)^{-1} M^\top\right) \otimes I_{N^2}. \end{aligned} \tag{2.100}$$

This gives us

$$\hat{x} = \left[\left(\left(M^\top M\right)^{-1} M^\top\right) \otimes I_{N^2}\right] b. \tag{2.101}$$

2.6.2.2 Method 2 — Minimizing the Frobenius Norm

For this method we define the matrix $\tilde{M} = M \otimes I_N \in \mathbb{R}^{nN \times 4N}$ to set up the minimization problem

$$\min_X \left\|\tilde{M}X - B\right\|_{\text{F}}^2 . \tag{2.102}$$

To solve this minimization problem we use the Frobenius inner product $\langle A, B\rangle_{\text{F}} = \text{tr}(A^\top B)$ to find when the Fréchet derivative of X is zero.

$$\begin{aligned} \left\|\tilde{M}(X+H) - B\right\|_{\text{F}}^2 &= \left\|(\tilde{M}X - B) + \tilde{M}H\right\|_{\text{F}}^2 \\ &= \left\|\tilde{M}X - B\right\|_{\text{F}}^2 + 2\left\langle \tilde{M}X - B, \tilde{M}H\right\rangle_{\text{F}} + \left\|\tilde{M}H\right\|_{\text{F}}^2 \\ \left\langle \tilde{M}X - B, \tilde{M}H\right\rangle_{\text{F}} &= \text{tr}\left(\left(\tilde{M}X - B\right)^\top \tilde{M}H\right) \\ = \text{tr}\left(\left(\tilde{M}^\top\left(\tilde{M}X - B\right)\right)^\top H\right) &= \left\langle \tilde{M}^\top\left(\tilde{M}X - B\right), H\right\rangle_{\text{F}}. \end{aligned} \tag{2.103}$$

In (2.103) H is the Fréchet derivative of X and it is zero when $\tilde{M}^\top\left(\tilde{M}X - B\right) = 0$ giving us

$$\hat{X} = \left(\tilde{M}^\top \tilde{M}\right)^{-1} \tilde{M}^\top B. \tag{2.104}$$

By using the same rules for the Kronecker product as shown in (2.100) we get

$$\hat{X} = \left[\left(\left(M^\top M\right)^{-1} M^\top\right) \otimes I_N\right] B. \tag{2.105}$$

2.6.2.3 The Resulting Algorithm

Looking at the results form the two methods in (2.101) and (2.105) we see that they are equal taking the vector mapping into consideration. This is as expected since the Frobinus norm for matrices is equivalent to the 2-norm, i.e

$$\langle A, B\rangle_{\mathrm{F}} = \operatorname{tr}\left(A^\top B\right) = \operatorname{vec}(A)^\top \operatorname{vec}(B). \tag{2.106}$$

Furthermore both (2.101) and (2.105) use the matrix $C = \left(M^\top M\right)^{-1} M^\top \in \mathbb{R}^{4\times n}$ for the Kronecker product with the corresponding identity matrix. This means that in both cases we can compute the matrices in $\hat{X}$ as follows

$$\hat{X}_q = \sum_{k=1}^{n} C_{qk} A(\mu_k). \tag{2.107}$$

Remark 2.12. *For Method 1 — Mapping to vectors this means that we do not map to vectors physically, but implement the effect on the matrices via* (2.107)*.*

Redefining X and B as vectors of matrices,

$$X = [A_0, A_1, A_2, A_3] \in \mathbb{R}^{4\times N\times N}, \quad B = [A(\mu_1), \cdots, A(\mu_n)] \in \mathbb{R}^{n\times N\times N}, \tag{2.108}$$

and generalizing from $Q = 3$ and $A_q, A(\mu_k) \in \mathbb{R}^{N\times N}$ gives rise to the following Least Squares algorithm, 3.

Algorithm 3 An algorithm solving problem (2.6.3) by least squares.

function $\hat{X} = MLS(M, B)$
  $B \in \mathbb{R}^{n\times N\times m}, M \in \mathbb{R}^{n\times(Q+1)}$
  compute the matrix $C = \left(M^\top M\right)^{-1} M^\top$
  for $q = 0$ to Q **do**
    compute the matrix $\hat{X}_q = \sum_{k=1}^{n} C_{qk} B_k$
  end for
end function

2.6.3 The General Matrix Least Square Problem

Going back to the general problem in (2.93),

$$\sum_{q=0}^{Q} g_q(\boldsymbol{\mu}_k) A_q := A(\boldsymbol{\mu}_k), \quad 1 \le k \le n, \tag{2.109}$$

where $A(\boldsymbol{\mu}_k)$ and A_q are matrices of size $N \times m$. The matrix $M^\top M$, in algorithm 3, represents the matrix of discrete inner products between the functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Furthermore, it is well known that in case when the functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ are orthogonal, the matrix $M^\top M$ will have a low *condition number*. This is important, because the higher the condition number, the more noise our data has. As a consequence, we need considerably more matrices $A(\boldsymbol{\mu}_k)$ than functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. For more specifics on how we obtain the functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ we refer to section 3.3.

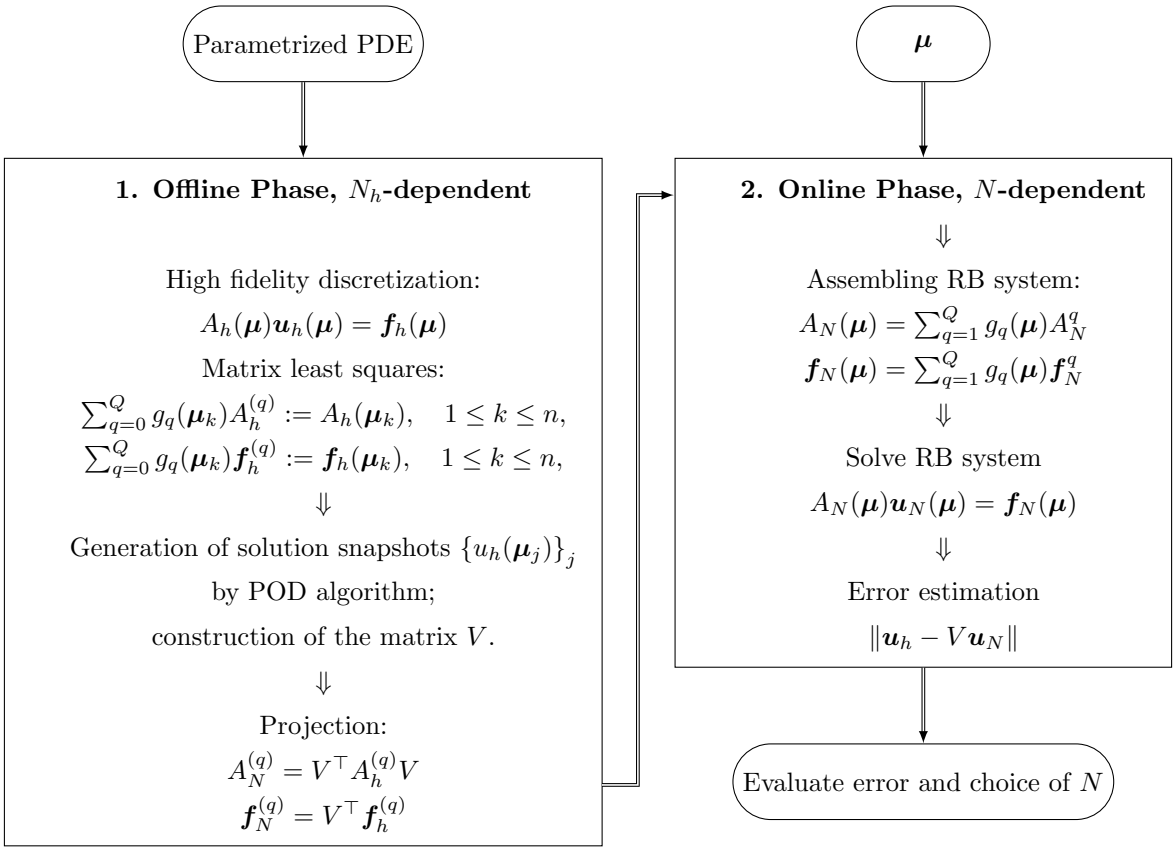


Figure 2.8. The non-affine reduced basis (RB) workflow at a glance.

Remark 2.13. *In general one needs $n+1$ snapshots in each direction for an approximation of order n.*

Using Matrix least squares instead off an affine mapping in the RB workflow described in figure 2.7 would give a new non-affine RB workflow shown in figure 2.8.

2.7 The Linear Elasticity Equations

In this section we present The Linear Elasticity Equations as our equations of interest for building a reduced-order model. We start by presenting the strong and weak formulations of the problem, and then present the mapping to the reference domain. Lastly we present our general algebraic system by restricting the problem arising from The Linear Elasticity Equations with the potential non-affine mapping to the reference domain.

2.7.1 Strong Formulation

The linear elasticity equations are described in terms of the stress tensor $\boldsymbol{\sigma} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, the strain tensor $\boldsymbol{\varepsilon} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, the body force $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^d$ and the displacement field $\boldsymbol{u} : \mathbb{R}^d \to \mathbb{R}^d$, where the latter is the unknown of the problem. Using this notation, we can describe linear elastic deformations of an isotropic solid occupying the domain $\Omega \subset \mathbb{R}^d$ as presented in section 2.1.2 in QMN2016 [1];

The governing equations consist of an equation stating the equilibrium of forces

$$-\operatorname{div}(\boldsymbol{\sigma}) = \boldsymbol{f} \quad \text{in} \quad \Omega, \tag{2.110}$$

the strain-displacement relation

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\top}\right) \tag{2.111}$$

and the constitutive law, which in the linear isotropic case takes the form

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) + \lambda(\operatorname{div}(\boldsymbol{u}))I. \tag{2.112}$$

Here μ and λ are the Lamé coefficients, which can be expressed in terms of the Young modulus E and the Poisson coefficient ν as

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{2.113}$$

The equilibrium problem for a linear elastic material can therefore be written as follows:

$$\begin{cases} -\operatorname{div}\left(\mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\top}\right) + \lambda(\operatorname{div}(\boldsymbol{u}))I\right) = \boldsymbol{f} & \text{in} \quad \Omega \\ \boldsymbol{u} = \boldsymbol{g} & \text{on} \quad \Gamma_{\text{D}} \\ \boldsymbol{\sigma}\boldsymbol{n} = \boldsymbol{h} & \text{on} \quad \Gamma_{\text{N}}. \end{cases} \tag{2.114}$$

Note that on the Dirichlet boundary Γ_{D} we impose a prescribed displacement,whereas on the Neumann boundary Γ_{N} we impose that the normal stress $\boldsymbol{\sigma}\boldsymbol{n}$ equals a prescribed traction vector $\boldsymbol{h}$.

Here (2.114) denotes the strong formulation of our problem. Using the Lamé coefficients in (2.113) we solve the *plane strain* case. However, if we want solve the *plane stress* case we need to replace the Lamé coefficient λ in (2.113) with $\bar{\lambda}$ as described by Hughes in [17], whereas μ stays the same as before,

$$\bar{\lambda} = \frac{2\lambda\mu}{\lambda + 2\mu} = \frac{E\nu}{1-\nu^2}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{2.115}$$

2.7.2 Weak Formulation

To derive the weak solution we multiply (2.114) by a test function $\boldsymbol{v} \in \mathbb{V}$ and integrate by parts. Noting that we can substitute $\nabla \boldsymbol{v}$ with $\boldsymbol{\varepsilon}(\boldsymbol{v})$ because $\boldsymbol{\varepsilon}(\boldsymbol{v})$ is the symmetric part of $\nabla \boldsymbol{v}$ and the product of the symmetric stress tensor and the anti-symmetric part of $\nabla \boldsymbol{v}$ is zero. This gives us the weak formulation of our problem as described in section 2.3.2 in QMN2016 [1];

The weak formulation of problem (2.114) reads: find $u \in \mathbb{V} = \left[\mathbb{H}^1_{\Gamma_{\text{D}}}(\Omega)\right]^d$ such that

$$a(\boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{v}) \quad \boldsymbol{v} \in \mathbb{V} \tag{2.116}$$

where we have defined the bilinear form $a : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$ as

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega} 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v})\, d\Omega + \int_{\Omega} \lambda \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{v})\, d\Omega, \tag{2.117}$$

and the linear form $f : \mathbb{V} \to \mathbb{R}$ as

$$f(\boldsymbol{v}) = \int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v}\, d\Omega + \int_{\Gamma_{\text{N}}} \boldsymbol{h} \cdot \boldsymbol{v}\, d\Gamma - a(\boldsymbol{r}_g, \boldsymbol{v}). \tag{2.118}$$

Here $\boldsymbol{r}_g \in \left[\mathbb{H}^1_{\Gamma_{\text{D}}}(\Omega)\right]^d$ is a lifting vector function such that $\boldsymbol{r}_g|_{\Gamma_{\text{D}}} = \boldsymbol{g}$; the solution to problem (2.114) is then obtained as $\boldsymbol{u} + \boldsymbol{r}_g$.

Here $\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v})$ denotes the tensor scalar product

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) = \sum_{i=1}^{d} \sum_{j=1}^{d} \boldsymbol{\varepsilon}(\boldsymbol{u})_{ij} \boldsymbol{\varepsilon}(\boldsymbol{v})_{ij}. \tag{2.119}$$

Furthermore, the bilinear form (2.117) is symmetric and strongly coercive, owing to Korn's inequality. Therefore problem (2.114) can be cast in the form (P_1) and admits a unique solution $u \in \mathbb{V}$ thanks to Lax-Milgram Lemma 2.1, which we discussed in section 2.1.3. We here note that if we have a pure traction problem, i.e $\Gamma_{\text{D}} = \emptyset$, $\boldsymbol{f}$ and $\boldsymbol{h}$ must satisfy some compatibility conditions, which is out of scope for this thesis. For the full computations, the Korn's inequality and the mentioned compatibility conditions, we refer the interested reader to chapter 11 in *The Mathematical Theory of Finite Element Methods* by Brenner and Scott (BS2008) [10].

2.7.3 Mapping to the Reference Domain

In this section we look at the mapping to a reference domain in case of The Linear Elasticity Equations. The mapping may be non-affine, which then will make the whole problem non-affine.

2.7.3.1 General notation

We denote by Ω the real domain, by $\tilde{\Omega}$ the reference domain and by Φ the coordinate mapping the reference coordinates $(\tilde{x}_1, \tilde{x}_2)$ to the real coordinates (x_1, x_2). For more specifics on how the mapping Φ looks we refer to section 3.1. We also denote the displacement vector and its gradient as

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \nabla \boldsymbol{u} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_2}{\partial x_1} \\ \frac{\partial u_1}{\partial x_2} & \frac{\partial u_2}{\partial x_2} \end{bmatrix}. \tag{2.120}$$

The symmetric gradient and the divergence are defined as

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\top}\right), \quad \operatorname{div}(\boldsymbol{u}) = \operatorname{tr}(\nabla \boldsymbol{u}) = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}. \tag{2.121}$$

Using this, we can now express the differentials

$$\frac{\partial u_1}{\partial x_1} = \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}\frac{\partial \tilde{x}_1}{\partial x_1} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}\frac{\partial \tilde{x}_2}{\partial x_1} \tag{2.122}$$

$$\frac{\partial u_1}{\partial x_2} = \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}\frac{\partial \tilde{x}_1}{\partial x_2} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}\frac{\partial \tilde{x}_2}{\partial x_2} \tag{2.123}$$

$$\frac{\partial u_2}{\partial x_1} = \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}\frac{\partial \tilde{x}_1}{\partial x_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}\frac{\partial \tilde{x}_2}{\partial x_1} \tag{2.124}$$

$$\frac{\partial u_2}{\partial x_2} = \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}\frac{\partial \tilde{x}_1}{\partial x_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}\frac{\partial \tilde{x}_2}{\partial x_2} \tag{2.125}$$

by the Jacobian

$$J = \begin{bmatrix} \frac{\partial \tilde{x}_1}{\partial x_1} & \frac{\partial \tilde{x}_1}{\partial x_2} \\ \frac{\partial \tilde{x}_2}{\partial x_1} & \frac{\partial \tilde{x}_2}{\partial x_2} \end{bmatrix}, \tag{2.126}$$

of the coordinate mapping Φ. Meaning that we can write the real gradient $\nabla \boldsymbol{u}$ in terms of the reference gradient $\nabla \tilde{\boldsymbol{u}}$ as

$$\nabla \boldsymbol{u} = J^{-\top} \nabla \tilde{\boldsymbol{u}}, \tag{2.127}$$

or simply

$$\frac{\partial u_i}{\partial x_j} = J_{1j}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} + J_{2j}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} \tag{2.128}$$

Next we denote the inverse of the Jacobian and its entries by

$$J^{-1} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \tag{2.129}$$

for simplicity, and by $|J|$ we denote the determinant of the Jacobian

$$|J| = \det(J) = \frac{\partial \tilde{x}_1}{\partial x_1}\frac{\partial \tilde{x}_2}{\partial x_2} - \frac{\partial \tilde{x}_1}{\partial x_2}\frac{\partial \tilde{x}_2}{\partial x_1}, \tag{2.130}$$

and note that

$$\text{Area}(\Omega) = \int_\Omega d\Omega = \int_{\tilde{\Omega}} |J|\, d\tilde{\Omega} \tag{2.131}$$

and

$$\int_\Omega \boldsymbol{g} \cdot \boldsymbol{v}\, d\Omega = \int_{\tilde{\Omega}} \boldsymbol{g}(\Phi) \cdot \tilde{\boldsymbol{v}} |J|\, d\tilde{\Omega} \tag{2.132}$$

Remark 2.14. *All index-mappings in the following sections assume a* 1*-indexing. If* 0*-indexing is used a shift is needed.*

2.7.3.2 The Symmetric Gradient Terms and the Encoding Matrix

Looking at the term $\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v})$ we have

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) = \frac{1}{4}\,\text{tr}\left(\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^\top\right)\left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^\top\right)\right) \tag{2.133}$$

$$= \frac{1}{4}\left[\text{tr}\left(\nabla \boldsymbol{u} \nabla \boldsymbol{v}\right) + \text{tr}\left(\nabla \boldsymbol{u} \nabla \boldsymbol{v}^\top\right) + \text{tr}\left(\nabla \boldsymbol{u}^\top \nabla \boldsymbol{v}\right) + \text{tr}\left(\nabla \boldsymbol{u}^\top \nabla \boldsymbol{v}^\top\right)\right] \tag{2.134}$$

$$= \frac{1}{2}\left(\nabla \boldsymbol{u} : \nabla \boldsymbol{v} + \nabla \boldsymbol{u} : \nabla \boldsymbol{v}^\top\right) \tag{2.135}$$

Using (2.128) for the fist term, we get

$$\nabla \boldsymbol{u} : \nabla \boldsymbol{v} = \sum_{i,j=1}^{2} \frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j} \tag{2.136}$$

$$= \sum_{i,j=1}^{2} \left(J_{1j}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} + J_{2j}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}\right)\left(J_{1j}\frac{\partial \tilde{v}_i}{\partial \tilde{x}_1} + J_{2j}\frac{\partial \tilde{v}_i}{\partial \tilde{x}_2}\right) \tag{2.137}$$

$$= \sum_{i,j=1}^{2} \left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} J_{1j}^2 \frac{\partial \tilde{v}_i}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} J_{1j} J_{2j} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} J_{2j} J_{1j} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} J_{2j}^2 \frac{\partial \tilde{v}_i}{\partial \tilde{x}_2}\right) \tag{2.138}$$

$$= \sum_{i=1}^{2} \left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1}(J_{11}^2 + J_{12}^2)\frac{\partial \tilde{v}_i}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_1}(J_{11}J_{21} + J_{12}J_{22})\frac{\partial \tilde{v}_i}{\partial \tilde{x}_2}\right. \tag{2.139}$$

$$\left. + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}(J_{21}J_{11} + J_{22}J_{12})\frac{\partial \tilde{v}_i}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}(J_{21}^2 + J_{22}^2)\frac{\partial \tilde{v}_i}{\partial \tilde{x}_2}\right). \tag{2.140}$$

This gives us

$$\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v}\, d\Omega = \sum_{i,j,k=1}^{2} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} W_{jk} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_k}\, d\tilde{\Omega}, \tag{2.141}$$

where $W = J^{-1}J^{-\top}|J|$. This result corresponds with what is presented in section 8.2.1 in QMN2016 [1].

Using (2.128) for the second term, we get

$$\nabla \boldsymbol{u} : \nabla \boldsymbol{v}^\top = \sum_{i,j=1}^{2} \frac{\partial u_i}{\partial x_j} \frac{\partial v_j}{\partial x_i} \tag{2.142}$$

$$= \sum_{i,j=1}^{2} \left(J_{1j} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} + J_{2j} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} \right) \left(J_{1i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + J_{2i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_2} \right) \tag{2.143}$$

$$= \sum_{i,j=1}^{2} \left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} J_{1j} J_{1i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} J_{1j} J_{2i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} J_{2j} J_{1i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2} J_{2j} J_{2i} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_2} \right) \tag{2.144}$$

$$= \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} J_{11}^2 \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} J_{12} J_{11} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} J_{11} J_{12} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} J_{12}^2 \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} \tag{2.145}$$

$$+ \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} J_{11} J_{21} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} J_{12} J_{21} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} J_{11} J_{22} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} J_{12} J_{22} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} \tag{2.146}$$

$$+ \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} J_{21} J_{11} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} J_{22} J_{11} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} J_{21} J_{12} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} J_{22} J_{12} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} \tag{2.147}$$

$$+ \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} J_{21}^2 \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} J_{22} J_{21} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} J_{21} J_{22} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} J_{22}^2 \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2}. \tag{2.148}$$

This gives us

$$\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v}^\top \, d\Omega = \sum_{i,j,k,l=1}^{2} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_k} Z_{\hat{i}(k,l),\hat{j}(i,j)} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_l} \, d\tilde{\Omega}, \tag{2.149}$$

where the *encoding matrix*

$$Z = \left(J^{-1} \otimes J^{-1} \right) |J| = \frac{1}{|J|} \begin{bmatrix} J_{11}^2 & J_{11}J_{12} & J_{12}J_{11} & J_{12}^2 \\ J_{11}J_{21} & J_{11}J_{22} & J_{12}J_{21} & J_{12}J_{22} \\ J_{21}J_{11} & J_{21}J_{12} & J_{22}J_{11} & J_{22}J_{12} \\ J_{21}^2 & J_{21}J_{22} & J_{22}J_{21} & J_{22}^2 \end{bmatrix} \tag{2.150}$$

encodes all information form the Jacobian, and the mappings $\hat{i}$ and $\hat{j}$ are defined as follows

$$\hat{i}(k,l) = l + 2\delta_{2k}, \quad \hat{j}(i,j) = i + 2\delta_{2j}. \tag{2.151}$$

Noticing that the element W_{jk} corresponds to two entries in Z, where $i = j$ in the mapping $\hat{j}(i,j)$, we have

$$W_{jk} = Z_{\hat{i}(j,k),\hat{j}(1,1)} + Z_{\hat{i}(j,k),\hat{j}(2,2)}. \tag{2.152}$$

This gives us

$$\int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \, d\Omega = \sum_{i,j,k,l=1}^{2} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} Z_{\hat{i}(j,k),\hat{j}(l,l)} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_k} \, d\tilde{\Omega}. \tag{2.153}$$

Now finally combining the terms we get

$$\int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \, d\Omega = \frac{1}{2} \sum_{i,j,k,l=1}^{2} \int_{\tilde{\Omega}} \left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} Z_{\hat{i}(j,k),\hat{j}(l,l)} \frac{\partial \tilde{v}_i}{\partial \tilde{x}_k} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_k} Z_{\hat{i}(k,l),\hat{j}(i,j)} \frac{\partial \tilde{v}_j}{\partial \tilde{x}_l} \right) d\tilde{\Omega}. \tag{2.154}$$

2.7.3.3 The Divergence Terms

We now look at the term $\operatorname{div}(\boldsymbol{u})\operatorname{div}(\boldsymbol{u})$. Using (2.128) we get

$$\operatorname{div}(\boldsymbol{u})\operatorname{div}(\boldsymbol{u}) = \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right)\left(\frac{\partial v_1}{\partial x_1} + \frac{\partial v_2}{\partial x_2}\right) \tag{2.155}$$

$$= \frac{\partial u_1}{\partial x_1}\frac{\partial v_1}{\partial x_1} + \frac{\partial u_1}{\partial x_1}\frac{\partial v_2}{\partial x_2} + \frac{\partial u_2}{\partial x_2}\frac{\partial v_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\frac{\partial v_2}{\partial x_2} = \sum_{i,j=1}^{2}\frac{\partial u_i}{\partial x_i}\frac{\partial v_j}{\partial x_j} \tag{2.156}$$

$$= \sum_{i,j=1}^{2}\left(J_{1i}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1} + J_{2i}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}\right)\left(J_{1j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + J_{2j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_2}\right) \tag{2.157}$$

$$= \sum_{i,j=1}^{2}\left(\frac{\partial \tilde{u}_i}{\partial \tilde{x}_1}J_{1i}J_{1j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_1}J_{1i}J_{2j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}J_{2i}J_{1j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_i}{\partial \tilde{x}_2}J_{2i}J_{2j}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_2}\right) \tag{2.158}$$

$$= \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}J_{11}^2\frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}J_{11}J_{12}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}J_{12}J_{11}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}J_{12}^2\frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} \tag{2.159}$$

$$+\frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}J_{11}J_{21}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}J_{11}J_{22}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}J_{12}J_{21}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}J_{12}J_{22}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} \tag{2.160}$$

$$+\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}J_{21}J_{11}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}J_{21}J_{12}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}J_{22}J_{11}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}J_{22}J_{12}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} \tag{2.161}$$

$$+\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}J_{21}^2\frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}J_{21}J_{22}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}J_{22}J_{21}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}J_{22}^2\frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} \tag{2.162}$$

This gives us

$$\int_{\Omega}\operatorname{div}(\boldsymbol{u})\operatorname{div}(\boldsymbol{u})\,d\Omega = \sum_{i,j,k,l=1}^{2}\int_{\tilde{\Omega}}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_k}Z_{\hat{i}(k,l),\hat{i}(i,j)}\frac{\partial \tilde{v}_j}{\partial \tilde{x}_l}\,d\tilde{\Omega}. \tag{2.163}$$

Do note that we here use the mapping $\hat{i}(i,j)$ instead of $\hat{j}(i,j)$ for the second index of Z, because this gives the difference from $\nabla\boldsymbol{u} : \nabla\boldsymbol{v}^\top$.

2.7.3.4 Needed Computations

Which combinations of $\tilde{u}$, Z and $\tilde{v}$ do we need? Many of the 16 components in the three computations are equal. Firstly all the 8 components in the "left" and "right" columns of all terms are equal. Furthermore, considering that $J_{ij}J_{kl} = J_{kl}J_{ij}$, the 4 components in middle of the "top" and "bottom" rows in the terms $\nabla\boldsymbol{u} : \nabla\boldsymbol{v}^\top$ and $\operatorname{div}(\boldsymbol{u})\operatorname{div}(\boldsymbol{u})$ are equal. In other words what seams to be 48 total components are in reality 28 unique components. This should be exploited.

Inspired by this we define

$$I_1 = \int_{\tilde{\Omega}}\sum_{i,j,k=1}^{2}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j}Z_{\hat{i}(j,k),\hat{k}(i)}\frac{\partial \tilde{v}_i}{\partial \tilde{x}_k}\,d\tilde{\Omega}, \tag{2.164}$$

$$I_2 = \int_{\tilde{\Omega}}\sum_{i,j,k=1}^{2}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j}Z_{\hat{i}(j,k),\hat{h}(i)}\frac{\partial \tilde{v}_i}{\partial \tilde{x}_j}\,d\tilde{\Omega}, \tag{2.165}$$

$$I_3 = \int_{\tilde{\Omega}}\sum_{i,j=1}^{2}\frac{\partial \tilde{u}_i}{\partial \tilde{x}_j}Z_{\hat{k}(j),\hat{n}(i)}\frac{\partial \tilde{v}_{\hat{l}(i)}}{\partial \tilde{x}_k}\,d\tilde{\Omega}, \tag{2.166}$$

$$I_4 = \int_{\tilde{\Omega}} \sum_{i,j=1}^{2} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} Z_{\hat{m}(i),\hat{n}(j)} \frac{\partial \tilde{v}_{\hat{l}(i)}}{\partial \tilde{x}_{\hat{l}(j)}} \, d\tilde{\Omega}, \tag{2.167}$$

$$I_5 = \int_{\tilde{\Omega}} \sum_{i,j=1}^{2} \frac{\partial \tilde{u}_i}{\partial \tilde{x}_j} Z_{\hat{m}(i),\hat{m}(j)} \frac{\partial \tilde{v}_{\hat{l}(i)}}{\partial \tilde{x}_{\hat{l}(j)}} \, d\tilde{\Omega}, \tag{2.168}$$

where

$$\begin{aligned} \hat{l}(i) &= 1 + \delta_{1i}, \qquad & \hat{n}(i) &= 2 + \delta_{1i}, \\ \hat{m}(i) &= 2 + \delta_{2i}, \qquad & \hat{h}(i) &= 1 + 3\delta_{1i} \end{aligned} \tag{2.169}$$

and

$$\hat{k}(i) = \hat{i}(i,i) = \hat{j}(i,i) = 1 + 3\delta_{2i}. \tag{2.170}$$

Then

$$\begin{aligned} \int_{\Omega} \nabla \boldsymbol{u} : \nabla \boldsymbol{v} \, d\Omega &= I_1 + I_2, \\ \int_{\Omega} \nabla \boldsymbol{u} : \nabla \boldsymbol{v}^{\top} \, d\Omega &= I_1 + I_3 + I_4, \end{aligned} \tag{2.171}$$

giving

$$\begin{aligned} \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \, d\Omega &= I_1 + \frac{1}{2} \left(I_2 + I_3 + I_4 \right), \\ \int_{\Omega} \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{u}) \, d\Omega &= I_1 + I_3 + I_5. \end{aligned} \tag{2.172}$$

Meaning that our two bilinear forms are defined by five main integrals, which will simplify the needed computations greatly, especially taking remark 2.17 into account.

2.7.4 The Algebraic System

In this section we restrict our problem and then deduce and present our algebraic systems resulting from the Linear Elasticity Equations, the mapping to the reference element and the use of matrix least squares.

2.7.4.1 Restricting the Problem; Our Cases and the Affine Parametization of the Problem

Noting the definition of the weak formulation (2.116) for our problem in section 2.7.2, we define our parameters of interest. Our chosen parameters of interest are split into two groups, $\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_{\text{mat}} & \boldsymbol{\mu}_{\text{geo}} \end{bmatrix}^{\top}$. The first group consist of the Young modulus E and the Poisson coefficient ν, $\boldsymbol{\mu}_{\text{mat}} = \begin{bmatrix} E & \nu \end{bmatrix}^{\top}$, i.e the affine material parameters in our problem. The second group, $\boldsymbol{\mu}_{\text{geo}}$, consists of the geometry parameters mapping the reference domain $\tilde{\Omega}$ to our real domain Ω will be discussed in section 3.1.

Continuing with just our affine material parameters $\boldsymbol{\mu}_{\text{mat}}$, the relation (2.112), gives us our affine parametric dependence as described in section 2.4.2. Furthermore, we restrict ourselves to the cases where the body force $\boldsymbol{f}$, the prescribed displacement $\boldsymbol{g}$ on Γ_{D}, and the prescribed traction vector $\boldsymbol{h}$ on Γ_{N} do not depend on the choice parameters $\boldsymbol{\mu}_{\text{mat}}$, and $\Gamma_{\text{D}} \neq \emptyset$. However, as we show in section 3.1, the domain Ω and thereby the boundaries Γ_{D} and Γ_{N} will depend on $\boldsymbol{\mu}_{\text{geo}}$.

Using the restrictions above we can affinely parametrize our problem for the material parameters by defining our $\boldsymbol{\mu}_{\text{mat}}$-dependent functions and $\boldsymbol{\mu}_{\text{mat}}$-independent forms for the bilinear

form (2.117) and linear form (2.118). We do this by setting $Q_a = 2$ and

$$\begin{aligned} 2\mu = \theta_a^{(1)}(\boldsymbol{\mu}_{\text{mat}}) = \frac{E}{(1+\nu)}, \quad a_1(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v})\, d\Omega, \\ \bar{\lambda} = \theta_a^{(2)}(\boldsymbol{\mu}_{\text{mat}}) = \frac{E\nu}{1-\nu^2}, \quad a_2(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \text{div}(\boldsymbol{u})\,\text{div}(\boldsymbol{v})\, d\Omega. \end{aligned} \tag{2.173}$$

for the bilinear form $a(\cdot,\cdot;\boldsymbol{\mu})$, and $Q_f = 4$,

$$\begin{aligned} \theta_f^{(1)}(\boldsymbol{\mu}_{\text{mat}}) = 1, \quad f_1(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\, d\Omega, \\ \theta_f^{(2)}(\boldsymbol{\mu}_{\text{mat}}) = 1, \quad f_2(\boldsymbol{v}) = \int_{\Gamma_{\text{N}}} \boldsymbol{h}\cdot\boldsymbol{v}\, d\Gamma, \\ \theta_f^{(3)}(\boldsymbol{\mu}_{\text{mat}}) = -\theta_a^{(1)}(\boldsymbol{\mu}_{\text{mat}}), \quad f_3(\boldsymbol{v}) = a_1(\boldsymbol{r}_g,\boldsymbol{v}), \\ \theta_f^{(4)}(\boldsymbol{\mu}_{\text{mat}}) = -\theta_a^{(2)}(\boldsymbol{\mu}_{\text{mat}}), \quad f_4(\boldsymbol{v}) = a_2(\boldsymbol{r}_g,\boldsymbol{v}), \end{aligned} \tag{2.174}$$

for the linear form $f(.;\boldsymbol{\mu})$. Note that we in (2.173) used the *plane stress* Lamé coefficient from (2.115).

Remark 2.15. *If we zero prescribed traction, i.e* $\boldsymbol{h} = \boldsymbol{0}$*, or only Dirichlet conditions, then the second linear form is zero, i.e* $f_2(\boldsymbol{\mu}) = 0$*. If also the mentioned Dirichlet conditions are zero, i.e* $\boldsymbol{g} = \boldsymbol{0}$*, then the two last linear forms are zero too, and effectively* $Q_f = 1$*.*

Finally we define the parameter space $\mathscr{P}_{\text{mat}} = E \times \nu \subset \mathbb{R}^2$, where we use

$$\begin{aligned} E \in [10, 310] \quad \text{GPa}, \\ \nu \in [0, 0.4], \end{aligned} \tag{2.175}$$

for all numerical examples.

Remark 2.16. *Since we for all our numerical examples will consider plane stress in 2D, we use the Lamé coefficient in* (2.115) *as described by Hughes in [17];*

$$\bar{\lambda} = \frac{2\lambda\mu}{\lambda+2\mu} = \frac{E\nu}{1-\nu^2}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{2.176}$$

gives us the ranges

$$\begin{aligned} \bar{\lambda} \in [0, 147.6] \quad GPa \\ \mu \in [3.6, 155] \quad GPa. \end{aligned} \tag{2.177}$$

for the Lamé coefficients.

2.7.4.2 The High-Fidelity System

Before using our definitions for the $\boldsymbol{\mu}_{\text{mat}}$-dependent functions and $\boldsymbol{\mu}_{\text{mat}}$-independent forms in section 2.7.4.1 and the Galerkin high-fidelity approximation in section 2.2.1 to assemble our high-fidelity system, we need two more definitions. First, since we have a vector test function $\tilde{\boldsymbol{v}}$, we need to define an *index mapping*, mapping the d dimensions to one single running index. This can be done through defining the mapping $j = d\cdot\hat{j} + d_j$ where $\hat{j}$ is the node number in the triangulation and d_j is the vector component of the function.

Example 2.2. *In 2D* ($d = 2$) *the mapping defined above gives*

$$\tilde{\boldsymbol{\varphi}}_{j=2\hat{j}+1} = \begin{bmatrix} \tilde{\varphi}_{\hat{j}} \\ 0 \end{bmatrix}, \quad \textit{and} \quad \tilde{\boldsymbol{\varphi}}_{j=2\hat{j}+2} = \begin{bmatrix} 0 \\ \tilde{\varphi}_{\hat{j}} \end{bmatrix}, \tag{2.178}$$

where $\varphi_{\hat{j}}$ *is the normal 1D basis function.*

With this we can write

$$\boldsymbol{u}_h(\boldsymbol{x};\boldsymbol{\mu}) = \sum_{j=1}^{N_h} u_h^{(j)}(\boldsymbol{\mu})\tilde{\boldsymbol{\varphi}}_j(\tilde{\boldsymbol{x}}_j) \tag{2.179}$$

where $\boldsymbol{u_h}(\boldsymbol{\mu}) = \begin{bmatrix} u_h^{(1)}(\boldsymbol{\mu}) & \dots & u_h^{(N_h)}(\boldsymbol{\mu}) \end{bmatrix}^\top$. Second, we define the discrete lifting function

$$\boldsymbol{r}_{g_h}(\boldsymbol{x};\boldsymbol{\mu}_{\text{geo}}) = \sum_{j\in\bar{\Gamma}_{\text{D}}} \boldsymbol{g}(\phi(\tilde{\boldsymbol{x}}_j;\boldsymbol{\mu}_{\text{geo}}))\tilde{\boldsymbol{\varphi}}_j(\tilde{\boldsymbol{x}}_j), \quad \bar{\Gamma}_{\text{D}} = \left\{ j : \tilde{\boldsymbol{x}}_j \in \tilde{\Gamma}_{\text{D}} \right\} \tag{2.180}$$

as the approximation of $\boldsymbol{g}$ on the boundary Γ_D, as described in section 4.5.1 of *Numerical Models for Differential Problems* by Quarteroni (Q2009) [5], using the reference Dirichlet boundary $\tilde{\Gamma}_{\text{D}}$.

Now, having defined the index mapping and discrete lifting function above, we can assemble our high-fidelity system affinely for our material parameters $\boldsymbol{\mu}_{\text{mat}}$ as described in section 2.4.2;

$$\begin{aligned} A_h(\boldsymbol{\mu}) &= 2\mu A_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}) + \lambda A_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}), \\ \boldsymbol{f}_h(\boldsymbol{\mu}) &= \boldsymbol{f}_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}) + \boldsymbol{f}_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}) \\ &\quad - 2\mu B_h^{(1)}(\boldsymbol{\mu}_{\text{geo}})\boldsymbol{r}_{g_h}(\boldsymbol{\mu}_{\text{geo}}) - \lambda B_h^{(2)}(\boldsymbol{\mu}_{\text{geo}})\boldsymbol{r}_{g_h}(\boldsymbol{\mu}_{\text{geo}}), \end{aligned} \tag{2.181}$$

where for $i, j = 1, \cdots N_h$

$$\begin{aligned} (A_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}))_{ij} = a_1(\tilde{\boldsymbol{\varphi}}_j, \tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}), \quad (A_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}))_{ij} = a_2(\tilde{\boldsymbol{\varphi}}_j, \tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}), \\ (\boldsymbol{f}_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}))_i = \boldsymbol{f}_1(\tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}), \quad (\boldsymbol{f}_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}))_i = \boldsymbol{f}_2(\tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}), \end{aligned} \tag{2.182}$$

and for $j \in \tilde{\Gamma}_{\text{D}}$ and $i = 1, \cdots N_h$

$$\begin{aligned} \boldsymbol{r}_{g_h}^{(j)}(\boldsymbol{\mu}_{\text{geo}}) = \boldsymbol{g}(\phi(\tilde{\boldsymbol{x}}_j; \boldsymbol{\mu}_{\text{geo}})), \\ (B_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}))_{ij} = a_1(\tilde{\boldsymbol{\varphi}}_j, \tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}), \quad (B_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}))_{ij} = a_2(\tilde{\boldsymbol{\varphi}}_j, \tilde{\boldsymbol{\varphi}}_i; \boldsymbol{\mu}_{\text{geo}}). \end{aligned} \tag{2.183}$$

This gives us the high-fidelity system

$$A_h(\boldsymbol{\mu})\mathring{\boldsymbol{u}}_h = \boldsymbol{f}_h(\boldsymbol{\mu}), \tag{2.184}$$

where the high-fidelity solution is obtained as $\boldsymbol{u}_h = \mathring{\boldsymbol{u}}_h + \boldsymbol{r}_{g_h}$.

Remark 2.17. *When using* (2.172) *to assemble the local matrices* $A^{(1)}(\boldsymbol{\mu}_{geo})$ *and* $A^{(2)}(\boldsymbol{\mu}_{geo})$ *on an element, one should exploit the symmetry of these matrices. Furthermore, if we in 2D use the basis functions*

$$\tilde{\boldsymbol{\varphi}}_{i0} = \begin{bmatrix} \tilde{\varphi}_i \\ 0 \end{bmatrix} \text{ and } \tilde{\boldsymbol{\varphi}}_{i1} = \begin{bmatrix} 0 \\ \tilde{\varphi}_i \end{bmatrix} \tag{2.185}$$

we see that

$$\begin{aligned} I_1(\tilde{\boldsymbol{\varphi}}_{i0}, \tilde{\boldsymbol{\varphi}}_{i0}) = I_2(\tilde{\boldsymbol{\varphi}}_{i1}, \tilde{\boldsymbol{\varphi}}_{i1}), \quad I_2(\tilde{\boldsymbol{\varphi}}_{i0}, \tilde{\boldsymbol{\varphi}}_{i0}) = I_1(\tilde{\boldsymbol{\varphi}}_{i1}, \tilde{\boldsymbol{\varphi}}_{i1}), \\ I_3(\tilde{\boldsymbol{\varphi}}_{i0}, \tilde{\boldsymbol{\varphi}}_{i1}) = I_3(\tilde{\boldsymbol{\varphi}}_{i1}, \tilde{\boldsymbol{\varphi}}_{i0}), \quad I_4(\tilde{\boldsymbol{\varphi}}_{i0}, \tilde{\boldsymbol{\varphi}}_{i1}) = I_5(\tilde{\boldsymbol{\varphi}}_{i1}, \tilde{\boldsymbol{\varphi}}_{i0}) \end{aligned} \tag{2.186}$$

and

$$I_5(\tilde{\boldsymbol{\varphi}}_{i0}, \tilde{\boldsymbol{\varphi}}_{i1}) = I_4(\tilde{\boldsymbol{\varphi}}_{i1}, \tilde{\boldsymbol{\varphi}}_{i0}). \tag{2.187}$$

This symmetry should also be exploited. All in all, reduces the number of integral per node of an element from 10 *to* 5. *This gives a total reduction of*

$$\frac{48 - 28 \cdot \frac{5}{10}}{48} = 70.8\% \tag{2.188}$$

per node of an element.

2.7.4.3 The Matrix Least Squares High-Fidelity System

Studying the high-fidelity system in section 2.7.4.2 we see that the affine mapping is not independent of the chosen parameters $\boldsymbol{\mu}$, since the affinity only applies to the material part $\boldsymbol{\mu}_{\text{mat}}$ and not the geometry part $\boldsymbol{\mu}_{\text{geo}}$. To solve this we generate multiple snapshots of the matrices $A_h^{(1)}(\boldsymbol{\mu}_{\text{geo}})$ and $A_h^{(2)}(\boldsymbol{\mu}_{\text{geo}})$, and the vectors

$$\begin{aligned}
\boldsymbol{f}_{h,\text{mls}}^{(0)}(\boldsymbol{\mu}_{\text{geo}}) &= \boldsymbol{f}_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}) + \boldsymbol{f}_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}), \\
\boldsymbol{f}_{h,\text{mls}}^{(1)}(\boldsymbol{\mu}_{\text{geo}}) &= B_h^{(1)}(\boldsymbol{\mu}_{\text{geo}})\boldsymbol{r}_{g_h}(\boldsymbol{\mu}_{\text{geo}}), \\
\boldsymbol{f}_{h,\text{mls}}^{(2)}(\boldsymbol{\mu}_{\text{geo}}) &= B_h^{(2)}(\boldsymbol{\mu}_{\text{geo}})\boldsymbol{r}_{g_h}(\boldsymbol{\mu}_{\text{geo}}),
\end{aligned} \tag{2.189}$$

and apply the matrix least squares algorithm, algorithm 3, on each of them,

$$\begin{aligned}
A_h^{(1)}(\boldsymbol{\mu}_{\text{geo}}) &= \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{h,\text{mls}}^{(1,q)}, \qquad A_h^{(2)}(\boldsymbol{\mu}_{\text{geo}}) = \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{h,\text{mls}}^{(2,q)} \\
\boldsymbol{f}_{h,\text{mls}}^{(0)}(\boldsymbol{\mu}_{\text{geo}}) &= \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(0,q)}, \\
\boldsymbol{f}_{h,\text{mls}}^{(1)}(\boldsymbol{\mu}_{\text{geo}}) &= \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(1,q)}, \quad \boldsymbol{f}_{h,\text{mls}}^{(2)}(\boldsymbol{\mu}_{\text{geo}}) = \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(2,q)}.
\end{aligned} \tag{2.190}$$

This gives us

$$\begin{aligned}
A_{h,\text{mls}}(\boldsymbol{\mu}) &= 2\mu \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{h,\text{mls}}^{(1,q)} + \lambda \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{h,\text{mls}}^{(2,q)}, \\
\boldsymbol{f}_{h,\text{mls}}(\boldsymbol{\mu}) &= \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(0,q)} - 2\mu \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(1,q)} - \lambda \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{h,\text{mls}}^{(2,q)},
\end{aligned} \tag{2.191}$$

leading to the *matrix least squares high-fidelity system*

$$A_{h,\text{mls}}(\boldsymbol{\mu}) \mathring{\boldsymbol{u}}_{h,\text{mls}} = \boldsymbol{f}_{h,\text{mls}}(\boldsymbol{\mu}), \tag{2.192}$$

where the solution is obtained as $\boldsymbol{u}_{h,\text{mls}} = \mathring{\boldsymbol{u}}_{h,\text{mls}} + \boldsymbol{r}_{g_h}$.

Remark 2.18. *We have here chosen to use the affine splitting from section 2.7.4.2 giving us two matrices and three vectors to preform matrix least squares on. This choice was made to reduce the number of functions $g_q(\boldsymbol{\mu}_{geo})$ and thereby the number of snapshots needed, since now each snapshot can represent multiple high-fidelity systems given the material parameters $\boldsymbol{\mu}_{mat}$.*

2.7.4.4 The Reduced-Order System

We can now assemble the reduced-order system. We do this by first constructing the transformation matrix V by the Proper Orthogonal Decomposition (POD) algorithm with respect to the energy inner product, algorithm 2, and then projecting the matrices and vectors from the matrix least squares high-fidelity system with it,

$$\begin{aligned}
A_{N,\text{mls}}^{(1,q)} &= V^\top A_{h,\text{mls}}^{(1,q)} V, \qquad A_{N,\text{mls}}^{(2,q)} = V^\top A_{h,\text{mls}}^{(2,q)} V \\
\boldsymbol{f}_{N,\text{mls}}^{(0,q)} &= V^\top \boldsymbol{f}_{h,\text{mls}}^{(0,q)}, \\
\boldsymbol{f}_{N,\text{mls}}^{(1,q)} &= V^\top \boldsymbol{f}_{h,\text{mls}}^{(1,q)}, \qquad \boldsymbol{f}_{N,\text{mls}}^{(2,q)} = V^\top \boldsymbol{f}_{h,\text{mls}}^{(2,q)}.
\end{aligned} \tag{2.193}$$

This gives us

$$\begin{aligned} A_N(\boldsymbol{\mu}) &= 2\mu \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{N,\text{mls}}^{(1,q)} + \lambda \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) A_{N,\text{mls}}^{(2,q)}, \\ \boldsymbol{f}_N(\boldsymbol{\mu}) &= \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{N,\text{mls}}^{(0,q)} - 2\mu \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{N,\text{mls}}^{(1,q)} - \lambda \sum_{q=1}^{Q} g_q(\boldsymbol{\mu}_{\text{geo}}) \boldsymbol{f}_{N,\text{mls}}^{(2,q)}, \end{aligned} \tag{2.194}$$

resulting in the reduced-order system

$$A_N(\boldsymbol{\mu}) \mathring{\boldsymbol{u}}_N = \boldsymbol{f}_N(\boldsymbol{\mu}), \tag{2.195}$$

where the recovered reduced-order solution is obtained as $\bar{\boldsymbol{u}}_N = V \mathring{\boldsymbol{u}}_N + \boldsymbol{r}_{g_h}$.

Remark 2.19. *It is important to note that for the POD algorithm with respect to the energy inner product, algorithm 2, we need the matrix $X_h = A_h(\boldsymbol{\mu})$ to be independent of the choice of parameters. We do this by setting $X_h = A_h(\boldsymbol{\mu}_{mean})$,*

$$\boldsymbol{\mu}_{mean} = \begin{bmatrix} E_{mean} & \nu_{mean} & \mu_{1,mean} & \mu_{2,mean} & \cdots \end{bmatrix}^\top, \tag{2.196}$$

where $\mu_{i,mean}$ is the mean value of geometry parameter μ_i.

Chapter 3

Case Studies

In this chapter we present our case studies on the *linear elasticity equations* for three different cases of a mapping to the reference domain. First we start by giving an introduction to our three numerical cases. Then we implement a solver in Python and test it using the *Patch Test*. Next, knowing that our solver works, we study how to determine the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Then we do some problem analysis to study the effect of the Matrix Least Squares algorithm 3 depending on (i) different choices of the order p of approximation for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ and (ii) the choice of geometry parameter range $\bar{G}_{\text{geo}}$ inside the maximum ranges found under the study how to determine the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Finally we end the problem analysis by doing a final analysis where we use more than 10 000 degrees of freedom to get a picture of the computational of the non-affine reduced-order workflow in figure 2.8.

The Python code building the solver, the Python code used in this thesis and the log files can be found via the doi-link [4].

3.1 An Introduction to Our Numerical Cases

As an introduction to our numerical cases, we present the computational results for three cases of the coordinate mapping Φ, mapping from the reference domain $\tilde{\Omega}$ to the real domain Ω, as mentioned in section 2.7.3. The three cases are:

Case 1 — Scaling of a Rectangle,

Case 2 — Dragging One Corner of a Rectangle, and

Case 3 — Dragging All Corners of a Rectangle.

These three cases are of interest since they cover most of the geometry deformations that can happen on a rectangle.

Remark 3.1. *By geometry deformations we mean deviations from the reference geometry, i.e the reference domain* $\tilde{\Omega}$.

3.1.1 Case 1 — Scaling of a Rectangle

In this section we look at the case where the real domain Ω is a scaling of the reference domain $\tilde{\Omega} = (0,1)^2$, i.e $\Omega = (0, L_x) \times (0, L_y)$, giving us the coordinate mapping

$$\Phi(\tilde{\boldsymbol{x}}) = \begin{bmatrix} x_0 + L_x \tilde{x}_1 \\ y_0 + L_y \tilde{x}_2 \end{bmatrix}, \tag{3.1}$$

where $x_0 = y_0 = 0$ in our case, since we do not translate the whole real domain away from the origin. The Jacobian of the coordinate mapping is

$$J = \begin{bmatrix} L_x & 0 \\ 0 & L_y \end{bmatrix} \tag{3.2}$$

and $|J| = L_x L_y$. This gives

$$J^{-1} = \frac{1}{|J|} \begin{bmatrix} L_y & 0 \\ 0 & L_x \end{bmatrix} \tag{3.3}$$

and the encoding matrix

$$Z = \left(J^{-1} \otimes J^{-1}\right) |J| = \begin{bmatrix} \frac{L_y}{L_x} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{L_x}{L_y} \end{bmatrix}. \tag{3.4}$$

Using the equations in section 2.7.3.4 this gives

$$I_1 = \frac{L_y}{L_x} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} \, d\tilde{\Omega} + \frac{L_x}{L_y} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} \, d\tilde{\Omega} = I_{11} + I_{12}, \tag{3.5}$$

$$I_2 = \frac{L_y}{L_x} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} \, d\tilde{\Omega} + \frac{L_x}{L_y} \int_{\tilde{\Omega}} \frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} \, d\tilde{\Omega} = I_{21} + I_{22}, \tag{3.6}$$

$$I_3 = 0, \tag{3.7}$$

$$I_4 = \int_{\tilde{\Omega}} \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} \right) d\tilde{\Omega} \tag{3.8}$$

and

$$I_5 = \int_{\tilde{\Omega}} \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} \right) d\tilde{\Omega}. \tag{3.9}$$

All in all this gives us 6 unique integrals, since I_1 and I_2 split in two and $I_3 = 0$. The computation of our two bilinear forms now follows from (2.172), giving us

$$\begin{aligned} \int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \, d\Omega &= \frac{L_y}{L_x} \left(I_{11} + \frac{1}{2} I_{21} \right) + \frac{L_x}{L_y} \left(I_{12} + \frac{1}{2} I_{22} \right) + \frac{1}{2} I_4, \\ \int_{\Omega} \operatorname{div}(\boldsymbol{u}) \operatorname{div}(\boldsymbol{u}) \, d\Omega &= \frac{L_y}{L_x} I_{11} + \frac{L_x}{L_y} I_{12} + I_5. \end{aligned} \tag{3.10}$$

Remark 3.2. *For this case we note that if the determinat of the Jacobian, $|J|$, of the coordinate transformation Φ is constant with respect to the spacial coordinates $\tilde{x}_1$ and $\tilde{x}_2$, the problem will still be affine.*

3.1.2 Case 2 — Dragging One Corner of a Rectangle

In this section we look at the case where the real domain Ω is the reference domain $\tilde{\Omega} = (0,1)^2$ and the real domain Ω has the vertices $(0,0), (1,0), (a,b), (0,1)$ where a and b are parameters dragging one corner of the rectangle. This gives us the coordinate mapping

$$\Phi(\tilde{\boldsymbol{x}}) = \begin{bmatrix} x_0 + \tilde{x}_1 \tilde{x}_2 \mu_1 + \tilde{x}_1 \\ y_0 + \tilde{x}_1 \tilde{x}_2 \mu_2 + \tilde{x}_2 \end{bmatrix}, \tag{3.11}$$

where $\mu_1 = a - 1$ and $\mu_2 = b - 1$, i.e. the deviation from the reference domain. Also, $x_0 = y_0 = 0$ in our case, since we again do not translate the whole real domain away from the origin. The Jacobian of the coordinate mapping is

$$J = \begin{bmatrix} \tilde{x}_2\mu_1 + 1 & \tilde{x}_1\mu_1 \\ \tilde{x}_2\mu_2 & \tilde{x}_1\mu_2 + 1 \end{bmatrix} \tag{3.12}$$

and $|J| = \mu_1\tilde{x}_2 + \mu_2\tilde{x}_1 + 1$. This gives

$$J^{-1} = \frac{1}{|J|} \begin{bmatrix} \tilde{x}_1\mu_2 + 1 & -\tilde{x}_1\mu_1 \\ -\tilde{x}_2\mu_2 & \tilde{x}_2\mu_1 + 1 \end{bmatrix} \tag{3.13}$$

and the encoding matrix $Z = \left(J^{-1} \otimes J^{-1}\right)|J|$ results in 10 unique entries, taking the symmetries into account.

$$\begin{aligned} &(\tilde{x}_2\mu_1 + 1)^2, && -\tilde{x}_2\mu_2\,(\tilde{x}_2\mu_1 + 1), \\ -\tilde{x}_2\mu_2\,&(\tilde{x}_1\mu_2 + 1), && -\tilde{x}_1\mu_1\,(\tilde{x}_2\mu_1 + 1), \\ &(\tilde{x}_1\mu_2 + 1)^2, && \tilde{x}_1\tilde{x}_2\mu_1\mu_2, \\ &\tilde{x}_2^2\mu_2^2, && (\tilde{x}_1\mu_2 + 1)\,(\tilde{x}_2\mu_1 + 1), \\ -\tilde{x}_1\mu_1\,&(\tilde{x}_1\mu_2 + 1), && \tilde{x}_1^2\mu_1^2. \end{aligned} \tag{3.14}$$

Using the equations in section 2.7.3.4 this gives

$$I_1 = \int_{\tilde{\Omega}} \left[\frac{\tilde{x}_1^2\mu_1^2}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} \right) + \frac{\tilde{x}_2^2\mu_2^2}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} \right) \right. \tag{3.15}$$

$$+ \frac{(\tilde{x}_1\mu_2 + 1)^2}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) + \frac{-\tilde{x}_2\mu_2\,(\tilde{x}_1\mu_2 + 1)}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) \tag{3.16}$$

$$\left. + \frac{(\tilde{x}_2\mu_1 + 1)^2}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} \right) + \frac{-\tilde{x}_1\mu_1\,(\tilde{x}_2\mu_1 + 1)}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} \right) \right] d\tilde{\Omega}, \tag{3.17}$$

$$I_2 = \int_{\tilde{\Omega}} \left[\frac{\tilde{x}_1^2\mu_1^2}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) + \frac{\tilde{x}_2^2\mu_2^2}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} \right) \right. \tag{3.18}$$

$$+ \frac{(\tilde{x}_1\mu_2 + 1)^2}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} \right) + \frac{-\tilde{x}_2\mu_2\,(\tilde{x}_1\mu_2 + 1)}{|J|} \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} \right) \tag{3.19}$$

$$\left. + \frac{(\tilde{x}_2\mu_1 + 1)^2}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} \right) + \frac{-\tilde{x}_1\mu_1\,(\tilde{x}_2\mu_1 + 1)}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) \right] d\tilde{\Omega}, \tag{3.20}$$

$$I_3 = \int_{\tilde{\Omega}} \left[\frac{-\tilde{x}_2\mu_2\,(\tilde{x}_2\mu_1 + 1)}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} \right) \right. \tag{3.21}$$

$$\left. + \frac{-\tilde{x}_1\mu_1\,(\tilde{x}_1\mu_2 + 1)}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) \right] d\tilde{\Omega}, \tag{3.22}$$

$$I_4 = \int_{\tilde{\Omega}} \left[\frac{\tilde{x}_1\tilde{x}_2\mu_1\mu_2}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_1} \right) \right. \tag{3.23}$$

$$\left. + \frac{(\tilde{x}_1\mu_2 + 1)\,(\tilde{x}_2\mu_1 + 1)}{|J|} \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2} \frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_1} \frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} \right) \right] d\tilde{\Omega} \tag{3.24}$$

and

$$I_5 = \int_{\tilde{\Omega}} \left[\frac{(\tilde{x}_1\mu_2 + 1)(\tilde{x}_2\mu_1 + 1)}{|J|} \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_1}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} \right) \right. \tag{3.25}$$

$$\left. + \frac{\tilde{x}_1\tilde{x}_2\mu_1\mu_2}{|J|} \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2}\frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1}\frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} \right) \right] d\tilde{\Omega}, \tag{3.26}$$

where

$$|J| = \mu_1\tilde{x}_2 + \mu_2\tilde{x}_1 + 1, \tag{3.27}$$

All in all this gives us 18 unique components, for "something" over $|J|$. The computation of our two bilinear forms now follows from (2.172).

3.1.3 Case 3 — Dragging All Corners of a Rectangle

In this section we look at the case where the real domain Ω is the reference domain $\tilde{\Omega} = (0,1)^2$ and the real domain Ω has the vertices $(0,0), (a_2,b_2), (a_3,b_3), (a_4,b_4)$ where a_i and b_i are parameters dragging three corners of the rectangle. The last corner, $(0,0)$, is not directly dragged, since dragging this corner is equivalent to translating the whole rectangle away from $(0,0)$ to (x_0,y_0) for the lower left corner.

This gives us the coordinate mapping

$$\Phi(\tilde{\boldsymbol{x}}) = \begin{bmatrix} x_0 + \tilde{x}_1\mu_1(1-\tilde{x}_2) + \tilde{x}_1\tilde{x}_2\mu_3 + \tilde{x}_2\mu_5(1-\tilde{x}_1) + \tilde{x}_1 \\ y_0 + \tilde{x}_1\mu_2(1-\tilde{x}_2) + \tilde{x}_1\tilde{x}_2\mu_4 + \tilde{x}_2\mu_6(1-\tilde{x}_1) + \tilde{x}_2 \end{bmatrix}, \tag{3.28}$$

where

$$\begin{aligned} \mu_1 &= a_2 - 1, & \mu_2 &= b_2, \\ \mu_3 &= a_3 - 1, & \mu_4 &= b_3 - 1, \\ \mu_5 &= a_4, & \mu_6 &= b_4 - 1, \end{aligned} \tag{3.29}$$

i.e. the deviation from the reference domain. Here again $x_0 = y_0 = 0$ in our case, since we again do not translate the whole real domain away from the origin, i.e. do not move the lower left corner. The Jacobian of the coordinate mapping is

$$J = \begin{bmatrix} \tilde{x}_2(\mu_3 - \mu_1 - \mu_5) + \mu_1 + 1 & \tilde{x}_1(\mu_3 - \mu_1 - \mu_5) + \mu_5 \\ \tilde{x}_2(\mu_4 - \mu_2 - \mu_6) + \mu_2 & \tilde{x}_1(\mu_4 - \mu_2 - \mu_6) + \mu_6 + 1 \end{bmatrix} \tag{3.30}$$

and

$$\begin{aligned} |J| = {} & 1 + \mu_1 + \mu_6 - \mu_2\mu_5 + \mu_1\mu_6 \\ & + \tilde{x}_1(\mu_4 - \mu_2 - \mu_6 + \mu_1\mu_4 - \mu_1\mu_6 - \mu_2\mu_3 + \mu_2\mu_5) \\ & + \tilde{x}_2(\mu_3 - \mu_1 - \mu_5 - \mu_1\mu_6 + \mu_2\mu_5 + \mu_3\mu_6 - \mu_4\mu_5). \end{aligned} \tag{3.31}$$

This gives

$$J^{-1} = \frac{1}{|J|} \begin{bmatrix} \tilde{x}_1(\mu_4 - \mu_2 - \mu_6) + \mu_6 + 1 & -\tilde{x}_1(\mu_3 - \mu_1 - \mu_5) - \mu_5 \\ -\tilde{x}_2(\mu_4 - \mu_2 - \mu_6) - \mu_2 & \tilde{x}_2(\mu_3 - \mu_1 - \mu_5) + \mu_1 + 1 \end{bmatrix} \tag{3.32}$$

and the encoding matrix $Z = \left(J^{-1} \otimes J^{-1}\right)|J|$ again resulting in 10 unique entries, taking the

symmetries into account.

$$\begin{gathered}
\left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)\left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right), \\
-\left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right), \\
\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right)^2, \\
\left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)^2, \\
-\left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right), \\
-\left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right), \\
\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right)\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right), \\
\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right)^2, \\
-\left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right), \\
\left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)^2.
\end{gathered} \tag{3.33}$$

Using the equations in section 2.7.3.4 this gives

$$I_1 = \int_{\tilde{\Omega}} \frac{1}{|J|} \Bigg[\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right)^2 \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_1}\right) \tag{3.34}$$

$$+ \left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)^2 \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_1}\right) \tag{3.35}$$

$$- \left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right) \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_1}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_1}\right) \tag{3.36}$$

$$- \left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right) \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_1}\right) \tag{3.37}$$

$$+ \left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)^2 \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_2}\right) \tag{3.38}$$

$$+ \left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right)^2 \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_2}\right)\Bigg] \, d\tilde{\Omega}, \tag{3.39}$$

$$I_2 = \int_{\tilde{\Omega}} \frac{1}{|J|} \Bigg[\left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - 1\mu_5\right)^2 \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_1}\right) \tag{3.40}$$

$$+ \left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right)^2 \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_1}\right) \tag{3.41}$$

$$- \left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right)\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right) \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_1}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_1}\right) \tag{3.42}$$

$$- \left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right) \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_1}\right) \tag{3.43}$$

$$+ \left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right)^2 \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_2}\right) \tag{3.44}$$

$$+ \left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right)^2 \left(\frac{\partial\tilde{u}_2}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_2}\right)\Bigg] \, d\tilde{\Omega}, \tag{3.45}$$

$$I_3 = \int_{\tilde{\Omega}} \frac{1}{|J|} \Bigg[- \left(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5\right) \cdot \tag{3.46}$$

$$\left(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1\right) \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_1} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_1}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_1}\right) \tag{3.47}$$

$$- \left(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2\right) \cdot \tag{3.48}$$

$$\left(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1\right) \left(\frac{\partial\tilde{u}_1}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_2}{\partial\tilde{x}_2} + \frac{\partial\tilde{u}_2}{\partial\tilde{x}_2}\frac{\partial\tilde{v}_1}{\partial\tilde{x}_2}\right)\Bigg] \, d\tilde{\Omega}, \tag{3.49}$$

$$I_4 = \int_{\tilde{\Omega}} \frac{1}{|J|} \Big[(\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5) \cdot \tag{3.50}$$

$$(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2) \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} \right) \tag{3.51}$$

$$+ (\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1) \cdot \tag{3.52}$$

$$(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1) \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} \right) \Big] \, d\tilde{\Omega}, \tag{3.53}$$

and

$$I_5 = \int_{\tilde{\Omega}} \frac{1}{|J|} \Big[(\tilde{x}_1\mu_2 - \tilde{x}_1\mu_4 + \tilde{x}_1\mu_6 - \mu_6 - 1) \cdot \tag{3.54}$$

$$(\tilde{x}_2\mu_1 - \tilde{x}_2\mu_3 + \tilde{x}_2\mu_5 - \mu_1 - 1) \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_2} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_1} \right) \tag{3.55}$$

$$+ (\tilde{x}_1\mu_1 - \tilde{x}_1\mu_3 + \tilde{x}_1\mu_5 - \mu_5) \cdot \tag{3.56}$$

$$(\tilde{x}_2\mu_2 - \tilde{x}_2\mu_4 + \tilde{x}_2\mu_6 - \mu_2) \left(\frac{\partial \tilde{u}_1}{\partial \tilde{x}_2} \frac{\partial \tilde{v}_2}{\partial \tilde{x}_1} + \frac{\partial \tilde{u}_2}{\partial \tilde{x}_1} \frac{\partial \tilde{v}_1}{\partial \tilde{x}_2} \right) \Big] \, d\tilde{\Omega} \tag{3.57}$$

where

$$\begin{aligned} |J| &= 1 + \mu_1 + \mu_6 - \mu_2\mu_5 + \mu_1\mu_6 \\ &\quad + \tilde{x}_1 (\mu_4 - \mu_2 - \mu_6 + \mu_1\mu_4 - \mu_1\mu_6 - \mu_2\mu_3 + \mu_2\mu_5) \\ &\quad + \tilde{x}_2 (\mu_3 - \mu_1 - \mu_5 - \mu_1\mu_6 + \mu_2\mu_5 + \mu_3\mu_6 - \mu_4\mu_5), \end{aligned} \tag{3.58}$$

All in all 18 unique components, for "something" over $|J|$. The computation of our two bilinear forms now follows from (2.172).

3.1.4 Some General Notes

As the observant reader may note Case 1 — Scaling a Rectangle is still affine as mentioned in remark 3.2, while Case 2 — Dragging One Corner of a Rectangle and Case 3 — Dragging All Corners of a Rectangle are not. This can be seen in the determinant of the Jacobian, $|J|$, of the coordinate transformation Φ. Furthermore, since the coordinate mapping Φ is bilinear in the special coordinates $\tilde{x}_1$ and $\tilde{x}_2$, as seen in Case 2 and Case 3, the use of bilinear elements, as the bilinear rectangle element 2.3, is recommended.

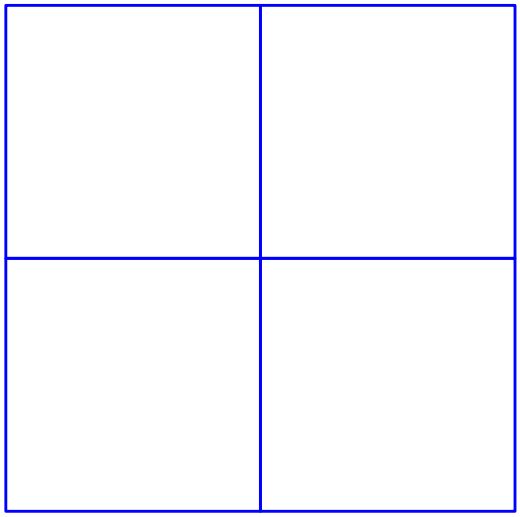
3.2 The Patch Test

For the following sections we need to implement a solver for the 2D case of the Linear Elastic Problems satisfying the restrictions described in section 2.7.4.1. So the goal of this section is to take a step to the side and test the implemented solver for all three cases mentioned in section 3.1. This was done by performing the Patch Test for all three cases. Focus was on testing that the solver solves linear problems exactly, even under our bilinear mapping Φ from the reference domain.

Remark 3.3. *For more information on the Patch Test we refer the reader to [18].*

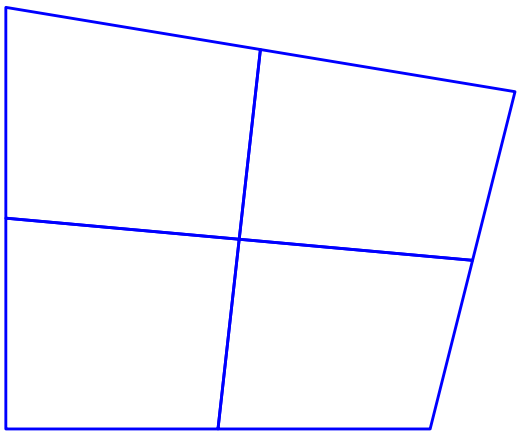
Test Case	1.	2.	3.	4.
$\boldsymbol{u}_{\text{ex}}$	$\begin{bmatrix} x & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & y \end{bmatrix}^\top$	$\begin{bmatrix} y & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & x \end{bmatrix}^\top$

Table 3.1. Patch Test — The different test cases for the exact solution $\boldsymbol{u}_{\text{ex}}$ in the patch tests.

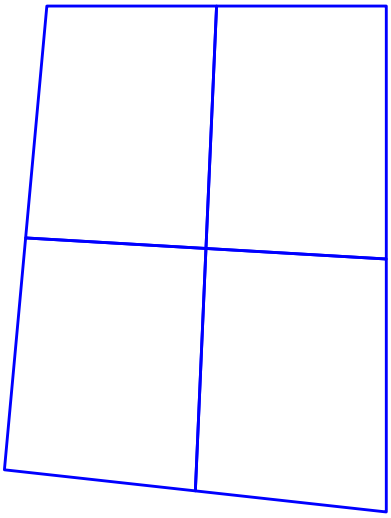


(a) The reference domain $\tilde{\Omega} = [0, 1]^2$.

(b) The real domain Ω in Case 1 — Scaling of a Rectangle using $L_x = 4$ and $L_y = 0.3$.



(c) The real domain Ω in Case 2 — Dragging One Corner of a Rectangle using $\mu_1 = 0.2$ and $\mu_2 = -0.2$.



(d) The real domain Ω in Case 3 — Dragging All Corners of a Rectangle using $\mu_1, \mu_2, \mu_3 = -0.1$ and $\mu_4, \mu_5, \mu_6 = 0.1$.

Figure 3.1. Patch Test — Domains with $n = 2$ element along the axes.

Test Case	1.	2.
$\boldsymbol{u}_{\text{ex}}(2, 0.15)$	$\begin{bmatrix} 2 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.15 \end{bmatrix}^\top$
$\boldsymbol{u}_h(2, 0.15)$	$\begin{bmatrix} 0.55 & 18.6 \cdot 10^{-18} \end{bmatrix}^\top$	$\begin{bmatrix} 7.8 \cdot 10^{-18} & 0.45 \end{bmatrix}^\top$
Test Case	3.	4.
$\boldsymbol{u}_{\text{ex}}(2, 0.15)$	$\begin{bmatrix} 0.45 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.55 \end{bmatrix}^\top$
$\boldsymbol{u}_h(2, 0.15)$	$\begin{bmatrix} 0.45 & 2.0 \cdot 10^{-18} \end{bmatrix}^\top$	$\begin{bmatrix} -3.5 \cdot 10^{-18} & 0.55 \end{bmatrix}^\top$

Table 3.2. Patch Test: Case 1 — Scaling of a Rectangle; A comparison of the displacement given by exact solution $\boldsymbol{u}_{\text{ex}}$ and finite element approximation $\boldsymbol{u}_h$ in the one free node $(2, 0.15)$ for the patch tests considering Case 1 — Scaling of a Rectangle, section 3.1.1, using $L_x = 4$ and $L_y = 0.3$.

3.2.1 General Patch Test Setup

Testing the solver in all three cases mentioned above is achieved by prescribing Dirichlet boundary conditions compatible with a linear exact solution u_{ex}. Since we do not have any Neumann boundary conditions, i.e., $\Gamma_{\text{N}} = \emptyset$ and we set the body force $\boldsymbol{f} = \boldsymbol{0}$. The components u_x and u_y of the displacement vector $\boldsymbol{u}$ are then both of the form $c_1 \cdot 1 + c_{\text{x}} \cdot x + c_{\text{y}} \cdot y$ where $c_1, c_{\text{x}}, c_{\text{y}} \in \mathbb{R}$. This gives us the problem

$$\begin{cases} -\operatorname{div}(\boldsymbol{\sigma}) = \boldsymbol{0} & \text{in } \Omega \\ \qquad \boldsymbol{u} = \boldsymbol{u}_{\text{ex}} & \text{on } \Gamma_{\text{D}}. \end{cases} \tag{3.59}$$

For the Patch Test we choose as few elements as possible for easing the search for any bugs. Thus, we have chosen a mesh with $n = 2$ elements along the axes shown on the reference domain in figure 3.1a, where the node located at $(0.5, 0.5)$ is the only free node, giving two degrees of freedom (dofs). Furthermore, the Patch Test is considered passed if the exact solution $\boldsymbol{u}_{\text{ex}}$ and the high-fidelity approximation $\boldsymbol{u}_h$ are exactly equal within machine precision on the patch.

Remark 3.4. *Since all three test cases depend on their respective geometry parameters the Patch Test is done for multiple values form the cross product of the parameters ranges G_{geo}.*

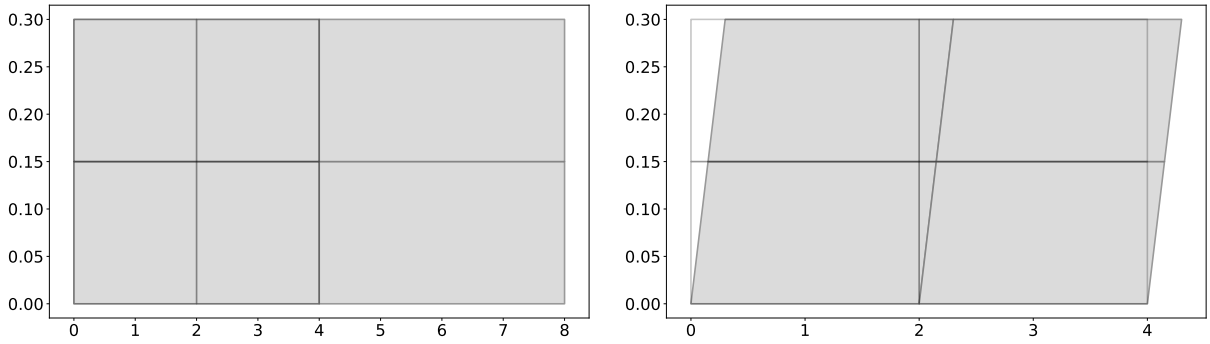
3.2.2 Results From the Patch Tests

Now, having briefly discussed the Patch Test, we consider the four test cases of $\boldsymbol{u}_{ex}$ in table 3.1

Because the exact solution $\boldsymbol{u}_{ex}$ is linear and we are using the Bilinear Lagrange Rectangle Element introduced in section 2.3, our four patch tests are considered passed if the exact solution $\boldsymbol{u}_{ex}$ and the high-fidelity approximation $\boldsymbol{u}_h$ coincide in the one free node.

3.2.2.1 Case 1 — Scaling of a Rectangle

We now look at Case 1 — Scaling of a Rectangle described in section 3.1.1. Choosing the values $L_x = 4$ and $L_y = 0.3$ gives the real domain Ω shown in figure 3.1b with the free node at $(2, 0.15)$. Performing the patch tests gives the results summarized in table 3.2. The patch tests are all considered passed since all values for the numerical solution $\boldsymbol{u}_h$ are within machine tolerance of their respective values for the exact solution $\boldsymbol{u}_{ex}$ in the free node $(2, 0.15)$. We also see in the plots of the displacement given by the high-fidelity solution in test cases 1 and 3, shown in figures 3.2a and 3.2b respectively, that the displacement of the node $(2, 0.15)$ is as expected given the exact solution. With this we mean that in test case 1, it moved $x = 2$ to the right, when in test case 3, it moved $y = 0.15$ to the right.



(a) Patch Test — Test Case 1. **(b)** Patch Test — Test Case 3.

Figure 3.2. Patch Test: Case 1 — Scaling of a Rectangle; The displacement given by the high-fidelity solutions of two patch tests for Case 1 — Scaling of a Rectangle, section 3.1.1, using $L_x = 4$ and $L_y = 0.3$. The displaced position is shown in gray with shading, whereas the initial position is shown in light gray without shading, i.e. the displaced position is the position being displaced to the right. The solutions were obtained using the mean-values for the Young modulus E and the Poisson coefficient ν. Please note the different scales on the x and y axes.

Test Case	1.	2.
$\boldsymbol{u}_{\text{ex}}(0.55, 0.45)$	$\begin{bmatrix} 0.55 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.45 \end{bmatrix}^\top$
$\boldsymbol{u}_h(0.55, 0.45)$	$\begin{bmatrix} 2 & 1.9 \cdot 10^{-18} \end{bmatrix}^\top$	$\begin{bmatrix} -3.8 \cdot 10^{-20} & 0.15 \end{bmatrix}^\top$
Test Case	3.	4.
$\boldsymbol{u}_{\text{ex}}(0.55, 0.45)$	$\begin{bmatrix} 0.45 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.15 \end{bmatrix}^\top$
$\boldsymbol{u}_h(0.55, 0.45)$	$\begin{bmatrix} 0.15 & -1.7 \cdot 10^{-19} \end{bmatrix}^\top$	$\begin{bmatrix} 4.6 \cdot 10^{-18} & 2 \end{bmatrix}^\top$

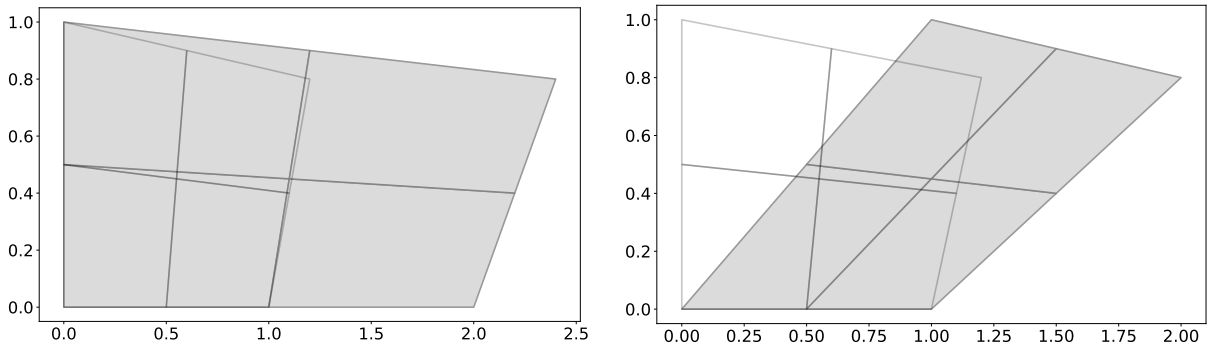
Table 3.3. Patch Test: Case 2 — Dragging One Corner of a Rectangle; A comparison of the displacement given by exact solution $\boldsymbol{u}_{\text{ex}}$ and finite element approximation $\boldsymbol{u}_h$ in the one free node $(0.55, 0.45)$ for the patch tests considering Case 2 — Dragging One Corner of a Rectangle, section 3.1.2, using $\mu_1 = 0.2$ and $\mu_2 = -0.2$.

Remark 3.5. *The patch tests also pass when performing them on all values for L_x and L_y in the cross product of the range* $(0.1, 5.1)$.

3.2.2.2 Case 2 — Dragging One Corner of a Rectangle

We now look at Case 2 — Dragging One Corner of a Rectangle described in section 3.1.2. Choosing the values $\mu_1 = 0.2$ and $\mu_2 = -0.2$ gives the real domain Ω shown in figure 3.1c and the free node at $(0.55, 0.45)$. Again performing the patch tests gives the results summarized in table 3.3. Again the patch tests are all considered passed since all values for the numerical solution $\boldsymbol{u}_h$ are within machine tolerance of their respective values for the exact solution $\boldsymbol{u}_{ex}$ in the free node $(0.55, 0.45)$. We also see in the plots of the displacement given by the high-fidelity solution in test cases 1 and 3, shown in figures 3.3a and 3.3b respectively, that the displacement of the node $(0.55, 0.45)$ is as expected given the exact solution. With this we mean that in test case 1, it moved $x = 0.55$ to the right, when in test case 3, it moved $y = 0.45$ to the right.

Remark 3.6. *The patch tests also pass when performing them on all values for μ_1 and μ_2 in the cross product of the range* $(-0.49, 0.49)$.



(a) Patch Test — Test Case 1.

(b) Patch Test — Test Case 3.

Figure 3.3. Patch Test: Case 2 — Dragging One Corner of a Rectangle; The displacement given by the high-fidelity solutions of two patch tests for Case 2 — Dragging One Corner of a Rectangle, section 3.1.2, using $\mu_1 = 0.2$ and $\mu_2 = -0.2$. The displaced position is shown in gray with shading, whereas the initial position is shown in light gray without shading, i.e. the displaced position is the position being displaced to the right. The solutions were obtained using the mean-values for the Young modulus E and the Poisson coefficient ν. Please note the different scales on the x and y axes.

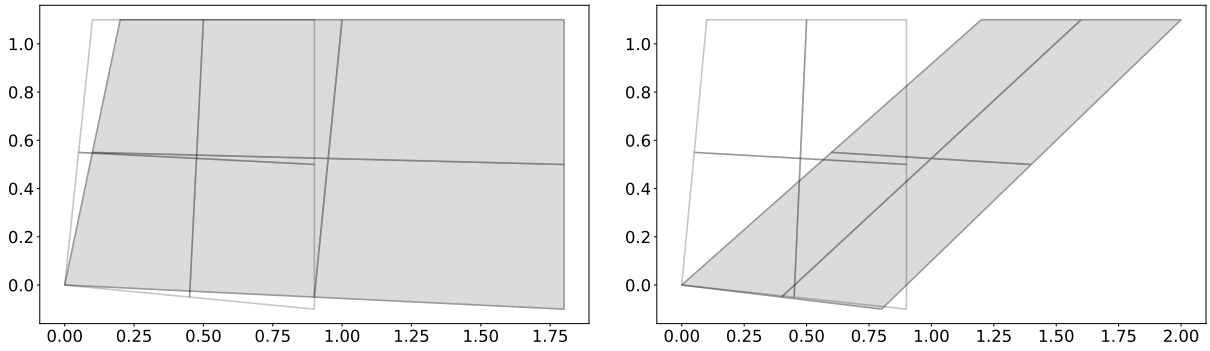
Test Case	1.	2.
$\boldsymbol{u}_{\text{ex}}(0.475, 0.525)$	$\begin{bmatrix} 0.55 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.45 \end{bmatrix}^\top$
$\boldsymbol{u}_h(0.475, 0.525)$	$\begin{bmatrix} 0.475 & -3.0 \cdot 10^{-17} \end{bmatrix}^\top$	$\begin{bmatrix} -3.3 \cdot 10^{-17} & 0.525 \end{bmatrix}^\top$
Test Case	3.	4.
$\boldsymbol{u}_{\text{ex}}(0.475, 0.525)$	$\begin{bmatrix} 0.45 & 0 \end{bmatrix}^\top$	$\begin{bmatrix} 0 & 0.15 \end{bmatrix}^\top$
$\boldsymbol{u}_h(0.475, 0.525)$	$\begin{bmatrix} 0.525 & -5.4 \cdot 10^{-17} \end{bmatrix}^\top$	$\begin{bmatrix} -2.4 \cdot 10^{-17} & 0.475 \end{bmatrix}^\top$

Table 3.4. Patch Test: Case 3 — Dragging All Corners of a Rectangle; A comparison of the displacement given by exact solution $\boldsymbol{u}_{\text{ex}}$ and finite element approximation $\boldsymbol{u}_h$ in the one free node $(0.55, 0.45)$ for the patch tests considering Case 3 — Dragging All Corners of a Rectangle, section 3.1.3, using $\mu_1, \mu_2, \mu_3 = -0.1$ and $\mu_4, \mu_5, \mu_6 = 0.1$.

3.2.2.3 Case 3 — Dragging All Corners of a Rectangle

We now look at Case 3 — Dragging All Corners of a Rectangle described in section 3.1.3. Choosing the values $\mu_1, \mu_2, \mu_3 = -0.1$ and $\mu_4, \mu_5, \mu_6 = 0.1$ gives the real domain Ω shown in figure 3.1d and the free node at $(0.55, 0.45)$. Again performing the patch tests gives the results summarized in table 3.4. Again the patch tests are passed since all values for $\boldsymbol{u}_h$ are within machine tolerance of their respective exact solution $\boldsymbol{u}_{ex}$ in the free node $(0.475, 0.525)$. We also see in the plots of the displacement given by the high-fidelity solution in test cases 1 and 3, shown in figures 3.4a and 3.4b respectively, that the displacement of the node $(0.475, 0.525)$ is as expected given the exact solution. With this we mean that in test case 1, it moved $x = 0.525$ to the right, when in test case 3, it moved $y = 0.475$ to the right. It is also worth mentioning that the lower right corner in test case 3 moves to the left since it initially had a negative y-value.

Remark 3.7. *The patch tests also pass when performing them on all values for $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$ and μ_6 in the cross product of the range $(-0.16, 0.16)$.*



(a) Patch Test — Test Case 1.

(b) Patch Test — Test Case 3.

Figure 3.4. Patch Test: Case 3 — Dragging All Corners of a Rectangle; The displacement given by the high-fidelity solutions of two patch tests for Case 3 — Dragging All Corners of a Rectangle, section 3.1.3, using $\mu_1, \mu_2, \mu_3 = -0.1$ and $\mu_4, \mu_5, \mu_6 = 0.1$. The displaced position is shown in gray with shading, whereas the initial position is shown in light gray without shading, i.e. the displaced position is the position being displaced to the right. The solutions were obtained using the mean-values for the Young modulus E and the Poisson coefficient ν. Please note the different scales on the x and y axes.

3.2.3 A Note on the use of Bilinear Elements

An important note here is the use of a bilinear element, in particular the Bilinear Lagrange Rectangle Element presented in section 2.3. If we would have used a linear element, e.g. the Linear Lagrange Triangle Element, see section 3.2 in BS2008 [10], all patch tests would have passed for Case 1 — Scaling of a Rectangle, since the coordinate mapping Φ from the reference domain $\tilde{\Omega}$ is still linear in the spacial coordinates $\tilde{x}_1$ and $\tilde{x}_2$. However, for Case 2 — Dragging One Corner of a Rectangle and Case 3 – Dragging All Corners of a Rectangle the patch tests would only pass when the parameters μ_i are such that they in principle just scale the rectangle. As a conclusion we state that the use of linear elements is unsuitable for our solver because we in general have a bilinear coordinate mapping Φ, which means we need bilinear elements in general.

3.3 Determining the Matrix Least Squares Functions

In this section we go back to the Matrix Least Square problem in section 2.6.3

$$\sum_{q=0}^{Q} g_q(\boldsymbol{\mu}_k) A_q := A(\boldsymbol{\mu}_k), \quad 1 \leq k \leq n, \tag{3.60}$$

where $A(\boldsymbol{\mu}_k)$ and A_q are matrices of size $N \times m$. More specifically we study the approximations needed for determining the Matrix Least Square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

We do this by first studying how to approximate the reciprocal of the determinant $\frac{1}{|J|}$, then studying how to approximate a function in general on the reference domain, and finally using this study, we determine the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

Remark 3.8. *By general function we refer to the function $\boldsymbol{g}(\boldsymbol{\phi})$ for the prescribed displacement $\boldsymbol{g}$ in the discrete lifting function*

$$\boldsymbol{r}_{g_h}(\boldsymbol{x}; \boldsymbol{\mu}_{geo}) = \sum_{j \in \bar{\Gamma}_{\mathrm{D}}} \boldsymbol{g}(\phi(\tilde{\boldsymbol{x}}_j; \boldsymbol{\mu}_{geo}))\tilde{\boldsymbol{\varphi}}_j(\tilde{\boldsymbol{x}}_j), \quad \bar{\Gamma}_{\mathrm{D}} = \left\{ j : \tilde{\boldsymbol{x}}_j \in \tilde{\Gamma}_{\mathrm{D}} \right\} \tag{3.61}$$

for the evaluation of the bilinear form $a(\boldsymbol{r}_{g_h}, \boldsymbol{v})$ on the reference domain, and the functions $\boldsymbol{f}(\boldsymbol{\phi})$ and $\boldsymbol{h}(\boldsymbol{\phi})$ for the body force $\boldsymbol{f}$, and the prescribed traction vector $\boldsymbol{h}$ in the integrals

$$\int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v} \, d\Omega = \int_{\tilde{\Omega}} \boldsymbol{f}(\Phi) \cdot \tilde{\boldsymbol{v}} |J| \, d\tilde{\Omega}, \tag{3.62}$$

and

$$\int_{\Omega} \boldsymbol{h} \cdot \boldsymbol{v} \, d\Omega = \int_{\tilde{\Omega}} \boldsymbol{h}(\Phi) \cdot \tilde{\boldsymbol{v}} |J| \, d\tilde{\Omega} \tag{3.63}$$

on the reference domain.

3.3.1 The Reciprocal of the Determinant

In general the reciprocal of the determinant takes the form

$$\frac{1}{k + \boldsymbol{c}^{\top} \tilde{\boldsymbol{x}}}, \tag{3.64}$$

which has the Taylor expansion

$$\frac{1}{k + \boldsymbol{c}^{\top} \tilde{\boldsymbol{x}}} = \frac{1}{k} - \frac{\boldsymbol{c}^{\top} \tilde{\boldsymbol{x}}}{k^2} + \frac{(\boldsymbol{c}^{\top} \tilde{\boldsymbol{x}})^2}{k^3} - \cdots, \text{ for } |\boldsymbol{c}^{\top} \tilde{\boldsymbol{x}}| < |k|. \tag{3.65}$$

3.3.1.1 Case 1 — Scaling of a Rectangle

In this case we know from section 3.1.1 that $|J| = L_x L_y$ and that $L_x, L_y > 0$. This gives $k = L_x L_y$ and $\boldsymbol{c} = \boldsymbol{0}$ in (3.65), resulting in an invalid Taylor expansion. However, this is not a problem since the determinant is constant and $L_x, L_y > 0$. This gives us the possible range for the geometry parameters L_x and L_y as $G_{\text{SR}} = (0, \infty)$ and the term

$$\left\{ \frac{1}{L_x L_y} \right\} \tag{3.66}$$

for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

3.3.1.2 Case 2 — Dragging One Corner of a Rectangle

In this case we know from section 3.1.2 that $|J| = 1 + \mu_1 \tilde{x}_2 + \mu_2 \tilde{x}_1$, giving us $k = 1$ and $\boldsymbol{c} = \begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix}^{\top}$. This results in the Taylor expansion

$$\frac{1}{k + \boldsymbol{c}^{\top} \tilde{\boldsymbol{x}}} = 1 - (\mu_1 \tilde{x}_2 + \mu_2 \tilde{x}_1) + (\mu_1 \tilde{x}_2 + \mu_2 \tilde{x}_1)^2 - \cdots, \text{ for } |\boldsymbol{c}^{\top} \boldsymbol{x}| < 1, \tag{3.67}$$

giving us the possible range for the geometry parameters μ_1 and μ_2 as $G_{\text{DR}} = (-0.5, 0.5)$ since $|\tilde{x}_i| \leq 1$ on the reference domain and the terms

$$\left\{ 1, \mu_1, \mu_2, \mu_1^2, \mu_1 \mu_2, \mu_2^2, \cdots \right\} \tag{3.68}$$

for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

Remark 3.9. *We count the term $\mu_i \mu_j, i \neq j$ as a second order term, and similar for higher order terms.*

3.3.1.3 Case 3 — Dragging All Corners of a Rectangle

In this case we know from section 3.1.3 that

$$\begin{aligned} |J| &= 1 + \mu_1 + \mu_6 - \mu_2\mu_5 + \mu_1\mu_6 \\ &\quad + \tilde{x}_1 \left(\mu_4 - \mu_2 - \mu_6 + \mu_1\mu_4 - \mu_1\mu_6 - \mu_2\mu_3 + \mu_2\mu_5\right) \\ &\quad + \tilde{x}_2 \left(\mu_3 - \mu_1 - \mu_5 - \mu_1\mu_6 + \mu_2\mu_5 + \mu_3\mu_6 - \mu_4\mu_5\right), \end{aligned} \tag{3.69}$$

giving us $k = 1 + \mu_1 + \mu_6 - \mu_2\mu_5 + \mu_1\mu_6$ and

$$\boldsymbol{c} = \begin{bmatrix} \mu_4 - \mu_2 - \mu_6 + \mu_1\mu_4 - \mu_1\mu_6 - \mu_2\mu_3 + \mu_2\mu_5 \\ \mu_3 - \mu_1 - \mu_5 - \mu_1\mu_6 + \mu_2\mu_5 + \mu_3\mu_6 - \mu_4\mu_5 \end{bmatrix} \tag{3.70}$$

This results in the Taylor expansion

$$\frac{1}{k + \boldsymbol{c}^\top \tilde{\boldsymbol{x}}} = \frac{1}{k} - \frac{\boldsymbol{c}^\top \tilde{\boldsymbol{x}}}{k^2} + \frac{(\boldsymbol{c}^\top \tilde{\boldsymbol{x}})^2}{k^3} - \cdots, \text{ for } |\boldsymbol{c}^\top \tilde{\boldsymbol{x}}| < |k|, \tag{3.71}$$

giving us the possible range for the geometry parameters $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$ and μ_6 as $G_{\text{QS}} = (-0.1\bar{6}, 0.1\bar{6})$ since $|\tilde{x}_i| \leq 1$ on the reference domain. Concerning the terms for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$, $\left(\boldsymbol{c}^\top \tilde{\boldsymbol{x}}\right)^p$ will give the polynomial terms

$$\left\{ \begin{array}{c} 1, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_1^2, \mu_1\mu_2, \mu_1\mu_3, \mu_1\mu_4, \mu_1\mu_5, \mu_1\mu_6, \mu_2^2, \mu_2\mu_3, \\ \mu_2\mu_4, \mu_2\mu_5, \mu_2\mu_6, \mu_3^2, \mu_3\mu_4, \mu_3\mu_5, \mu_3\mu_6, \mu_4^2, \mu_4\mu_5, \mu_4\mu_6, \mu_5^2, \mu_5\mu_6, \mu_6^2, \cdots \end{array} \right\}. \tag{3.72}$$

However, $\frac{1}{k^{p+1}}$ gives problems since k depends on multiple μ_i. This also makes it difficult to orthogonalize the $\frac{1}{k^{p+1}}$ to the polynomial terms. As a solution to this problem we choose to only use the polynomial terms for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$, noting that this will correspond to further expanding $\frac{1}{k^{p+1}}$ in μ_1, μ_2 and μ_5, which is valid beyond our geometry parameter range G_{QS}.

3.3.2 Approximating a Function on the Reference Domain

In general, since we do not know what kind of functions $\boldsymbol{g}$, $\boldsymbol{f}$ and $\boldsymbol{h}$ are, we do not know what kind of functions $\boldsymbol{g}(\boldsymbol{\phi})$, $\boldsymbol{f}(\boldsymbol{\phi})$ and $\boldsymbol{h}(\boldsymbol{\phi})$ are, in terms of the the geometry parameters L_i or μ_i. However, both components of coordinate mapping ϕ in all the cases are linear in the geometry parameters L_i or μ_i. This means that the function components are multivariate polynomials of degree p in the spatial coordinates $\tilde{x}_1$ and $\tilde{x}_2$, and we get multivariate polynomials p in the geometry parameter. For instance, if the function components are multivariate polynomials of degree $p = 2$ in the spatial coordinates $\tilde{x}_1$ and $\tilde{x}_2$, and we have the geometry parameters μ_1 and μ_2, i.e. case 2, we get the terms

$$\{1, \mu_1, \mu_2, \mu_1\mu_2\} \tag{3.73}$$

for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

3.3.3 Approximating the Determinant and the Numerators of the Encoding Matrix

We know that in general the determinant $|J|$ is a "bilinear" polynomial and the numerators of the encoding matrix Z, (2.150) are second order, $p = 2$, polynomials in the geometry parameters, i.e they take the forms

$$a_0 \cdot 1 + \sum_i b_i\mu_i + \sum_{i \neq j} c_{ij}\mu_i\mu_j, \tag{3.74}$$

and

$$a_0 \cdot 1 + \sum_i b_i \mu_i + \sum_{ij} c_{ij} \mu_i \mu_j, \tag{3.75}$$

respectively. Which for case 2 gives the terms

$$\left\{1, \mu_1, \mu_2, \mu_1^2, \mu_1\mu_2, \mu_2^2\right\} \tag{3.76}$$

for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$.

3.3.4 Determining the Matrix Least Squares Functions.

To put all the terms from the previous sections together we need to "multiply" them, i.e. "Multiplying" (3.73) using order $p = 2$ and (3.76) would give the set

$$\left\{1, \mu_1, \mu_2, \mu_1^2, \mu_1\mu_2, \mu_2^2, \mu_1^3, \mu_1^2\mu_2, \mu_1\mu_2^2, \mu_2^3\right\} \tag{3.77}$$

and "multiplying" (3.66) and (3.76) using L_i would give the set

$$\left\{1, L_x, L_y, L_x^2, L_xL_y, L_y^2, \frac{1}{L_xL_y}, \frac{1}{L_y}, \frac{1}{L_x}, \frac{L_x}{L_y}, \frac{L_y}{L_x}\right\}. \tag{3.78}$$

Observe that the approximation set, (3.76), for the determinant and the numerator of the encoding matrix in general always is of order $p = 2$. Also observe that the reciprocal of the determinant may reduce the order of the resulting set. Therefore we suggest that we always compute the $p + 2$ polynomial set of terms for the approximation of a general function, the determinant and the numerator of the encoding matrix. This may then be "multiplied" with the non-polynomial terms of the set order p for the reciprocal of the determinant, if the non-polynomial terms are few and simple. Lastly we get the Matrix Least Squares function set from the resulting set by cutting out the terms higher than p in absolute value. This means that the $p = 2$ order set for Case 1 — Scaling of a Rectangle would be

$$\left\{1, \frac{1}{L_y}, \frac{1}{L_x}, L_y, L_x, \frac{1}{L_xL_y}, \frac{L_y}{L_x}, L_y^2, \frac{L_x}{L_y}, L_xL_y, L_x^2, \frac{L_y^2}{L_x}, \frac{L_x^2}{L_y}\right\}. \tag{3.79}$$

Remark 3.10. *The order of appearance of the terms in* (3.79) *are given to match the order of appearance in figure 3.8.*

Furthermore, as mentioned in section 2.6.3, the functions should be close to orthogonal. So we will be using the Legendre polynomials shifted to the currently used geometry parameter range G_{geo}. This makes it more efficient to store the order of the the terms, i.e for each term a list of the order for each geometry parameter, in the mentioned sets above. Then when "multiplying" instead "add" the orders of the terms together, cut off the terms higher than p in absolute value and then compute the resulting Matrix Least Square function set $G_{\text{MLS}} = \{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. This results in algorithm 4.

Remark 3.11. *It is not always necessary to make all the functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ orthogonal to each other. Here we have the orthogonal set of shifted Legendre polynomials $\tilde{L}(x)$ on the interval $[a, b], a, b > 0$ and the functions $\frac{\tilde{P}_n(x)}{x}$. Making $\frac{1}{x}$ orthogonal to $\tilde{P}_n(x)$ would give the function $\frac{1}{x} - T_n(x)$, where $T_n(x)$ is a polynomial. Hence, as mentioned above we see the case of Scaling a Rectangle to have simple non-polynomial parts in the set for $\frac{1}{|J|}$. However, in general we would only recommend to not use an orthogonal basis when we know that we have a good matrix least square fit for few matrix least square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$, as seen in section 3.1.1.*

Algorithm 4 An algorithm to construct the set of Martix Least Square functions $G_{\text{MLS}} = \{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ given the order p, the set of geometry parameters μ_{geo} and the geometry parameter range G_{geo}. The order p is interpreted as in remark 3.9.

function $G_{\text{MLS}} = FuncsMLS(p, \mu_{\text{geo}}, G_{\text{geo}})$
  Compute the set, C_{MLS}, of all orders of the all terms in the $p+2$ polynomial set
  if the set for $\frac{1}{|J|}$ will have simple non-polynomial parts **then**
    i.e, case: scaling of a rectangle
    Compute the set, K_{MLS}, of all orders of all terms in the p order set of non-polynomial terms in the p order set for $\frac{1}{|J|}$, i.e $\left\{\frac{1}{L_x L_y}\right\}$
    "Add" C_{MLS} and K_{MLS} to from $\bar{G}_{\text{MLS}}$
  else
    $\bar{G}_{\text{MLS}} = C_{\text{MLS}}$
  end if
  Cut out the lists where the absolute value of the sum of the entries is higher than p, the sum of the positive entries is higher than p and sum of the negative entries is smaller than $-p$ from $\bar{G}_{\text{MLS}}$
  for list of orders of the geometry parameters in $\bar{G}_{\text{MLS}}$ **do**
    Construct $g_q(\boldsymbol{\mu})$ by using Legendre polynomials shifted to G_{geo} for positive orders of the geometry parameters, use the negatives as they are.
  end for
end function

3.4 Constant Body Force in 2D

Before we can start our analysis we need a problem to study. For this analysis we use the case of Constant Body force in 2D or more precisely the case where one component of the body force $\boldsymbol{f}$ is a constant, we have homogeneous Dirichlet boundary conditions on the west side of the real domain Ω, i.e a clamped down side, and the rest of the boundary has homogeneous Neumann boundary conditions, i.e a traction free boundary. The proposed problem then becomes

$$\begin{cases} -\operatorname{div}(\boldsymbol{\sigma}) = \boldsymbol{f} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix} & \text{in } \Omega = [0,1]^2 \\ \boldsymbol{u} = \boldsymbol{0} & \text{on } \Gamma_{\text{D}} \\ \boldsymbol{\sigma n} = \boldsymbol{0} & \text{on } \Gamma_{\text{N}} \end{cases} \tag{3.80}$$

A picture of the proposed problem given on the domain $\Omega = \tilde{\Omega} = [0,1]^2$ can be seen in figure 3.5.

Furthermore, to get our field we set

$$\alpha = \rho \cdot 100g \cdot t = 8 \cdot 10^3 \text{ kg/m}^3 \cdot 100 \cdot 9.81\text{m/s}^2 \cdot 0.01\text{m} = 784.8 \cdot 10^2 \text{ N/m}, \tag{3.81}$$

approximately equal to the mass density of steal times 100 the gravitational acceleration times the thickness $t = 1$ cm.

Remark 3.12. *For Case 1 — Scaling of a Rectangle and Case 2 - Dragging one corner of a rectangle the boundary conditions and the body force $\boldsymbol{f}$ are given on the real domain Ω. However, for Case 3 — Dragging all corner of a rectangle the boundary conditions and the body force $\boldsymbol{f}$ are given on the reference domain $\tilde{\Omega}$ because here the left side deviates from the line $x = 0$.*

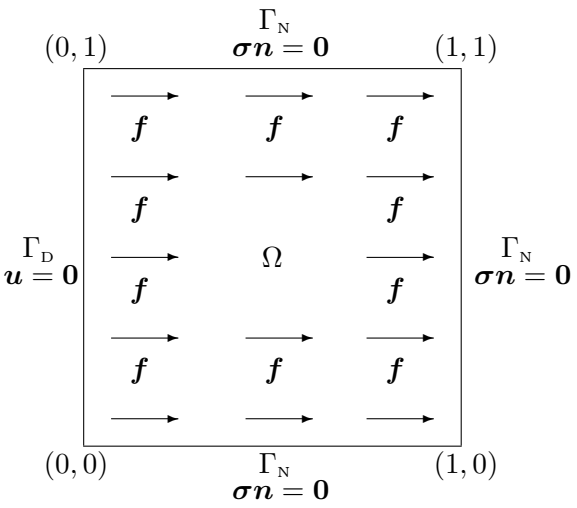


Figure 3.5. Constant Body force in 2D — A picture of describing the Constant Body force in 2D problem, showing the body force $\boldsymbol{f}$ and boundary conditions on the domain $\Omega = \tilde{\Omega} = [0,1]^2$.

3.5 Discussions

In this section we discuss the three cases presented in section 3.1. We do this by studying the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity MLS solution $u_{h,\text{mls}}(\boldsymbol{\mu})$,

$$\frac{\|u_h(\boldsymbol{\mu}) - \boldsymbol{u}_{h,\text{mls}}(\boldsymbol{\mu})\|_a}{\|u_h(\boldsymbol{\mu})\|_a}, \tag{3.82}$$

for orders p of approximation for the matrix least square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Then, having determined good choice for the order p, we study the relative contribution per term for the matrices from the MLS algorithm 3

$$\frac{\|g_q(\boldsymbol{\mu})A_{i,q}\|_{\text{F}}}{\|A_i\|_{\text{F}}}, i = 1, 2, \tag{3.83}$$

and the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$,

$$\frac{\|u_h(\boldsymbol{\mu}) - V\boldsymbol{u}_N(\boldsymbol{\mu})\|_a}{\|u_h(\boldsymbol{\mu})\|_a}, \tag{3.84}$$

for different N, to see the effect of the MLS algorithm on the reduced-order solution constructed by the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2.

We start by testing the Matrix Least Squares algorithm using Case 1 — Scaling a rectangle, then study Case 2 — Dragging one Corner of a Rectangle and Case 3 — Dragging all Corners of a Rectangle. In all these studies we are using the problem of Constant Body force in 2D, discussed in section 3.4, with $n = 20$ elements along the axes. Moreover, from some small scale testing for when the matrix $M^\top M$ has large condition numbers, given different orders p, in the Matrix Least Squares algorithm 3, we find that a 25×25 uniform gird works. This is a good compromise between accuracy and computational time given we have two geometry parameters as in Case 1 — Scaling of a Rectangle and Case 2 — Dragging all Corners of a Rectangle. This gives $25^2 = 625$ snapshots of matrices and vectors. For Case 3 — Dragging all Corners of a Rectangle we use a $3 \times 3 \times 3 \times 3 \times 3 \times 3$ uniform grid, giving $3^6 = 729$ snapshots of matrices and vectors. This may seem as too few snapshots, however, $5^6 = 15\,625$ would be the next alternative and this is too much considering computational time for this analysis.

(a) $L_x = 0.1, L_y = 5.1$ **(b)** $L_x = 5.1, L_y = 0.1$

Figure 3.6. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The two extremes of the geometry range $\bar{G}_{\text{SR}} = (0.1, 5.1)$ for L_x and L_y using $n = 2$ elements per axes.

Finally, we study Case 2 — Dragging one Corner of a Rectangle again using what we have learned from the previous studies, i.e. we have chosen the order p of approximation for the matrix least square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Moreover, we still use the problem of Constant Body force in 2D discussed in section 3.4 and a 25×25 uniform grid for μ_1 and μ_2. However, we are using $n = 90$ elements along the axes to give us $N_h = 16\,380$ degrees of freedom (dofs) or free nodes. This is more than $n_s = 15\,625 = 25 \times 25 \times 5 \times 5$, which is the number of solutions in the snapshot matrix (2.74) in the POD algorithm. Having more than 10 000 degrees of freedom here give us a good example to study the computational times.

The Python code building the solver, the Python code used in this thesis and the log files can be found via the doi-link [4].

Remark 3.13. *In this section we have used E_{mean} and ν_{mean} for our material parameters $\boldsymbol{\mu}_{mat}$.*

3.5.1 Testing the Matrix Least Squares algorithm; Case 1 — Scaling of a Rectangle

As mentioned in section 3.1.1 the case of scaling a rectangle is still affine, see (3.2). Therefore we are using this case to test the Matrix Least Squares algorithm, algorithm 3. Furthermore, we are restricting us to the geometry range $\bar{G}_{\text{SR}} = (0.1, 5.1)$ for L_x and L_y, where the two extremes of $L_x = 0.1, L_y = 5.1$ and $L_x = 5.1, L_y = 0.1$ are shown in figure 3.6 for $n = 2$ elements per axes.

3.5.1.1 The Relative Errors Between the High-fidelity Solution and the High-fidelity Matrix Least Squares Solution

Studying the relative errors in figure 3.7 we see that the errors are below 10^{-9} from order $p = 2$ until $p = 8$, and from $p = 8$ until $p = 10$ they increase again. This is as expected since the term $L_x L_y$, i.e. the determinant of the coordinate transformation Φ in this case, is considered a second order term and therefore is not included before order $p = 2$. This term is important

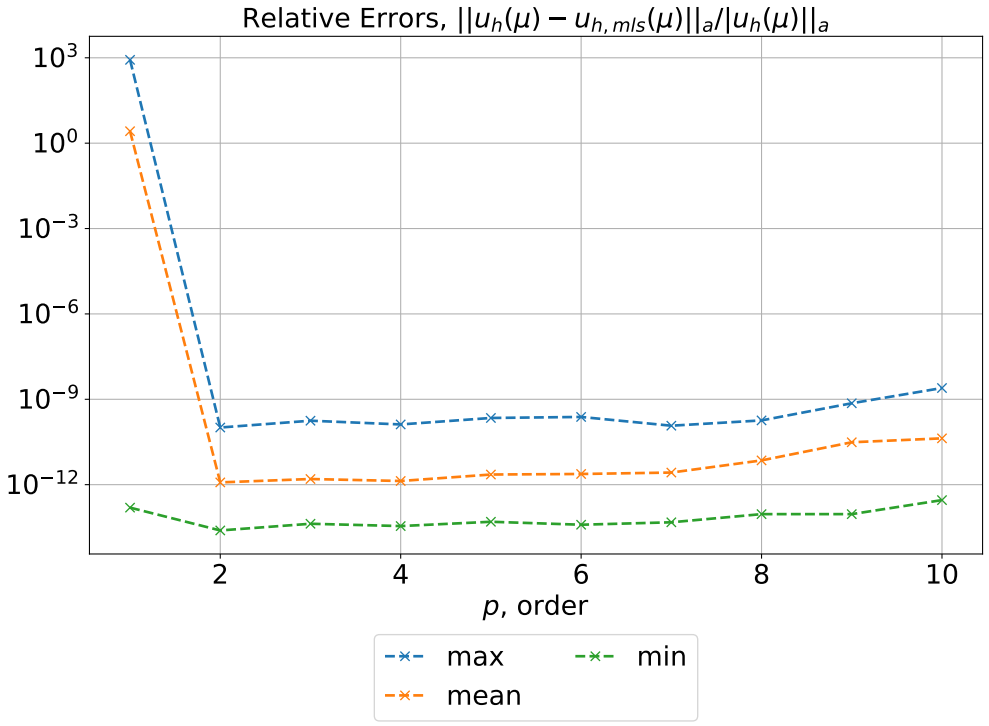


Figure 3.7. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and the geometry parameter range $\bar{G}_{\text{SR}} = (0.1, 5.1)$.

because we need to evaluate the integral

$$\int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v}\, d\Omega = \int_{\tilde{\Omega}} \boldsymbol{f}(\Phi) \cdot \tilde{\boldsymbol{v}} |J|\, d\tilde{\Omega}, \tag{3.85}$$

where the body force $\boldsymbol{f}$ is constant and $|J| = L_x L_y$ as shown in section 3.1.1. The increases in the errors from $p = 8$ until $p = 10$ can be explained by noise form the inverting of the matrix $M^\top M$ in Matrix Least Squares algorithm 3, since the condition numbers here are above 10^5, which we use as our noise limit.

3.5.1.2 The Relative Contribution per Term

From the previous section we note that order $p = 2$ is a good choice for the order, we study the relative contributions per term for the affine matrices, (2.181), $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ shown in figure 3.8.

Remark 3.14. *The terms in figure 3.8 are ordered as follows*

$$\left\{1, \frac{1}{L_y}, \frac{1}{L_x}, L_y, L_x, \frac{1}{L_x L_y}, \frac{L_y}{L_x}, L_y^2, \frac{L_x}{L_y}, L_x L_y, L_x^2, \frac{L_y^2}{L_x}, \frac{L_x^2}{L_y}\right\}, \tag{3.86}$$

and the polynomial terms and the numerator of the non-polynomial terms use Legendre polynomials shifted to $\bar{G}_{sr} = (0.1, 5.1)$. *This is not shown here for simplicity.*

Taking remark 3.14 into account, when studying the relative contributions per term in figure 3.8 we see that the terms 1, $\frac{L_y}{L_x}$ and $\frac{L_x}{L_y}$ are relevant as expected from (3.10). However, the terms $\frac{1}{L_x}$

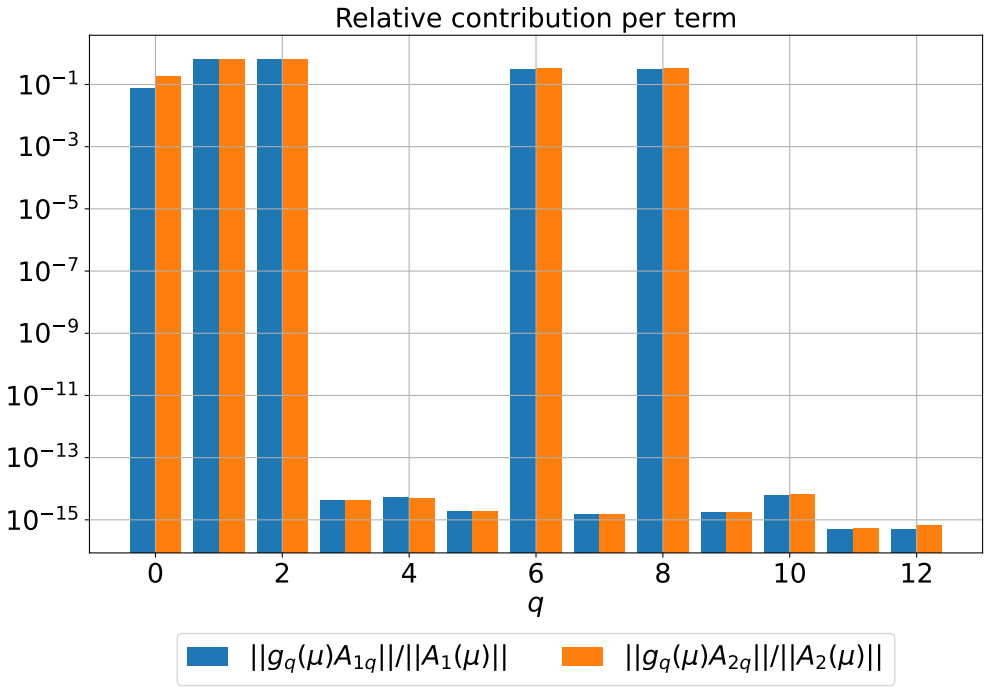


Figure 3.8. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The relative contribution per term solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{SR}} = (0.1, 5.1)$ and order $p = 2$. The terms are ordered as stated in remark 3.14.

and $\frac{1}{L_y}$ are also relevant. This is because of the use of shifted Legendre Polynomials which shifts the term $\frac{L_y}{L_x}$ to $\frac{L_y+k}{L_x}$ and therefore makes the term $k \cdot \frac{1}{L_x}$ relevant, and similar for $\frac{L_x}{L_y}$. From this we conclude that the Matrix Least Squares algorithm 3 does split the affine matrices $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ in the desired way for an affine problem.

3.5.1.3 The Relative Errors Between the High-fidelity Solution and the Reduced-order Solution

Finally we want to study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. For this we use the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2, with $\varepsilon_{\text{POD}} = 10^{-2}$ to capture at least 99.99% of the energy in the system. Studying the relative information content, $I(N)$, and the singular values in figures 3.9 and 3.10 respectively, we observe that the capture of at least 99.99% of the energy in the system is achieved for $N = 4$ singular values. We also observe that the singular values decrease rapidly until approximately $N = 75$ where the decrease flattens out.

Next we study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in figure 3.11. Here we observe that the mean error for $N = 2$ is a bit above 10^{-2}. However, more interesting is that the max and mean errors flatten out from around $N = 60$, where we observe that the max error flattens out at approximately 10^{-4} and the mean error at approximately 10^{-6}. Here we also observe that the min error preforms well and flattens out below 10^{-8}. Interestingly, when studying the log files from running the python scrip to plot the figures in this section, we observe that max error in general is achieved in the endpoints of the geometry parameter range $\bar{G}_{\text{SR}}$ for L_i.

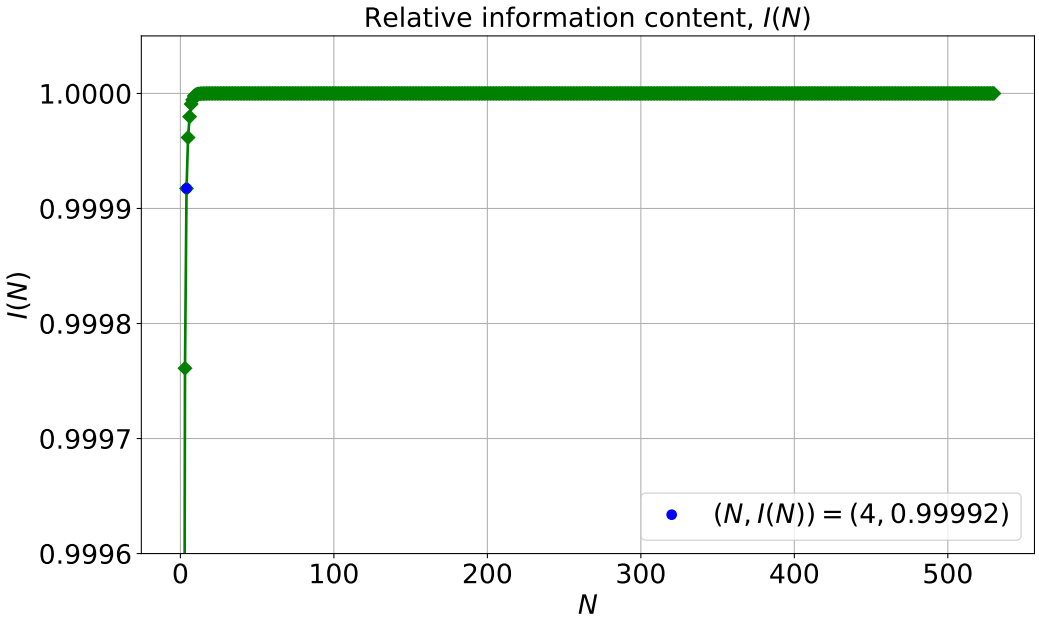


Figure 3.9. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The relative information content for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{SR}} = (0.1, 5.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

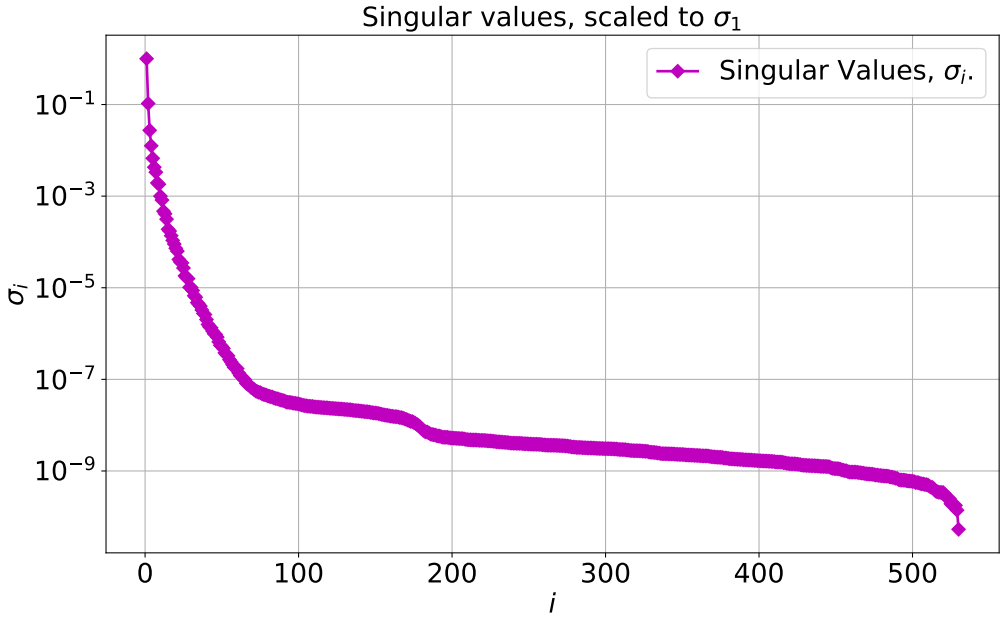


Figure 3.10. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The singular values for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{SR}} = (0.1, 5.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

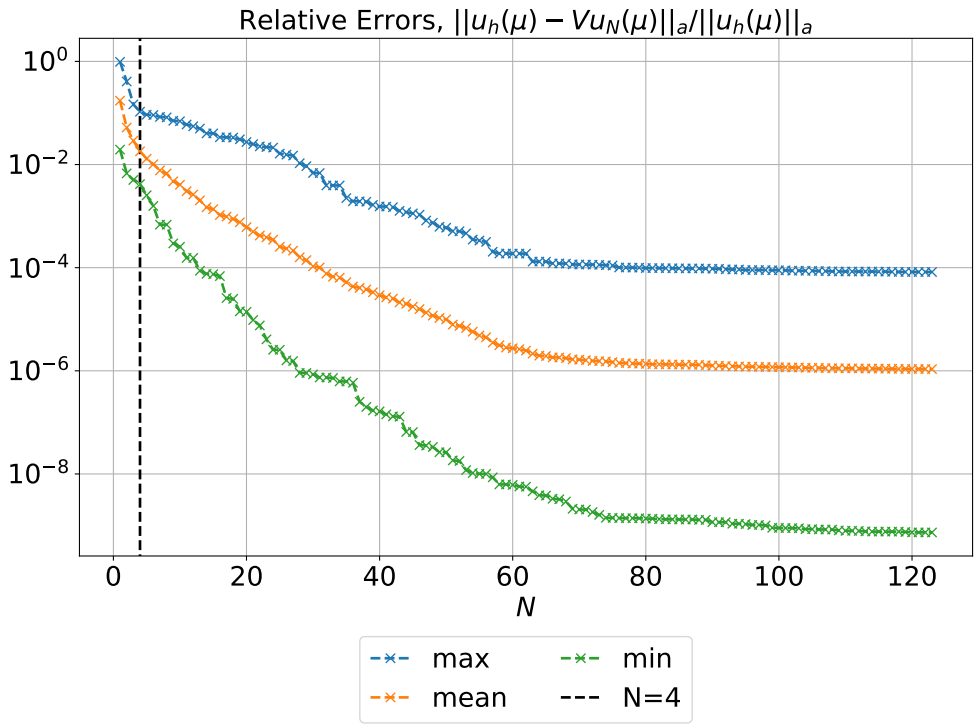


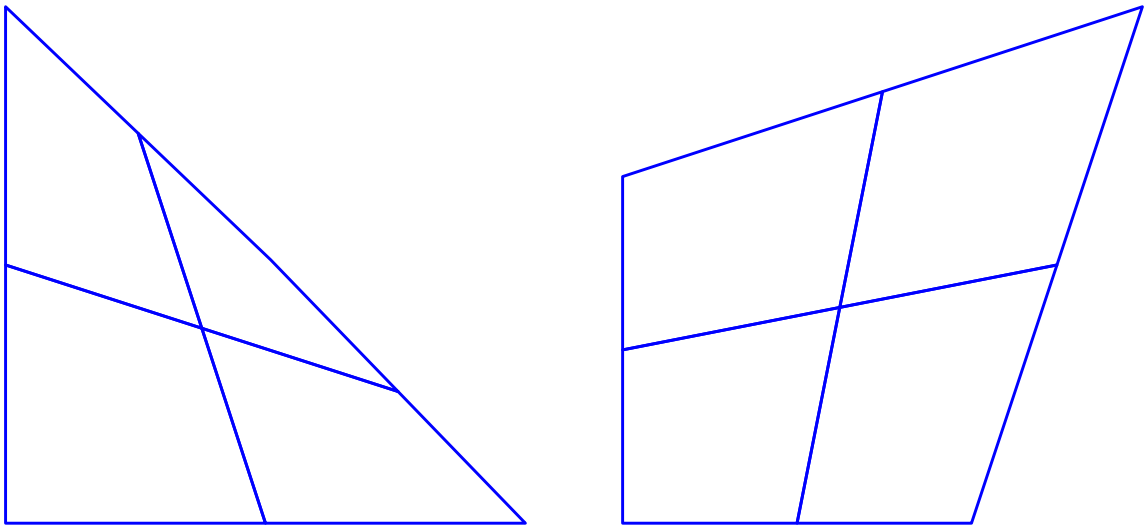
Figure 3.11. Testing the Matrix Least Squares algorithm: Case 1 — Scaling of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ for solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{SR}} = (0.1, 5.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The chosen $N = 4$ is marked by the black dashed line.

3.5.2 Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle

In this section we study the non-affine case of dragging one corner of a rectangle discussed in section 3.1.2 and the effect of the Matrix Least Squares algorithm 3. More specifically we study the problems that occur when the chosen geometry parameter range is too close to the singularities of the expansion, which in this case are the singularities at $\mu_i = \pm 0.5$, as seen in section 3.3.1.2. Therefore we use the geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$ where the two extremes of $\mu_1, \mu_2 = -0.49$ and $\mu_1, \mu_2 = 0.49$ are shown in figure 3.12 for $n = 2$ elements per axes. Looking at figure 3.12a especially, we see what will happen at the singularity of $\mu_1, \mu_2 = -0.5$ where we get a triangle.

3.5.2.1 The Relative Errors Between the High-fidelity Solution and the High-fidelity Matrix Least Squares Solution

Studying the relative errors in figure 3.13 we see that the max and mean errors are decreasing until order $p = 24$ and that the min error decreases until order $p = 18$, however, form here they increase quite rapidly. This is because for order $p = 25$ the matrix $M^\top M$ in Matrix Least Squares algorithm 3 can be considered singular since the condition number is above 10^{-17}. This is not a surprise since a 25×25 grid should only allow approximations up to order $p = 24$ since $n + 1$ snapshots in each direction are needed for order n as mentioned in remark 2.13. Furthermore, the increase in the min error from $p = 20$ can also be explained by the condition number of the matrix $M^\top M$, since it from here is higher than our noise limit of 10^5, and increases to above 10^{10}. Because of this we choose order $p = 19$ for further analysis. Another observation is that



(a) $\mu_1, \mu_2 = -0.49$ **(b)** $\mu_1, \mu_2 = 0.49$

Figure 3.12. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The two extremes of the geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$ for $\mu_1, \mu_2 = -0.49$ and $\mu_1, \mu_2 = 0.49$ for $n = 2$ elements per axes.

for order $p = 19$ the max and mean errors are right above 10^{-3} and around 10^{-4} respectively, which is a bit higher than we would have hoped for.

3.5.2.2 The Relative Contribution per Term

From the previous section we note that order $p = 19$ is a good choice for the order, we do a short study of the relative contributions per term for the affine matrices, (2.181), $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ shown in figure 3.14. Here we observe that the terms do decrease, but none of the terms are irrelevant in contrast to the affine case in section 3.5.1.2. Furthermore, the decrease may be a bit slow since relative contributions only decrease to below 10^{-4}.

3.5.2.3 The Relative Errors Between the High-fidelity Solution and the Reduced-order Solution

Lastly we again want to study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. For this we again use the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2, with $\varepsilon_{\text{POD}} = 10^{-2}$ to capture at least 99.99% of the energy in the system. Again studying the relative information content, $I(N)$, and the singular values in figures 3.15 and 3.16 respectively, we observe that the capture of at least 99.99% of the energy in the system is achieved for $N = 11$ singular values. We also observe that the singular values do not decrease as rapidly as in section 3.5.1.3, however, the decrease begins to flatten at approximately $N = 150$.

Now, studying the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in figure 3.17, we observe that the mean error for $N = 11$ is approximately 10^{-2}. However, more interesting, the max and the mean errors flatten out form $N = 50$. Here the max error flattens out above 10^{-3} and the mean error below 10^{-4}, which is quite close to the max and mean relative error between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ in section 3.5.2.1. This makes sense since we

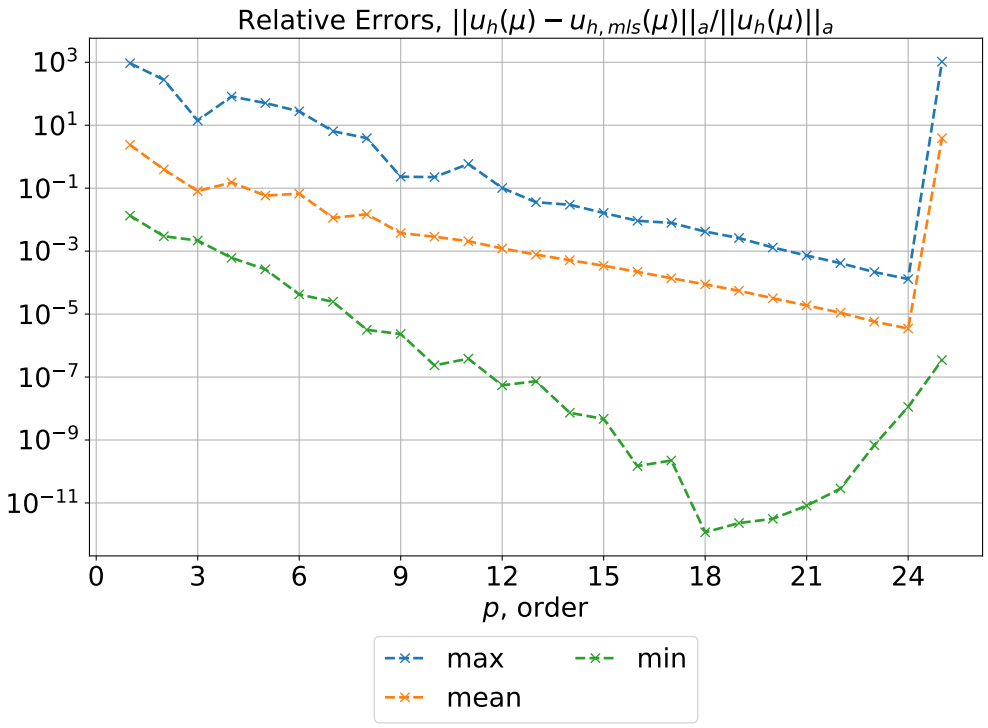


Figure 3.13. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and the geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$.

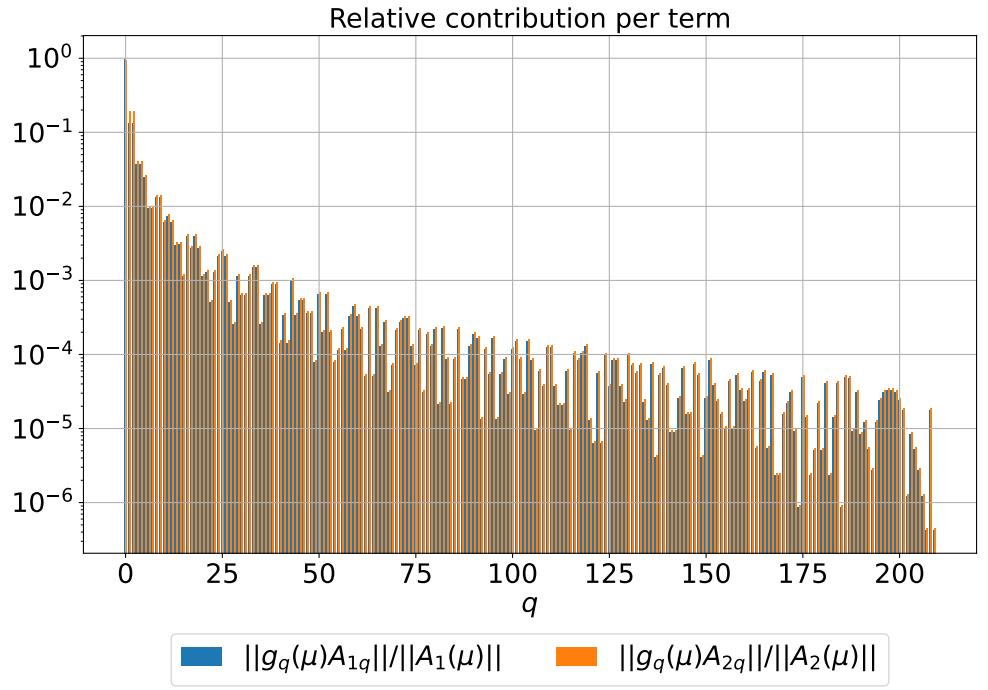


Figure 3.14. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative contribution per term solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and order $p = 19$.

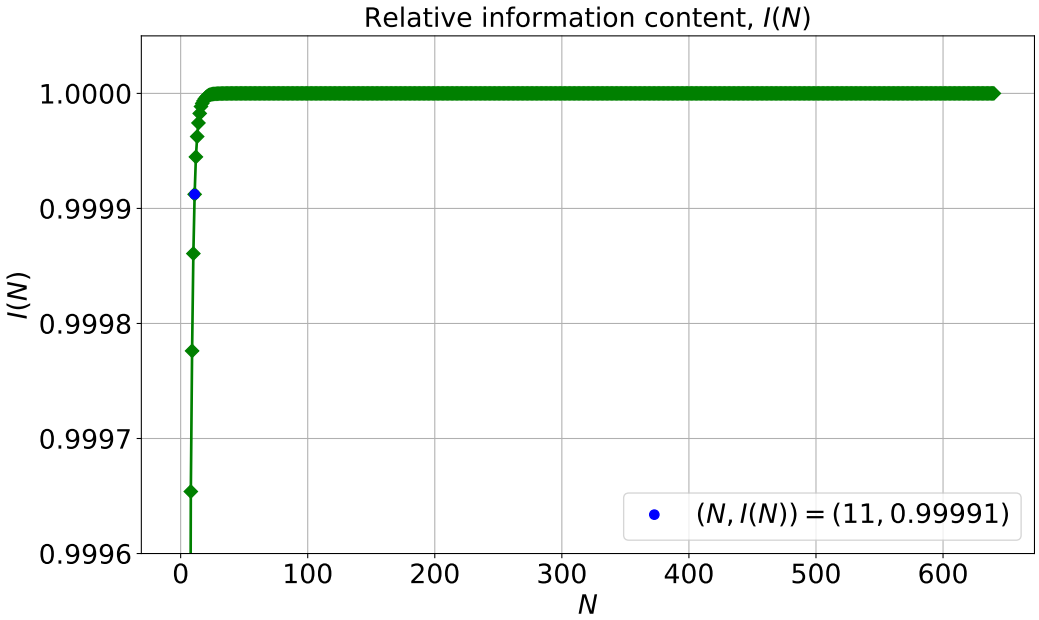


Figure 3.15. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative information content for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$, order $p = 19$ and and $\varepsilon_{\text{POD}} = 10^{-2}$.

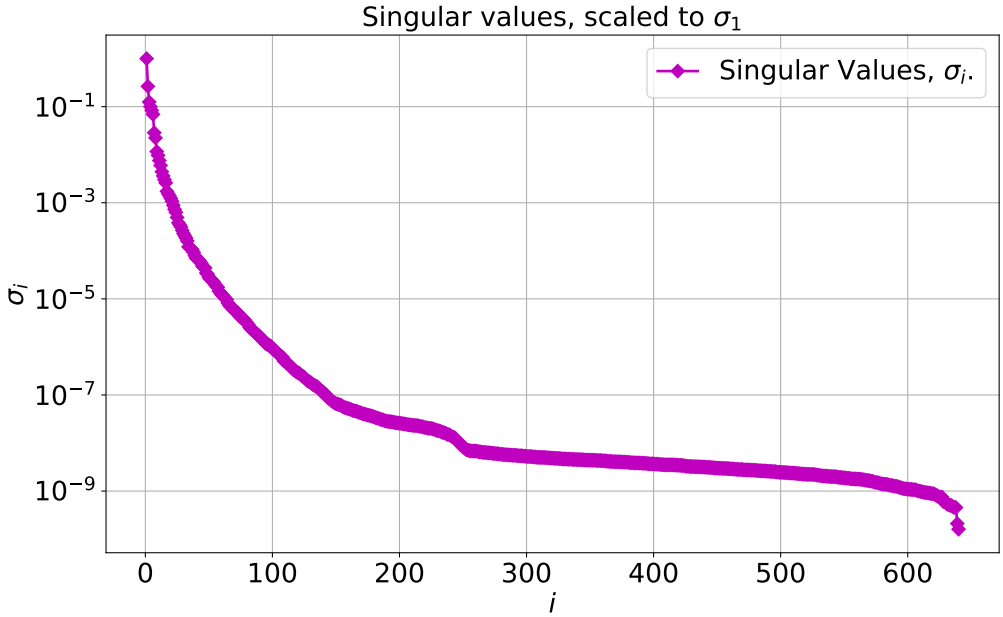


Figure 3.16. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The singular values for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$, order $p = 19$ and and $\varepsilon_{\text{POD}} = 10^{-2}$.

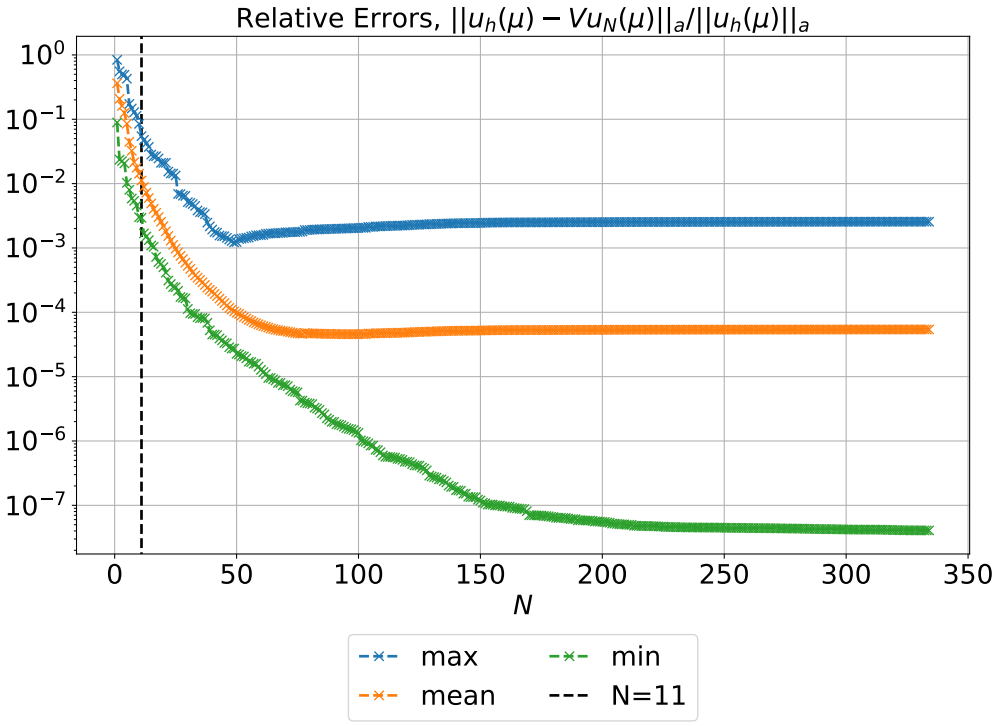


Figure 3.17. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ for solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes, geometry parameter range $\bar{G}_{\text{DR}} = (-0.49, 0.49)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The chosen $N = 11$ is marked by the black dashed line.

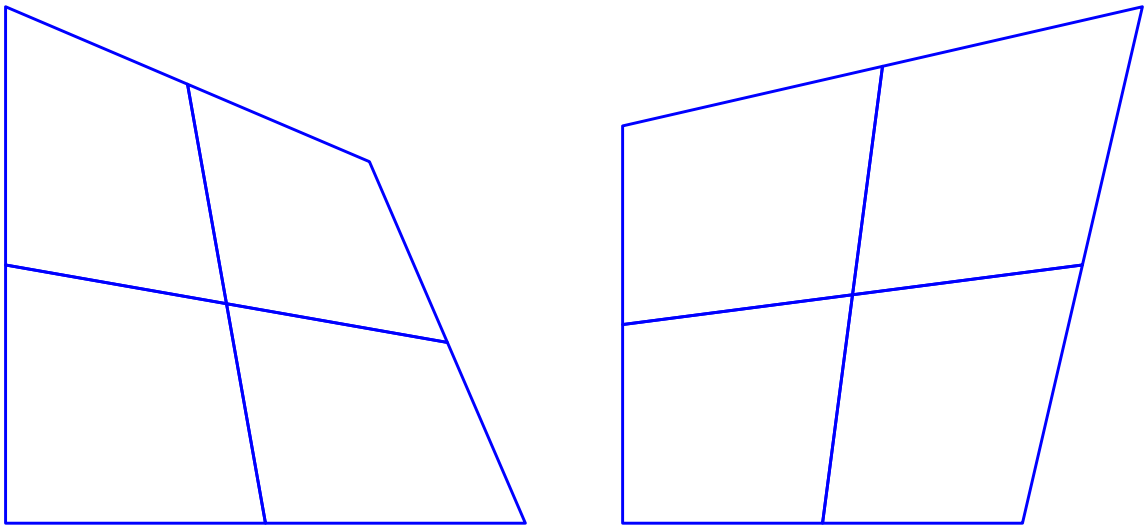
can estimate the error between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ by

$$\|u_h(\boldsymbol{\mu}) - V\boldsymbol{u}_N(\boldsymbol{\mu})\|_a \leq \|u_h(\boldsymbol{\mu}) - \boldsymbol{u}_{h,\text{mls}}(\boldsymbol{\mu})\|_a + \|\boldsymbol{u}_{h,\text{mls}}(\boldsymbol{\mu}) - V\boldsymbol{u}_N(\boldsymbol{\mu})\|_a, \tag{3.87}$$

using the triangle inequality. Moreover, again when studying the log files from running the python scrip to plot the figures in this section, we observe that max error in general is achieved in the endpoints of the geometry parameter range $\bar{G}_{\text{DR}}$ for μ_i. Noting this, we here hypothesise that the max and mean errors are dominated by the high-fidelity error.

3.5.2.4 Discussion

We have studied the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$, the relative contribution per term and the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in the previous sections. We see that our concerns around flattening out of the max and mean errors in section 3.5.2.1, and decrease of the relative contribution per term in section 3.5.2.2 result in the max and mean errors in section 3.5.2.3 being dominated by the high-fidelity error. Going back, we defined our geometry parameter range as $\bar{G}_{\text{DR}} = (-0.49, 0.49)$ in section 3.5.2 and noted that this is close to the singularities at $\mu_i = \pm 0.5$, as seen in section 3.3.1.2. Noting that Taylor expansions are most accurate when we are well within our maximum range, we conclude that we here may have been too close to the singularities and that smaller geometry ranges than the maximum ranges should be used.



(a) $\mu_1, \mu_2 = -0.3$ **(b)** $\mu_1, \mu_2 = 0.3$

Figure 3.18. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The two extremes of the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$ for $\mu_1, \mu_2 = -0.3$ and $\mu_1, \mu_2 = 0.3$ for $n = 2$ elements per axes.

3.5.3 Geometry Range Changes

Again making the note that Taylor expansions are most accurate when we are well within our maximum range, we want to make our ranges smaller. So we will use the range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$ for Case 2 — Dragging One Corner of a Rectangle, and $\bar{G}_{\text{QS}} = (-0.1, 0.1)$ instead of $G_{\text{QS}} = (-0.1\bar{6}, 0.1\bar{6})$.

Remark 3.15. *The maximum ranges for our geometry parameters in all three cases where found in section 3.3.*

3.5.4 Using a Smaller Geometry Range; Case 2 — Dragging One Corner of a Rectangle

In this section we again study the non-affine case of dragging one corner of a rectangle discussed in section 3.1.2 and the effect of the Matrix Least Squares algorithm 3. More specifically we study the problems that occur when the chosen geometry parameter range is well within the our maximum range as mentioned in section 3.5.3. Therefore we use the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$ where the two extremes of $\mu_1, \mu_2 = -0.3$ and $\mu_1, \mu_2 = 0.3$ are shown in figure 3.18 for $n = 2$ elements per axes.

3.5.4.1 The Relative Errors Between the High-fidelity Solution and the High-fidelity Matrix Least Squares Solution

Studying the relative errors in figure 3.19 we see that the max and mean errors are decreasing until order $p = 19$ and that the min error decreases until order $p = 18$, however, form here they again increase quite rapidly, even more so than in figure 3.13. Again, this is because for order $p = 25$ the matrix $M^\top M$ in Matrix Least Squares algorithm 3 can be considered singular since the condition number is above 10^{-17}. The increase from $p = 20$ can be explained by the condition number of the matrix $M^\top M$, since it from here is higher than our noise limit of 10^5,

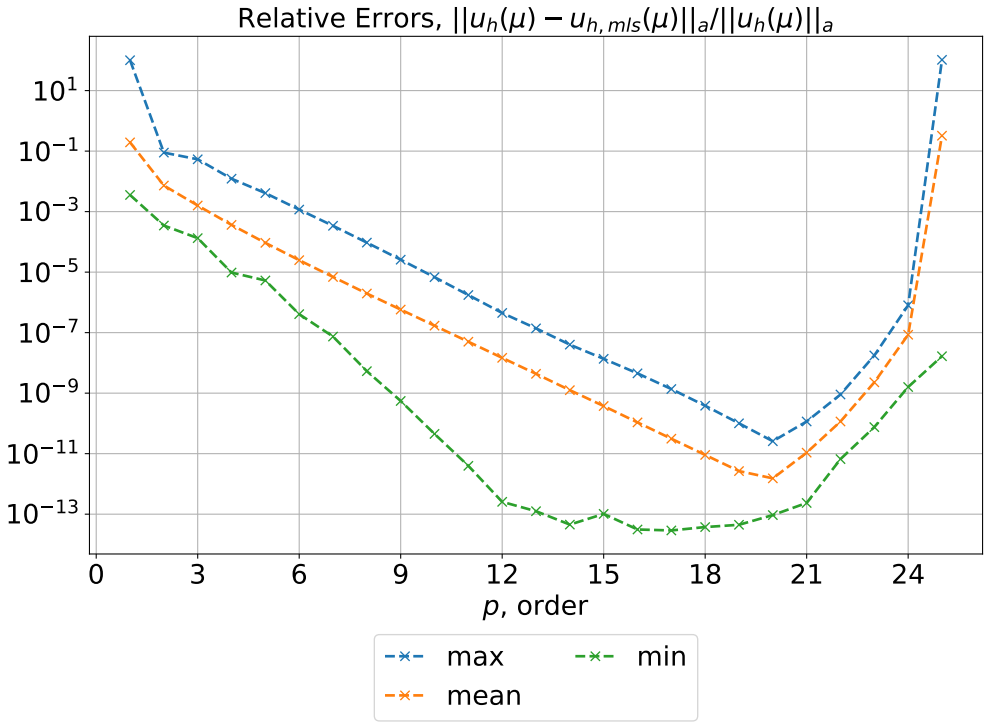


Figure 3.19. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$.

and increases to above 10^{10}. Because of this we again choose order $p = 19$ for further analysis. Another observation here is that for order $p = 19$ the max and mean errors are right above 10^{-11} and right below 10^{-11} respectively, which is really small and considerably lower than what we observed before in section 3.5.2.1.

3.5.4.2 The Relative Contribution per Term

From the previous section we note that order $p = 19$ is a good choice for the order, we again do a short study of the relative contributions per term for the affine matrices, (2.181), $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ shown in figure 3.20. Here we observe that the terms do decrease, and we have some terms, mostly after approximately $q = 150$, that are close to irrelevant. Furthermore, the decrease is better than in section 3.5.2.2 since relative contributions decrease to below 10^{-10}.

3.5.4.3 The Relative Errors Between the High-fidelity Solution and the Reduced-order Solution

Finally we again want to study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. For this we again use the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2, with $\varepsilon_{\text{POD}} = 10^{-2}$ to capture at least 99.99% of the energy in the system. Again studying the relative information content, $I(N)$, and the singular values in figures 3.21 and 3.22 respectively, we observe that the capture 99.99% of the energy in the system is achieved for $N = 8$ singular values and not $N = 11$ as in section 3.5.2.3. We observe that the singular values decrease as in section 3.5.2.3, however, the decrease begins to flatten at approximately $N = 100$.

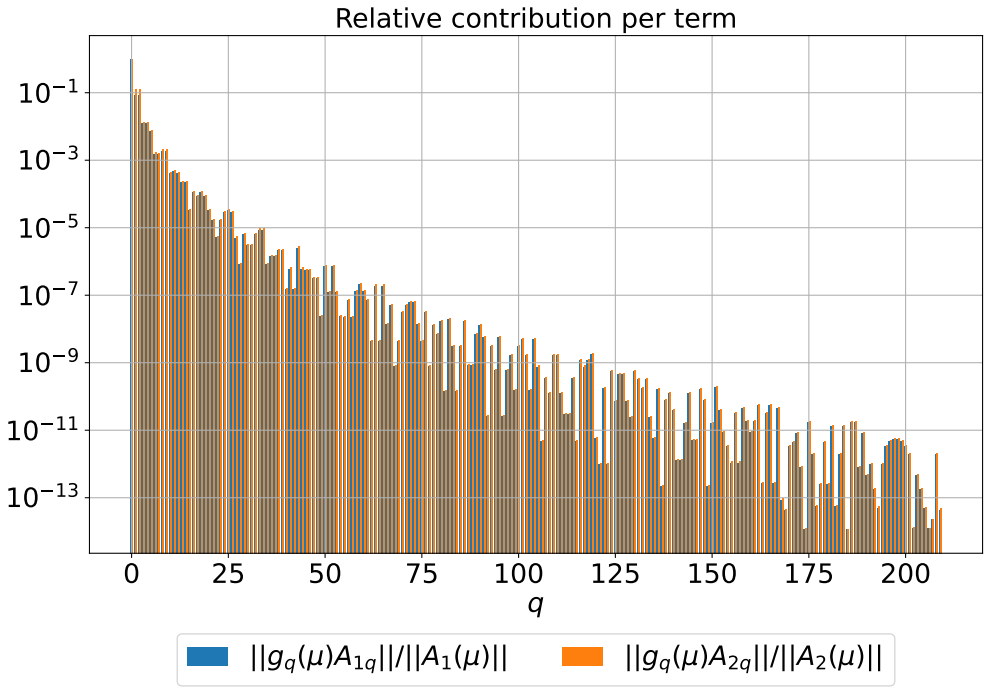


Figure 3.20. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative contribution per term solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and order $p = 19$.

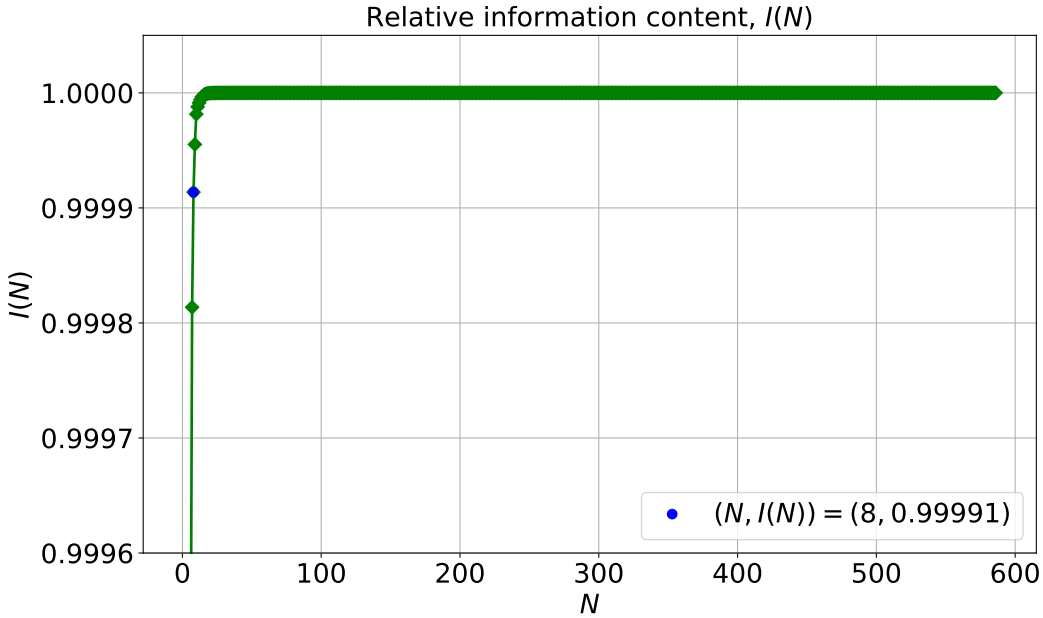


Figure 3.21. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative information content for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

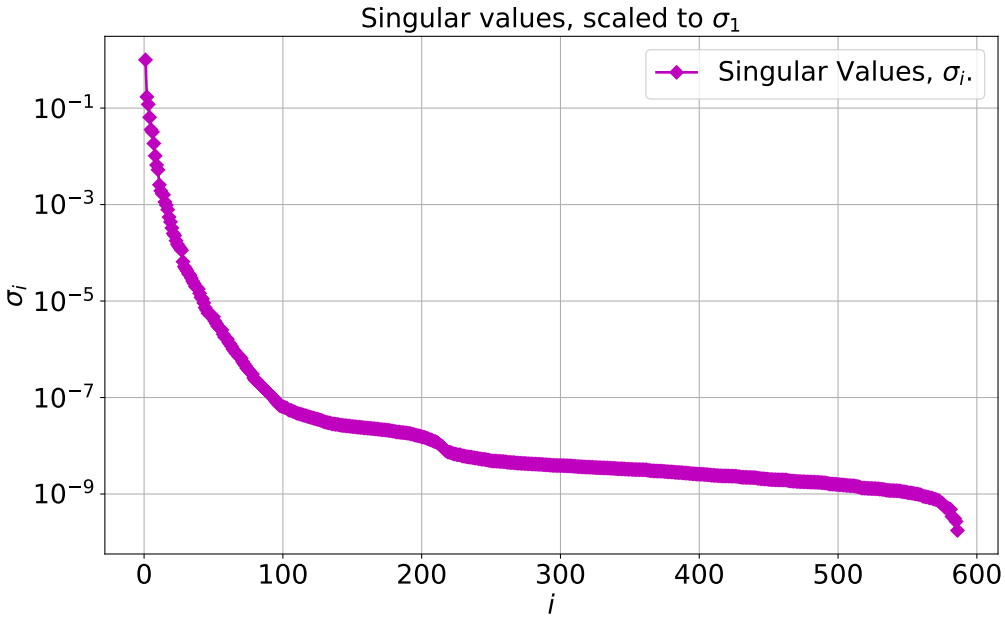


Figure 3.22. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The singular values for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

Now, again studying the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in figure 3.23, we observe that the mean error for $N = 8$ is approximately 10^{-2}, i.e. the same as we had for $N = 11$ in section 3.5.2.3. However, more interesting, the max and the mean errors flatten out form $N = 100$, and the max error flattens out right below 10^{-6} and the mean error right below 10^{-7}, which is much better than in figure 3.17 from section 3.5.2.3. Moreover, as in section 3.5.2.3, when studying the log files from running the python scrip to plot the figures in this section, we observe that max error in general is achieved in the endpoints of the geometry parameter range $\bar{G}_{\text{DR}}$ for μ_i. Noting this, we confirm our hypothesis from section 3.5.2.3 about the max and mean errors being dominated by the high-fidelity error.

3.5.4.4 Discussion

During the study of this example and though the comparison with the previous example in section 3.5.2, we conclude that the use of a smaller geometry parameter range, i.e. a range well within the maximum geometry parameter range form section 3.3, is impor.e.tant for the performance of the Matrix Least Square algorithm 3. This is because otherwise the relative error high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ may be dominated by the relative error between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ as hypothesised in section 3.5.2.3 and confirmed in section 3.5.4.3

3.5.5 Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle

In this section we study the non-affine case of dragging all corners of a rectangle discussed in section 3.1.3 and the effect of the Matrix Least Squares algorithm 3. More specifically we study the problems that occur when the chosen geometry parameter range is well within our maximum range as mentioned in section 3.5.3. Therefore we use the geometry parameter range

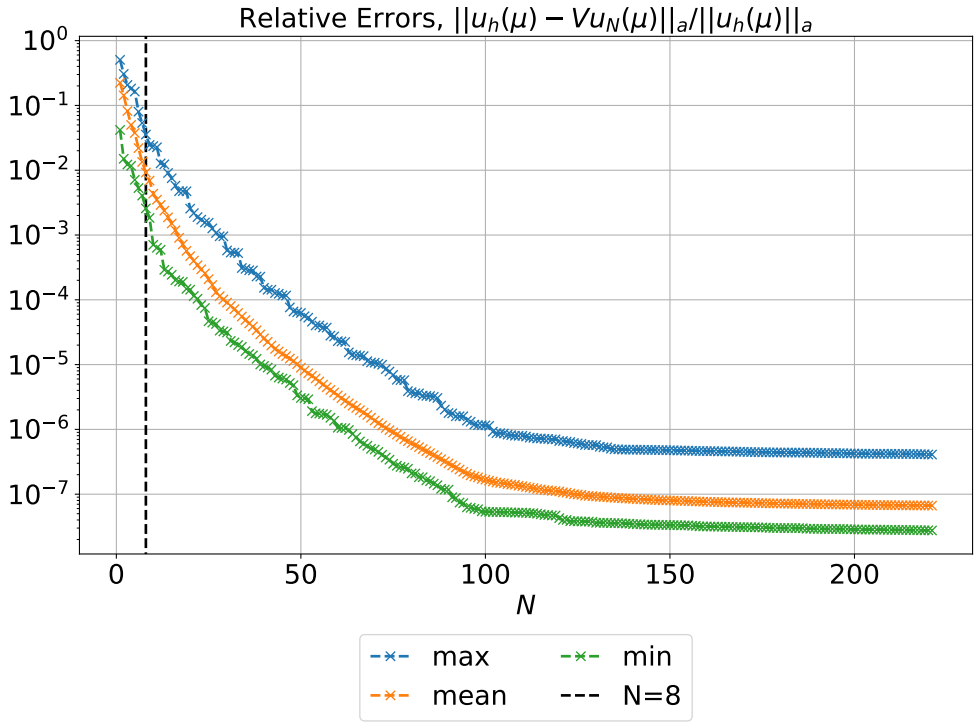
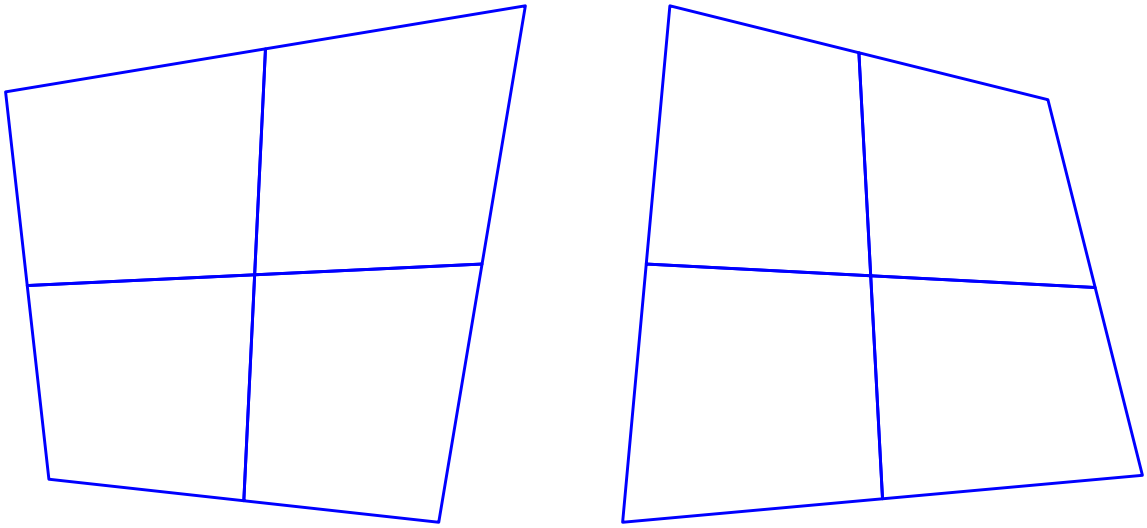


Figure 3.23. Problems with the Geometry Range; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ for solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The chosen $N = 8$ is marked by the black dashed line.

$\bar{G}_{\text{Qs}} = (-0.1, 0.1)$ where two of the extremes are shown in figure 3.24 for $n = 2$ elements per axes using of $\mu_1, \mu_2, \mu_5, \mu_6 = -0.1$ and $\mu_3, \mu_4 = 0.1$ for the left figure, and $\mu_1, \mu_2, \mu_5, \mu_6 = 0.1$ and $\mu_3, \mu_4 = -0.1$ for the right figure.

3.5.5.1 The Relative Errors Between the High-fidelity Solution and the High-fidelity Matrix Least Squares Solution

Studying the relative errors in figure 3.25, we see that the max and mean errors are decreasing until order $p = 2$ before increasing. This is again because for order $p = 3$ the matrix $M^\top M$ in Matrix Least Squares algorithm 3 can be considered singular since the condition number is above 10^{-17}. This is again caused by having 3 snapshots in each of our 6 directions, meaning that order $p = 2$ is the best order we can achieve, noting remark 2.13. Order $p = 2$ is less than we would have wished for, however, increasing the number of snapshots from $3^6 = 729$ to $5^6 = 15\,625$ or more will give longer computational times as motioned in section 3.5. Because of this we are again choosing order $p = 2$ for further analysis here. Another observation is that for order $p = 2$ the max and mean errors are right above 10^{-2} and right above 10^{-3} respectively, which is still is quite large and may lead to the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ dominates the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. Just as previously seen from the study in section 3.5.2.



(a) $\mu_1, \mu_2, \mu_5, \mu_6 = -0.1$ and $\mu_3, \mu_4 = 0.1$ **(b)** $\mu_1, \mu_2, \mu_5, \mu_6 = 0.1$ and $\mu_3, \mu_4 = -0.1$

Figure 3.24. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; Two of the extremes for the geometry parameter range $\bar{G}_{\text{QS}} = (-0.1, 0.1)$ for $n = 2$ elements per axes using of $\mu_1, \mu_2, \mu_5, \mu_6 = -0.1$ and $\mu_3, \mu_4 = 0.1$ for the left figure, and $\mu_1, \mu_2, \mu_5, \mu_6 = 0.1$ and $\mu_3, \mu_4 = -0.1$ for the right figure.

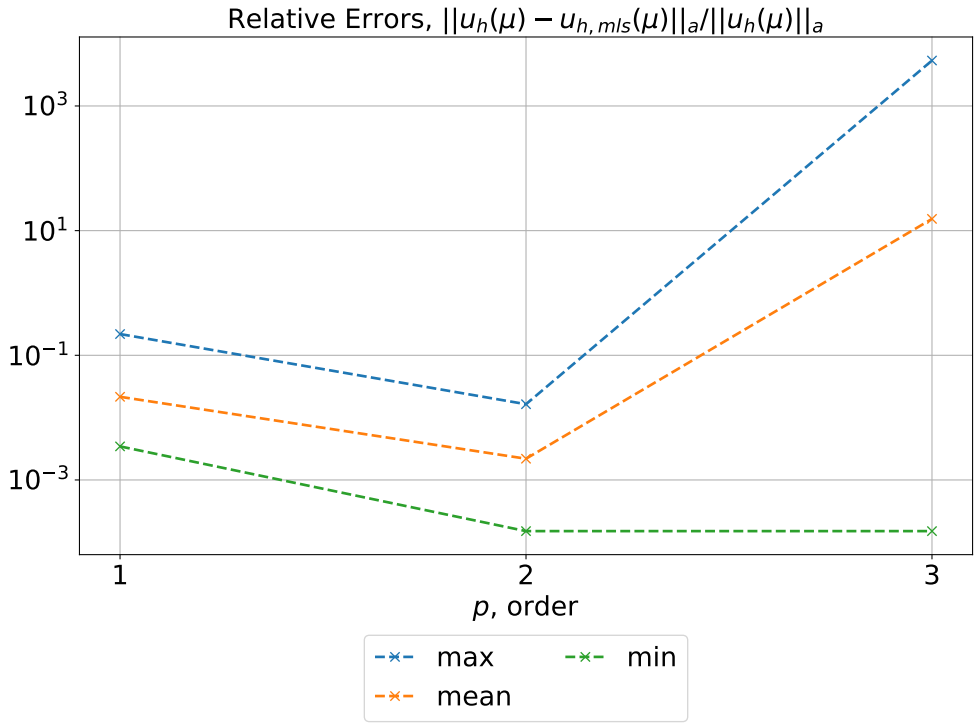


Figure 3.25. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and the geometry parameter range $\bar{G}_{\text{QS}} = (-0.1, 0.1)$.

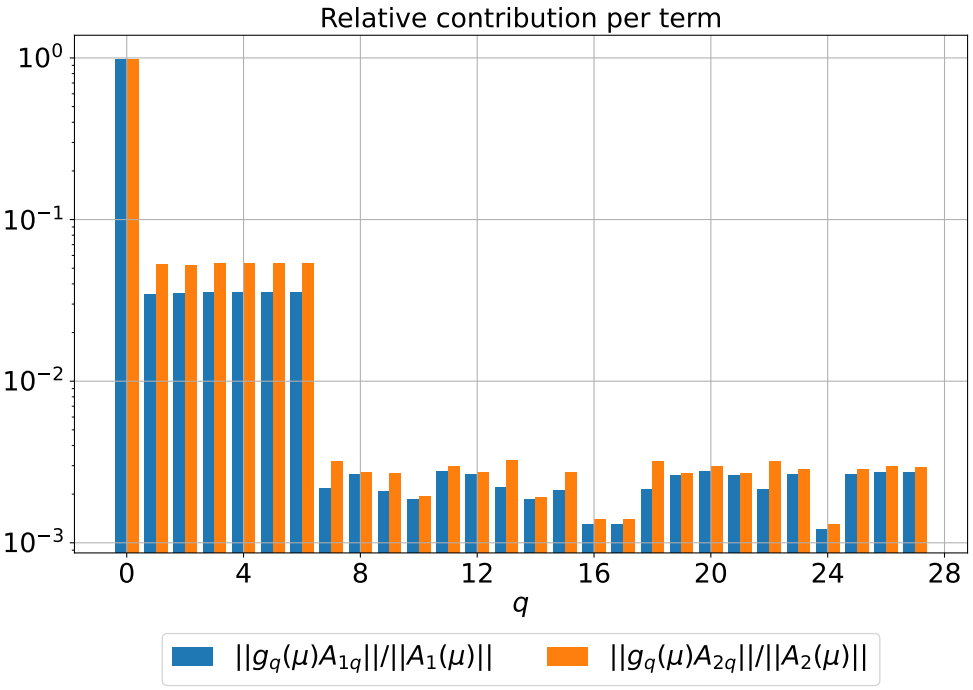


Figure 3.26. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; The relative contribution per term solving the problem of Constant Body force in 2D using $n = 20$ elements along the axes and order $p = 19$.

3.5.5.2 The Relative Contribution per Term

Having chosen $p = 2$ in the previous section we again do a short study of the relative contributions per term for the affine matrices, (2.181), $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ shown in figure 3.26. Here we observe that the terms do decrease, however none are close to irrelevant. Furthermore, the decrease happens in distinct steeps after $q = 0$ and $q = 6$. Looking at the log files we see that the step after $q = 1$ is the barrier between 0-order term 1 and the 1-order terms, and that the step after $q = 6$ is the barrier between the 1-order terms and 2-order terms. The fact that the relative contribution of all terms is above 10^{-3} again indicates that the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ will dominate the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ because of too few snapshots.

3.5.5.3 The Relative Errors Between the High-fidelity Solution and the Reduced-order Solution

Finally we again want to study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. For this we once more use the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2, with $\varepsilon_{\text{POD}} = 10^{-2}$ to capture at least 99.99% of the energy in the system. Studying the relative information content, $I(N)$, and the singular values in figures 3.27 and 3.28 respectively, we observe that the capture of at least 99.99% of the energy in the system is achieved for $N = 11$ singular values. We also observe that the singular values decrease slower than in section 3.5.2.3, and the decrease begins to flatten at approximately $N = 200$.

Now, studying the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in figure 3.29, we observe that the mean error for $N = 11$

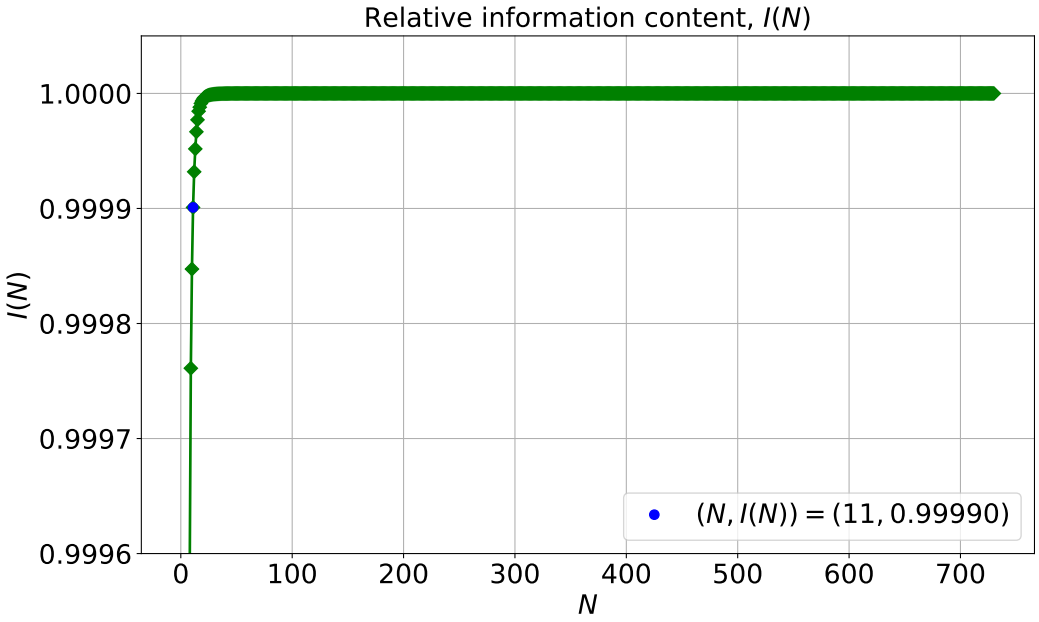


Figure 3.27. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; The relative information content for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{QS}} = (-0.1, 0.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

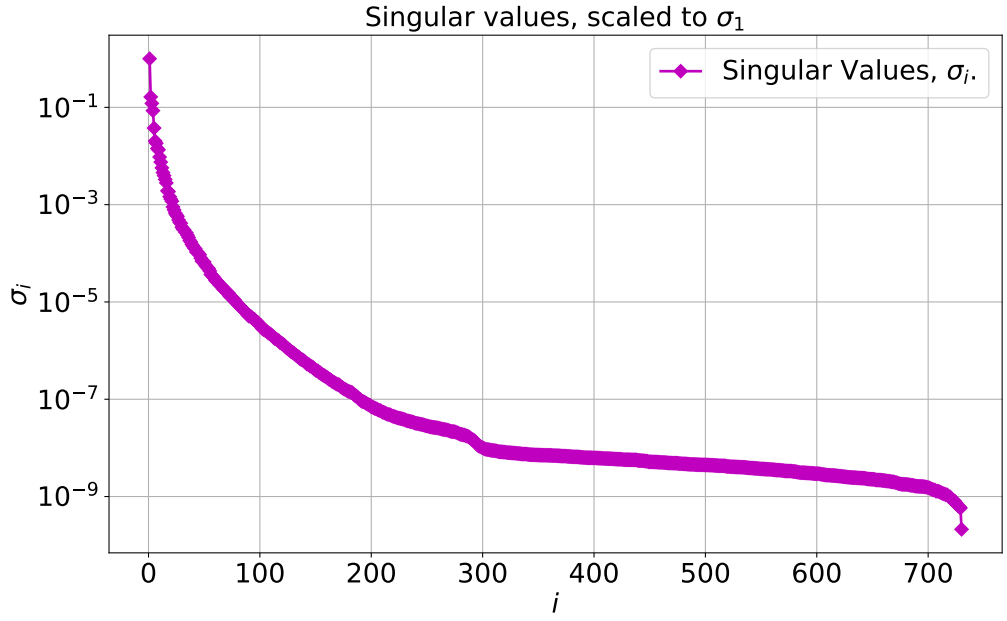


Figure 3.28. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; The singular values for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{QS}} = (-0.1, 0.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

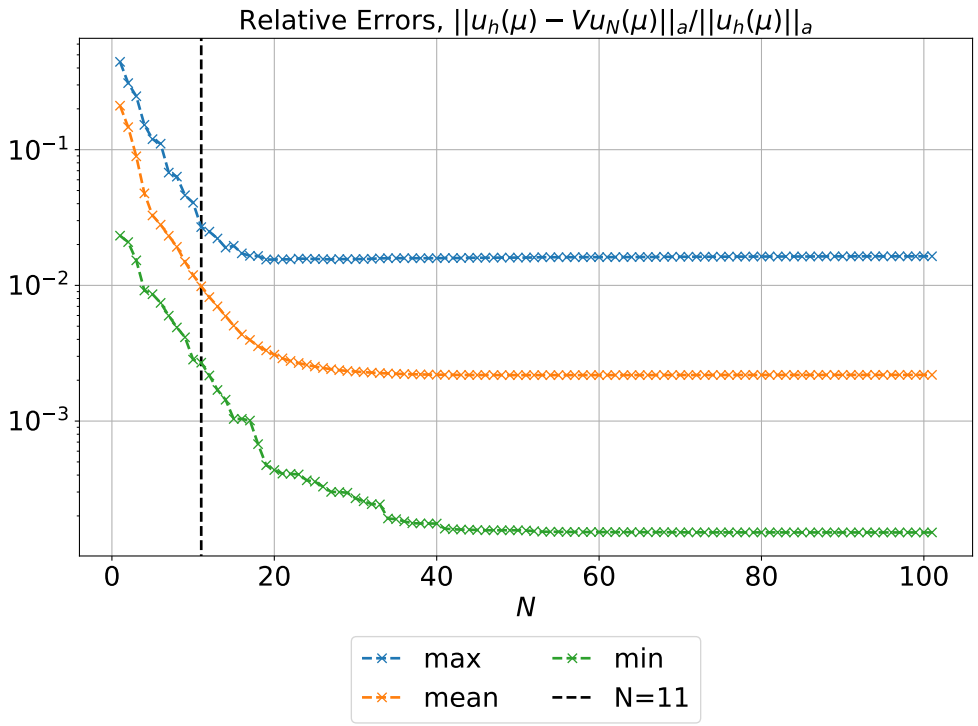


Figure 3.29. Too few Snapshots; Case 3 — Dragging All Corners of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 20$ elements along the axes, the geometry parameter range $\bar{G}_{\text{QS}} = (-0.1, 0.1)$, order $p = 2$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The chosen $N = 11$ is marked by the black dashed line.

is approximately 10^{-2}. However, more interesting, the max and the mean errors flatten out form $N = 20$ and here the max error flattens out above 10^{-2} and the mean error below 10^{-2}, which is quite bad. Taht is close to the max and mean errors for the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ in section 3.5.5.1. Moreover, when studying the log files from running the python scrip to plot the figures in this section, we observe that max error in general is achieved in the endpoints of the geometry parameter range $\bar{G}_{\text{QS}}$ for μ_i. Noting this we confirm our hypothesis from section 3.5.5.1 and section 3.5.5.2 about the max and mean errors in figure 3.17 being dominated by the high-fidelity errors.

3.5.5.4 Discussion

During the study of this example we concluded that the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ are dominated by the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ because we have too few snapshots. We also observed the effect of this in figure 3.26 where we had two distinct steps showing the decrease in relevance of the terms as the order p of the term gets larger. From this we see that too few snapshots is a problem to take into account.

3.5.6 More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle

In this section we do a final analysis on the case of dragging one corner of a rectangle. From the studies in section 3.5.2 and section 3.5.4 we learned that order $p = 19$ is a good choice for

	max	*mean*	*min*
$\frac{\|u_h(\boldsymbol{\mu})-\boldsymbol{u}_{h,\text{mls}}(\boldsymbol{\mu})\|_a}{\|u_h(\boldsymbol{\mu})\|_a}$	$1.2 \cdot 10^{-10}$	$3.3 \cdot 10^{-12}$	$2.7 \cdot 10^{-13}$

Table 3.5. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ solving the problem of Constant Body force in 2D using $n = 90$ elements along the axes and the geometry parameter range $\bar{G}_{\text{QS}} = (-0.3, 0.3)$ for order $p = 19$.

the order of approximation for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$. Furthermore, we observed that the geometry parameter range $\bar{G}_{\text{DR}}$ should bee chosen well within the maximum geometry parameter range mentioned in section 3.3. Therefore we will use $\bar{G}_{\text{DR}} = (-0.3, 0.3)$. Next, as mention in section 3.5, we use $n = 90$ elements along the axes to get $N_h = 16\,380$ degrees of freedom (dofs) or free nodes which is more than $n_s = 15\,625 = 25 \times 25 \times 5 \times 5$, which is the number of solutions in the snapshot matrix (2.74). Since we have chosen order $p = 19$ for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ we present the max, mean and min relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ in table 3.5. Here we see that errors are approximately equal to the errors for $p = 19$ in figure 3.19 in section 3.5.4.1.

3.5.6.1 The Relative Errors Between the High-fidelity Solution and the Reduced-order Solution

Having chosen order $p = 19$ for the Matrix Least Squares functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ we now study the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. For this we again use the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2 with $\varepsilon_{\text{POD}} = 10^{-2}$ to capture at least 99.99% of the energy in the system. Studying the relative information content, $I(N)$, and the singular values in figures 3.30 and 3.31 respectively, we observe that the capture of at least 99.99% of the energy in the system is achieved for $N = 8$ singular values, as in section 3.5.4.3. This gives us an approximate reduction factor of 2 000 in the degrees of freedom. We also observe that we have more singular values than for $n = 20$ elements along the axes and that they decrease rapidly until 10^{-9} for approximately $N = 200$, before beginning to flatten out.

Remark 3.16. *In all figures of the relative information and the singular values all singular values larger than* 0 *from the Proper Orthogonal Decomposition (POD) algorithm with the energy norm, algorithm 2, have been plotted.*

Studying the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ in figure 3.32, we observe that the mean error for $N = 8$ is approximately 10^{-2}. More interesting, the max and the mean errors decrease slower than in figure 3.23. However, in the end at $N = 200$ the errors are lower, i.e right below 10^{-8} versus right below 10^{-6}. This leads to the conclusion that more degrees of freedom slows down the decrease of the errors, but in the end gives lower errors. Furthermore, when studying the log files from running the python script to plot the figures we observe that max error in general is achieved in the endpoints of the geometry parameter range $\bar{G}_{\text{DR}}$ for μ_i.

Finally, studying the four POD modes in figures 3.33 and 3.34 we notice that the deformations get smaller and smaller in the zoomed-in plots to the left. This corresponds nicely to the reduction in the singular values. Furthermore, we observe that the first mode is traction free, whereas the following modes all display some traction.

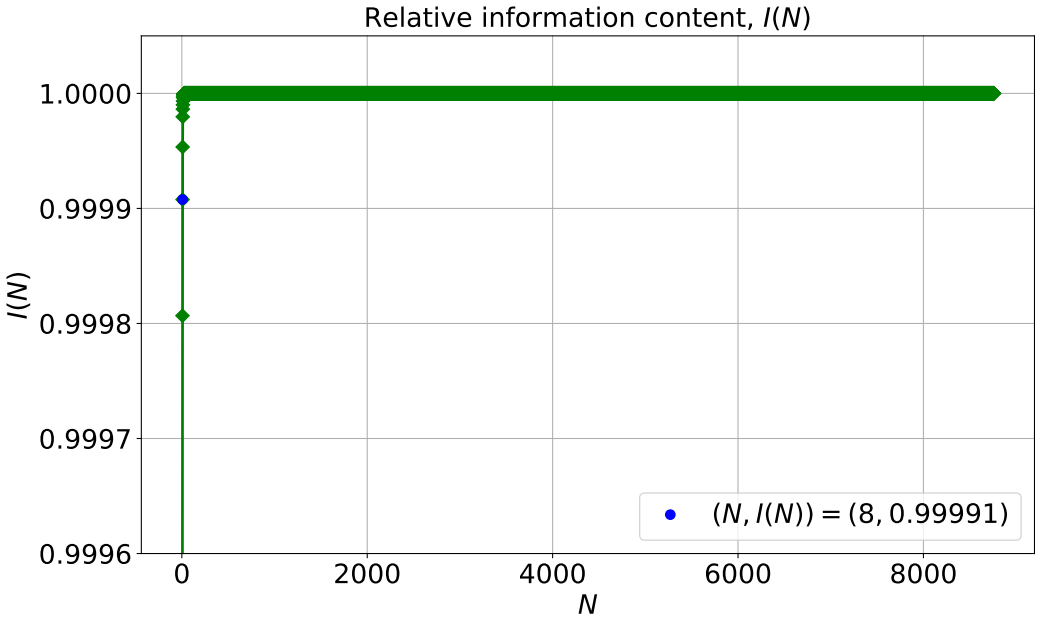


Figure 3.30. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The relative information content for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

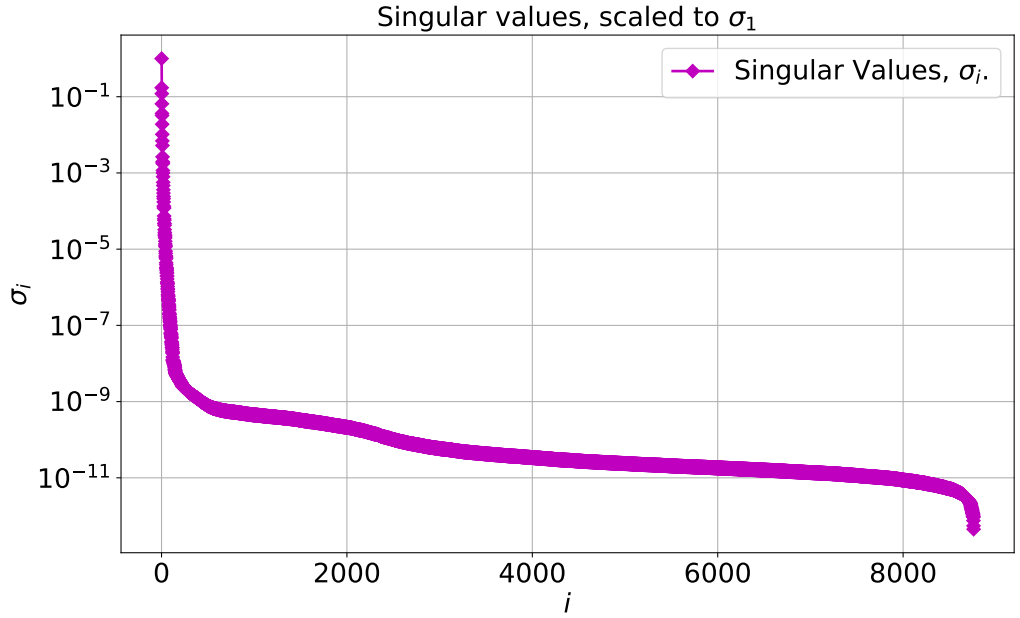


Figure 3.31. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The singular values for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$.

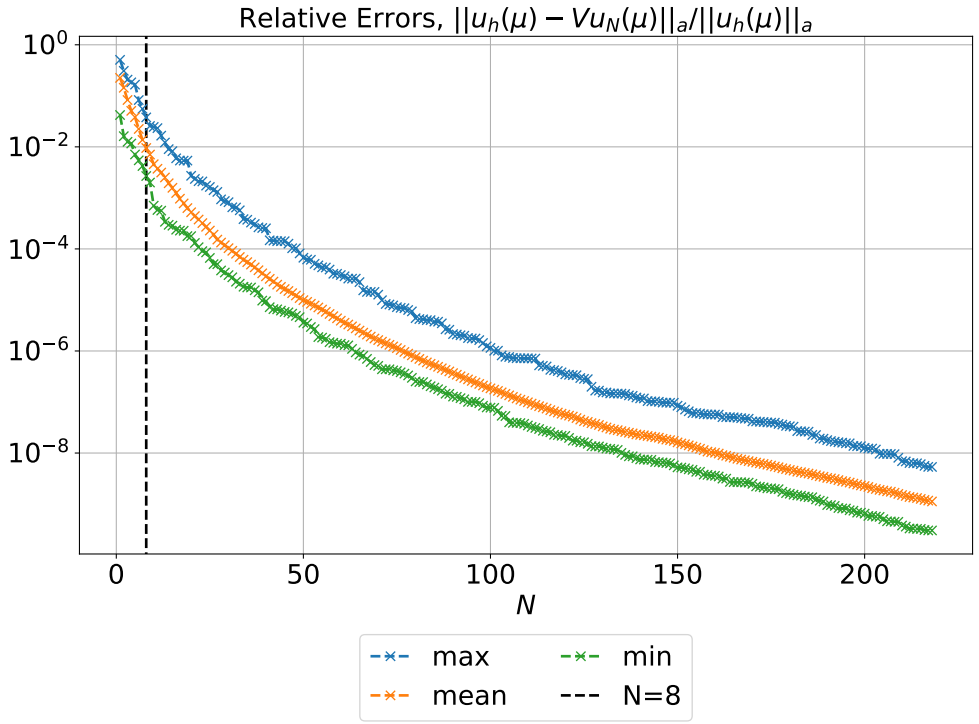


Figure 3.32. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The chosen $N = 8$ is marked by the black dashed line.

3.5.6.2 Displacement and Recovered von Mises Stress

In this section we want to study the displacement and recovered von Mises stress in the high-fidelity (HF) solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order (RB) solution $Vu_N(\boldsymbol{\mu})$. We chose to use $\mu_1 = 0.2$ and $\mu_2 = -0.2$. We get the recovered von Mises stress by doing simple *stress recovery*, inspired by [19]. This is done by first getting the stress at the nodal values, by averaging the stresses over all neighbouring elements, and then interpolate linearly within each element using the basis functions for the displacement. With the recovered stress $\boldsymbol{\sigma}$ we compute the recovered von Mises stress $\boldsymbol{\sigma}_{\text{M}}$ as

$$\boldsymbol{\sigma}_{\text{M}} = \sqrt{\frac{3}{2}\boldsymbol{s} : \boldsymbol{s}}, \tag{3.88}$$

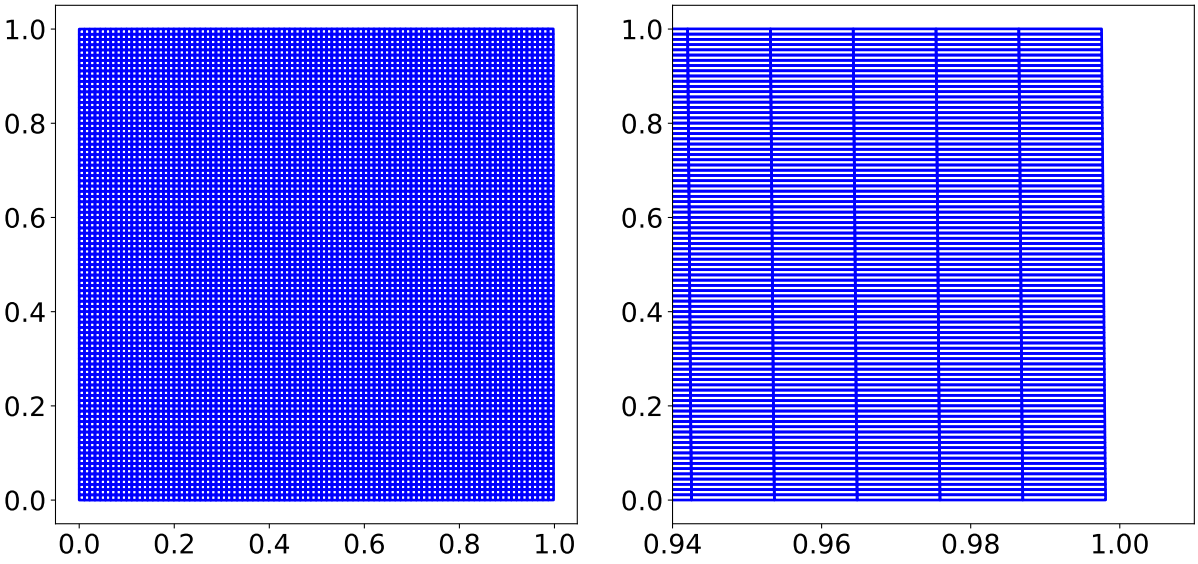
where $\boldsymbol{s}$ is the deviatoric stress tensor

$$\boldsymbol{s} = \boldsymbol{\sigma} - \frac{1}{3}\operatorname{tr}(\boldsymbol{\sigma})I, \tag{3.89}$$

as in [20].

Studying the displacement plots in figures 3.35 and 3.36 respectively, we do not see differences. Making us conclude that the RB solution for $N = 8$ will approximate the HF solution quite good for $N = 8$ POD modes.

Next, studying the recovered von Mises stress in figures 3.37 and 3.38 respectively, we see a small difference. Here the orange and yellow colored areas in the HF plot reach higher on the y-axis than in the RB plot, i.e. the HF plot has yellow above 0.4 and orange closer to 0.2 than the RB plot. Besides this the plots are pretty similar.



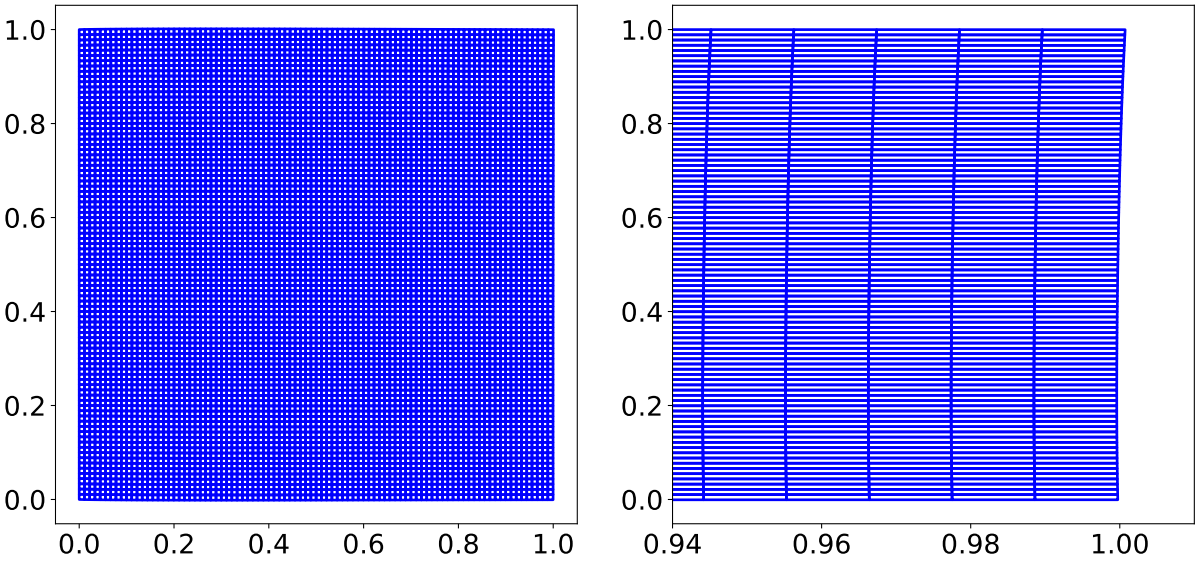
(a) POD mode — $N = 1$

(b) POD mode — $N = 1$ zoomed in

(c) POD mode — $N = 2$

(d) POD mode — $N = 2$ zoomed in

Figure 3.33. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The two first POD modes for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The plots to the left show the whole picture, whereas the plots to the right zoom in at the end $x = 1$.



(a) POD mode — $N = 3$

(b) POD mode — $N = 3$ zoomed in

(c) POD mode — $N = 4$

(d) POD mode — $N = 4$ zoomed in

Figure 3.34. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The third and fourth POD modes for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$ and $\varepsilon_{\text{POD}} = 10^{-2}$. The plots to the left show the whole picture, whereas the plots to the right zoom in at the end $x = 1$.

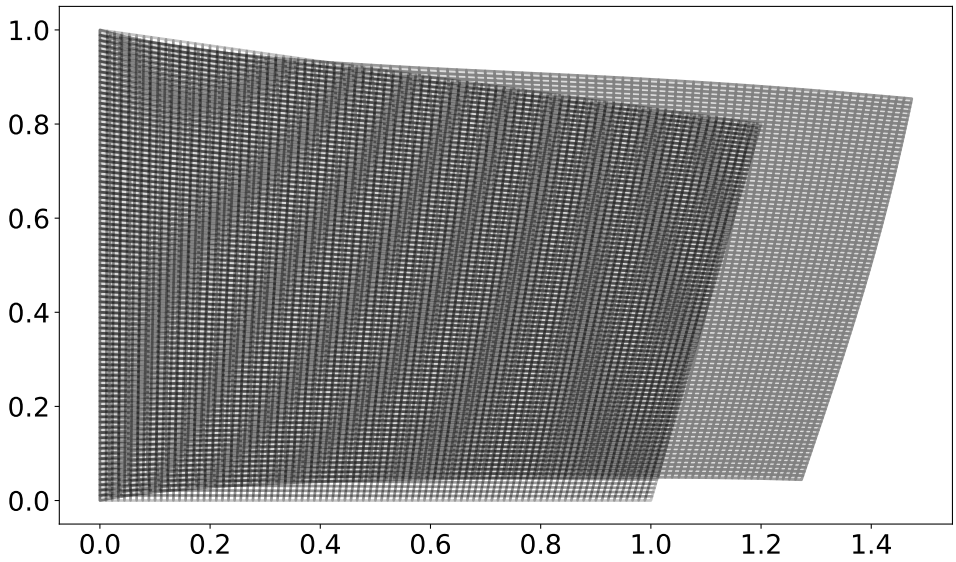


Figure 3.35. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The displacement in the high-fidelity (HF) solution $u_h(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$, and $\mu_1 = 0.2$ and $\mu_2 = -0.2$. The displaced position is shown in gray, whereas the initial position is shown in black, i.e. the displaced position is the position being displaced to the right.

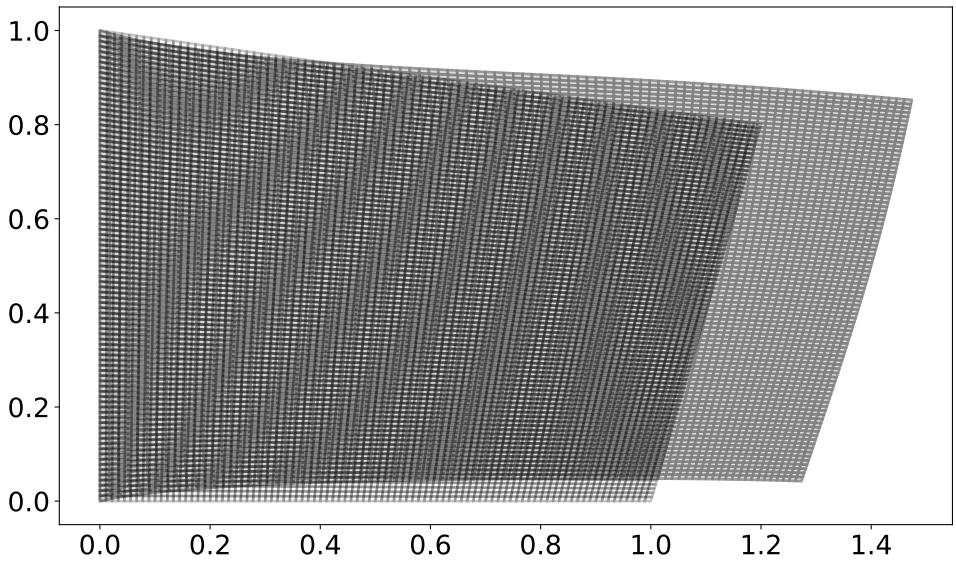


Figure 3.36. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The displacement in the recovered reduced-order (RB) solution $Vu_N(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$, $\varepsilon_{\text{POD}} = 10^{-2}$, $N = 8$ RB degrees of freedom, and $\mu_1 = 0.2$ and $\mu_2 = -0.2$. The displaced position is shown in gray, whereas the initial position is shown in black, i.e. the displaced position is the position being displaced to the right.

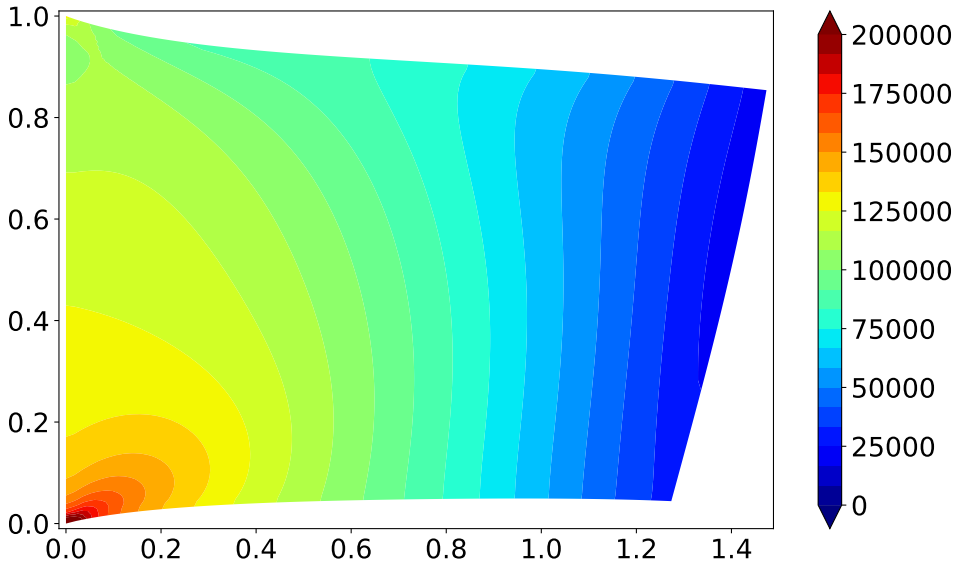


Figure 3.37. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The recovered von Mises stress in the high-fidelity (HF) solution $u_h(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$, and $\mu_1 = 0.2$ and $\mu_2 = -0.2$. Note that the von Mises stress approaches infinity at $(0, 0)$, therefore the plots are limited to a maximum stress of 200 000.

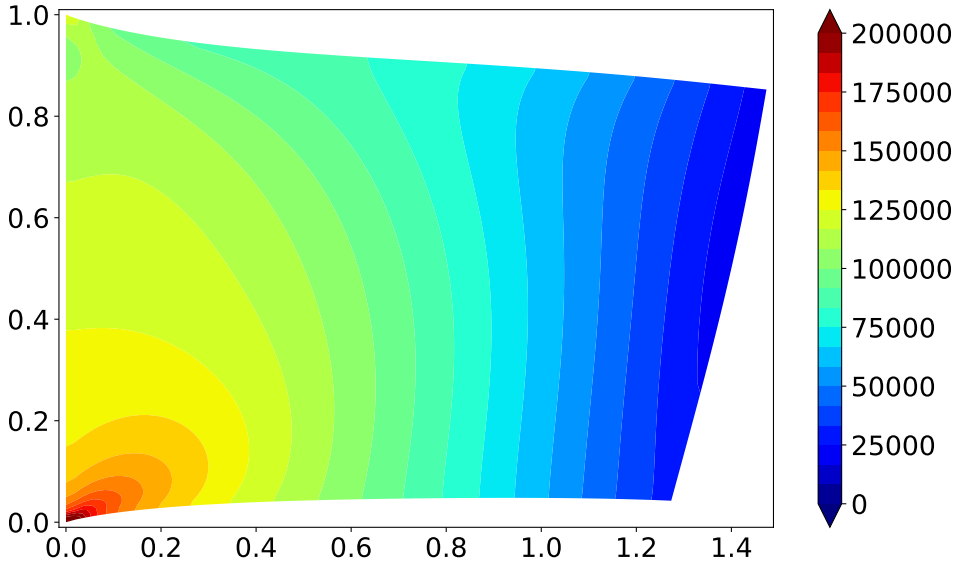


Figure 3.38. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; The recovered von Mises stress in the recovered reduced-order (RB) solution $Vu_N(\boldsymbol{\mu})$ for the solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$, $\varepsilon_{\text{POD}} = 10^{-2}$, $N = 8$ RB degrees of freedom, and $\mu_1 = 0.2$ and $\mu_2 = -0.2$. Note that the von Mises stress approaches infinity at $(0, 0)$, therefore the plots are limited to a maximum stress of 200 000.

High-fidelity model	
Number of HF dofs N_h	12 960
Number of Snapshots	625
MLS components Q	210
MLS fit time	$\approx$ 9 min
Time for saving one snapshot*	$\approx$ 2 min 30 s
HF assembly time	$\approx$ 1 min 52 s
HF solution time	$\approx$ 0.9 s
Reduced-order model	
Number of RB dofs N	8
Dofs reduction	4 095 : 2
Build RB model by POD time	$\approx$ 15 min
Offline CPU time**	$\approx$ 24 hours
Online CPU (RB solution) time	$\approx$ 1.3 ms

Table 3.6. More than Ten Thousand Degrees of Freedom; Case 2 — Dragging One Corner of a Rectangle; Computational details for the high-fidelity (HF) and reduced-order (RB) model built form solving of the problem of Constant Body force in 2D using $n = 90$ elements along the axes, the geometry parameter range $\bar{G}_{\text{DR}} = (-0.3, 0.3)$, order $p = 19$, $\mu_1 = 0.2$ and $\mu_2 = -0.2$, and $\varepsilon_{\text{POD}} = 10^{-2}$ for the proper orthogonal decomposition (POD) with respect to the energy norm, algorithm 2. *The time for saving one snapshot is based on saving the snapshot for $\mu_1 = 0.2$ and $\mu_2 = -0.2$ multiple times, different μ_1, μ_2 may result in a different time. **The time to save one snapshot is multiplied by $25^2 = 625$ to get the total time for saving snapshots, in addition comes the time of one HF assembly for the special mean snapshot. Note that the use off multiprocessing to save snapshots may speedup the saving of snapshots.

Remark 3.17. *The von Mises stress approaches infinity in the corner of* $(0, 0)$ *of the clamped boundary of* $x = 0$*, therefore the plots are limited to a maximum stress of* 20 000*.*

3.5.6.3 Computational Performance

We now test the computational performance of our solver for this case. As stated in section 3.5.6 we use $n = 90$ elements along the axes to get $N_h = 16\,380$ degrees of freedom (dofs) which is more than $n_s = 15\,625 = 25 \times 25 \times 5 \times 5$, which is the number of solutions in the snapshot matrix (2.74). This means that we have an approximate reduction factor of 2 000 in the degrees of freedom and that we use a 25×25 uniform grid for the geometry parameters μ_1 and μ_2, and a 5×5 uniform grid for the material parameters E and ν. Studying the computational details and times show in table 3.6 we see that we get a speedup of order 600 for the online stage. This is quite good taking into account that the Matrix Least Square algorithm gives Q above 200, meaning that since we here have homogeneous Dirichlet boundary conditions will need to sum above 200 matrices for $A_N^{(1)}(\boldsymbol{\mu})$ and $A_N^{(2)}(\boldsymbol{\mu})$, and above 200 vectors for $\boldsymbol{f}_N^{(0)}(\boldsymbol{\mu})$ in (2.194).

Remark 3.18. *The times in table 3.6 show the mean of* 1 000 *runs of the high-fidelity (HF) and reduced-order (RB) solutions,* 50 *runs of the HF assembly and saving of one snapshot, and* 10 *runs for the Matrix Least Square fit and the building of the RB model by the POD algorithm. The results where obtained on a private computer using a AMD Ryzen 9 5950x 16-Core Processor @ 3.4/4.9 GHz CPU and 32 GB RAM on an ASUS ROG Strix X570-F Gaming Motherboard with an AMD RX 6800XT Sapphire Nitro+ GPU. The computer was set to run with only Python (Anaconda3 + Pycharm) running in the foreground.*

3.5.6.4 Discussion

During the study of this example we concluded that decrease in the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ slows down because of the higher number of degrees of freedom than in the previous examples. Furthermore, in section 3.5.6.2 we observed that the reduced-order solution for $N = 8$ POD modes approximates the high-fidelity solution quite good since the displacement and recovered von Mises stress are similar. Using $N = 8$ POD mode lead to an approximate reduction factor of 2 000 in the degrees of freedom, which ultimately resulted in a speedup of order 600 for the online stage. This is quite good taking into account that the Matrix Least Square algorithm gives Q above 200, meaning that since we here have homogeneous Dirichlet boundary conditions will need to sum above 200 matrices for $A_N^{(1)}(\boldsymbol{\mu})$ and $A_N^{(2)}(\boldsymbol{\mu})$, and above 200 vectors for $\boldsymbol{f}_N^{(0)}(\boldsymbol{\mu})$ in (2.194).

3.6 Summarizing the Results

In section 3.2 we tested and observed that our implemented solvers work. In section 3.3 we sat up algorithm 4 to determine the Matrix Least Square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ and in section 3.4 we presented the problem of a Constant Body force in 2D. We used both in the studies in section 3.5.

In section 3.5.1 we tested the Matrix Least Squares algorithm 3 and concluded that it does split the affine matrices $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ in the desired way for an affine problem.

In sections 3.5.2 and 3.5.4 respectively, we observed what happens when being too close to the singularities of the maximum valid range for the Taylor expansion, and how this effects the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$. First in section 3.5.2, we hypothesised that the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ are dominated by the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$. We confirmed this in section 3.5.4 by using a smaller geometry parameter range well within the maximum geometry parameter range.

Next, we observed in section 3.5.5 that we had too few snapshots leading to the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ being dominated by the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$. This was caused by having 3 snapshots in each of our 6 directions, meaning that order $p = 2$ is the best order we can achieve, noting remark 2.13.

Finally, in section 3.5.6 we observed the effect of having more than 10 000 degrees of freedom, which lead to $N = 8$ POD modes for the reduced-order solution leading to an approximate reduction factor of 2 000 in the degrees of freedom. This, ultimately resulted in a computational speedup of order 600 for the online stage. That is quite good taking into account that the Matrix Least Square algorithm gives Q above 200, meaning that since we here have homogeneous Dirichlet boundary conditions, we will need to sum more than 200 matrices for $A_N^{(1)}(\boldsymbol{\mu})$ and $A_N^{(2)}(\boldsymbol{\mu})$, and more than 200 vectors for $\boldsymbol{f}_N^{(0)}(\boldsymbol{\mu})$ in (2.194).

The results regarding the geometry parameter range are as expected, since we know that the Taylor expansion is more accurate when well within the valid range for the parameters. However, the results regarding few snapshots in each direction are both expected and unexpected. Expected because we know we need $n + 1$ snapshots in each direction for approximations of order n. Unexpected because we would have hoped for better results for low orders. Studying the other cases, we observe that the results are similar to the low orders in these other cases , making the result expected. Finally, the results when studying the effect of over 10 000 degrees of freedom are expected in the form that we expected there to be little difference between the high-fidelity

solution and the reduced-order solution when capturing 99.99% of the energy in the system. Furthermore, the computational speedup of order 600 for the online stage is as expected, since the reduced-order solution needs to add over $Q = 200$ matrices and vectors three times.

As we have seen, tweaking the input parameters on how many snapshots we have in each direction greatly effects the results. This would also, to a lesser degree, be the case for our affine parameter E and ν, which we here only have used a 5×5 uniform grid for. Moreover, changing ε_{POD} form 10^{-2} to 10^{-3} to now capture 99.9999% of the energy in the system would only make the reduced-order solution closer to the high-fidelity solution. Changing the way the Matrix Least Square functions $\{g_q(\boldsymbol{\mu})\}_{q=0}^{Q}$ are determined would change the results. But a bad choice of functions will lead to that the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ do not decrease or decrease quit slowly. We experienced such a result when we ran a bugged analysis, not shown in this thesis, where some of the functions depending on μ_1 where missing.

Chapter 4

Conclusion and Future Work

In this chapter we first present the summary of the main results, then conclusions of our study on the *Matrix Least Square* technique. Finally, we state some recommend potential further work for continued studies.

4.1 A Summary of the Main Results

In section 3.5.1 we tested the *Matrix Least Squares* algorithm 3 and concluded that it does split the affine matrices $A_1(\boldsymbol{\mu})$ and $A_2(\boldsymbol{\mu})$ in the desired way for an affine problem. Studying the results section 3.5.2 we hypothesised that the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the recovered reduced-order solution $Vu_N(\boldsymbol{\mu})$ where dominated by the relative errors between the high-fidelity solution $u_h(\boldsymbol{\mu})$ and the high-fidelity Matrix Least Square solution $u_{h,\text{mls}}(\boldsymbol{\mu})$ because we were too close to the singularities of the maximum valid range for the Taylor expansion. This we confirmed in section 3.5.4 by using a smaller geometry parameter range well within the maximum geometry parameter range. Next, in section 3.1.3 we observed that this domination also occurs when we had to few snapshots because we need $n + 1$ snapshots in each of our directions for approximations of order n, noting remark 2.13. Finally, in section 3.5.6 we observed the effect of having more than 10 000 degrees of freedom, which lead to $N = 8$ POD modes for the reduced-order solution leading to an approximate reduction factor of 2 000 in the degrees of freedom. This, ultimately resulted in a computational speedup of order 600 for the online stage.

4.2 Conclusions

The main conclusion from these results is that the developed *Matrix Least Square* technique preforms well when we have enough snapshots and are well within the maximum valid geometry parameter range for the Taylor expansion. The technique also splits the matrices of an affine problem as desired. Furthermore, when using it on a problem with over 10 000 degrees of freedom we observed a computational speedup of order 600. Using this observation we conclude that the *Matrix Least Square* technique has a great potential for reduced basis modelling, even if the number of snapshots needed may increase greatly with the increase of number of non-affine parameters as observed in section 3.1.3. We conclude this even if we did restrict our problems heavily and thereby made them simple.

The overall objective of the thesis was the construction, presentation and testing of the *Matrix Least Square* technique for non-affine problems. We consider this objective achieved since we first through chapter 2 thoroughly studied the basic theory behind the reduced basis methods using finite elements, presented the *Matrix Least Square* technique, and applied the theory to our

restricted cases of the *Linear Elasticity Equations*. Then secondly, though chapter 3, we built a solver and tested it, and finally we used it to test the *Matrix Least Square* technique and studied a simple numerical example using the technique for different non-affine geometries.

4.3 Recommendations of Future Work

As mentioned in the conclusions in section 4.2 our study was done on a simple example for different non-affine geometries. However, all these cases are of great interest and have their applications in areas such as design, optimization and real-time control [9]. Therefore, we suggest to investigate the use of sparse snapshot generation, especially for problems containing a large number of design parameters. The impact of this on the performance is expected to be that we do not need as many snapshots as now, while still maintaining the order of approximation for the *Matrix Least Square* technique.

We also suggest to investigate the extension to the 3D case. This would give more geometry design parameters and it would be interesting to observe the speedup gain. Furthermore, we would like to see if results of this thesis extend to more complex geometry deformations and other partial differential equations.

Looking at physical applications of reduced-order modelling, especially modeling non-affine geometries, could be useful in for instance modeling a bridge that is about to collapse, like the one dividing northern Norway in half [21]. This could be a complex geometry problem in 3D with many design parameters.

Bibliography

[1] A. Quarteroni, A. Manzoni and F. Negri. *Reduced Basis Methods for Partial Differential Equations An Introduction*. Springer, 2016.

[2] A. Rasheed, O. San and T. Kvamsdal. 'Digital Twin: Values, Challenges and Enablers From a Modeling Perspective'. In: *IEEE Access* 8 2020, pp. 21980–22012. DOI: 10.1109/ACCESS.2020.2970143.

[3] E. Fonn et al. 'Fast divergence-conforming reduced basis methods for steady Navier–Stokes flow'. In: *Computer Methods in Applied Mechanics and Engineering* 346 2019, pp. 486–512. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2018.11.038.

[4] O.M.S. Gran. *OlavMSG/Code-Master-Thesis-Spring-2022: v1.0.1*. 6th June 2022. DOI: 10.5281/zenodo.6617578.

[5] A. Quarteroni. *Numerical Models for Differential Problems*. 2nd ed. Springer, 2009.

[6] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen*. 2nd ed. Vieweg & Teubner, 2010. ISBN: 978-3-8348-9357-4.

[7] A. Quarteroni, R. Sacco and F. Saleri. *Numerical Mathematics*. 2nd ed. Springer, 2007.

[8] R. Adams and J. Fournier. *Sobolev Spaces*. 2nd ed. Elsevier Science, 2003. ISBN: 978-0-12-044143-3.

[9] J.S. Hesthaven, G. Rozza and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. Springer, 2016.

[10] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. 3rd ed. Springer, 2008.

[11] J. Gareth et al. *An Introduction to Statistical Learning*. 2nd ed. Springer, 2013.

[12] W.K. Härdle and L. Simar. *Applied Multivariate Statistical Analysis*. 4th ed. Springer, 2015.

[13] G.W. Stewart. 'On the early history of the singular value decomposition'. In: Siam review 35(4), 1993, pp. 551–566.

[14] G.H. Golub and C.F. Van Loan. *Matrix Computations*. 4th ed. Baltimore: The John Hopkins University Press, 2013.

[15] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. Philadelphia: Society for Industrial and Applied Mathematic, 1997.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Siam, 2003. ISBN: 978-0-89871-534-7.

[17] T.J.R. Hughes. *The Finite Element Method. Linear Static and Dynamic Finite Element Analysis*. Dover Publications Inc., 2012.

[18] C.A. Felippa, B. Haugen and C. Militello. 'From the individual element test to finite element templates: Evolution of the patch test'. In: *International Journal for Numerical Methods in Engineering* 38.2 1995, pp. 199–229. DOI: https://doi.org/10.1002/nme.1620380204.

[19] *Programming project in TMA4220 - part 2*. 2020. URL: https://wiki.math.ntnu.no/tma4220/2020h/project (visited on 10/12/2021).

[20] *Solving PDEs in Python -
 The FEniCS Tutorial Volume I.* URL: `https://fenicsproject.org/pub/tutorial/html/._ftut1008.html` (visited on 07/12/2021).

[21] *Bro svikter på E6 i Troms: – Norge er delt i to.* URL: `https://www.vg.no/i/34a52v` (visited on 04/06/2022).

Olav Milian Schmitt Gran

NTNU
Norwegian University of
Science and Technology