Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

REDUCED ORDER MODELS FOR 3D PARAMETRIC LATTICE STRUCTURES: COMPUTATION AND POST-PROCESSING IN PORTABLE DEVICES

TREBALL FINAL DE MÀSTER

Treball realitzat per:
Gabriel Valdés Alonzo

Dirigit per:
Pedro Díez i Alberto García

Màster en:
Mètodes Numèrics en Enginyeria

Barcelona, 28 de Setembre de 2018

Departament d'Enginyeria Civil i Ambiental

COPYRIGHT NOTICE FOR THIS DOCUMENT

September, 2018

Barcelona, Spain
grvaldes@uc.cl

Acknowledgements

I want to start acknowledging the people who made this project possible. Thanks to professors Pedro Díez and Alberto García, for choosing me to work with them in this interesting project. Special thanks also to Alberto Sibileau for the countless help given during the semester and the commitment to this work.

Formally, I would like to acknowledge the contribution made by the University of Zaragoza. The PGD model of a cantilever beam served as a basis for the application developed in this work, and such long way wouldn't be achieved without the inspiration.

Lastly, I want to thank all the people who made this possible, both in a direct and indirect way. To my family, for both the financial and mental support through this two years. To all the friends that were there, the ones who came during these two years or just classmates that helped with ideas or just words of motivation. To my girlfriend, for both being there and helping even with the little stuff in here. Finally, to anyone that may have influenced the work in any way, or who may be influenced by this work.

Como siempre, dedicado a la familia y amigos que están y no,
dedicado a la polola
y dedicado incondicionalmente a Sasha Grey.

Contents

List of Figures

Abstract

Materials Science is a fast evolving field in engineering. *Architectured Materials* is a relatively new field in which materials properties are designed based on the geometry at the micro-scale of the constitutive elements. In this project, we are interested in auxetic materials, which main property is having negative Poisson's coefficients. In this regard, we focus on structures that display such behavior, in particular lattices formed by an "inverted honeycomb" unit cell. The main objective is to solve the structural problem associated with 3D lattices in a parametric fashion. The structure is modelled using linear 3D beam elements and analyzed through Proper Generalized Decomposition (PGD) method, which provides a *vademecum* of the available solutions for a set of parameters chosen in discretized intervals. Furthermore, taking advantage of the tabulation of the results, a post-processing tool was designed as a web application, in order to study the variation of the solutions given certain parameters in real-time. The analysis was performed on a unit cell, which was homogenized in order to obtain the effective elastic tensor, allowing us to compute the effective Poisson's coefficients, as an approximation for an "infinite lattice". Also, a full pattern of $3 \times 3 \times 3$ unit cells was studied and approximate values of the Poisson's ratios were obtained. The results show that the homogenized solution represents a valid approximation for "infinite lattices", while the full pattern allows us to obtain a detailed solution, accounting for the boundary effects in the structure, but with the associated computational cost. The PGD method proves as an effective tool for this type of parametric computations, which accuracy is directly linked with the amount of modes computed. Finally, the post-processing tool developed proves to be an excellent support for the understanding of the solutions, allowing both, experimented as well as unacquainted users, to visualize and design specific auxetic materials.

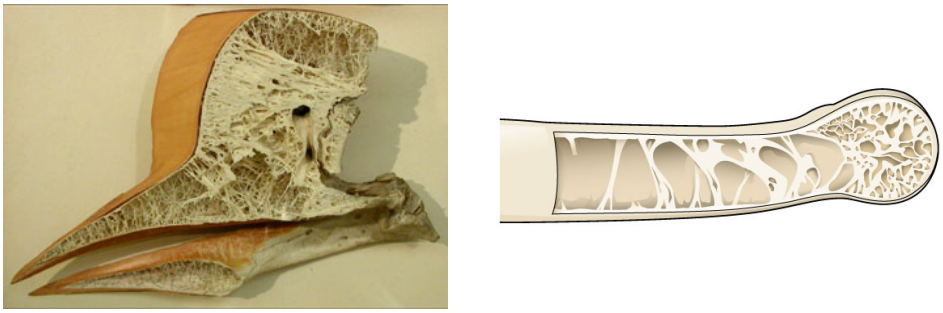
Chapter 1

Introduction

Materials Science is an always evolving field where engineering is used in order to develop new solutions to a big range of problems that arise from new technologies. Thanks to new manufacturing processes being created and massively used, new materials become available and optimized. This allows to implement new specific properties that classic materials do not possess. In this line, *Metamaterials* is a relatively new field in which inspiration is taken mainly from nature, which makes it possible to develop elements with properties that would have been unthinkable years ago.

1.1 Architectured Materials

Architectured Materials are a special kind of materials in engineering that belong to the category of metamaterials, which definition is not unique, but can be agreed on "materials with customized properties at the macro-scale that come from an engineered form, rather than the inherent properties of the material at the micro-scale" [1]. As mentioned before, this chosen form takes much inspiration from nature. For example, most of the materials with solid structure, if not all of them, have densities much higher than the density of water, which may not be suitable for life in animals like birds, due to the weight it carries. Nature has solved this problem in an intuitive way: adding voids to the solid, in order to reduce the general density of the body while retaining the stiffness and strength [2]. In these cases, the material from which the structure is formed (such as bones or cartilages among others, as can be seen in figure 1.1) still retains the original properties. However, at the macroscopic scale the properties differ, optimizing the design to reduce the density. In nature this design has evolved for centuries and it is done mostly in a seemingly random fashion. This randomness has been replicated successfully through materials like foams [3], which have allowed new technologies benefiting from the low density. Nonetheless, it has the drawback of not being able to imprint certain properties that may be wanted to fix, like stress or stiffness orientations, since randomness calls for isotropy.



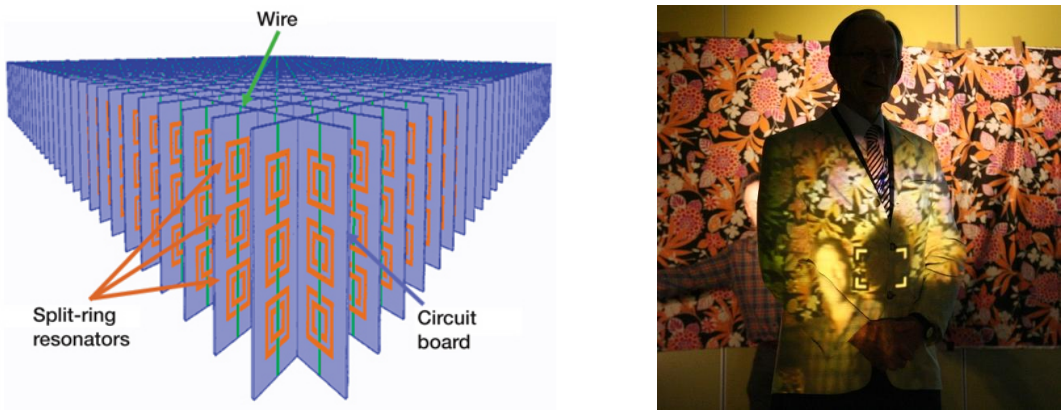
(a) **(b)**

Figure 1.1: Examples of uses of voids in nature. In **a)**, the inside of a Hornbill beak is shown (Source: Tetrapod Zoology). In **b)**, an example of a bird bone is displayed (Source: CNX OpenStax).

In order to take advantage of this design, and to optimize it for desired properties, the idea of adding voids is retained, but the randomness of the distribution of the material is replaced with an structured repetition of what is called a "unit cell" [4]. A unit cell is the basic structure that is repeated and tessellated in space into a lattice, which in turn represents the material used in the macro-scale. Easy designs like honeycomb patterns have been used for long time, but advances in computer assisted design as well as 3D printing technologies allow to design more intricate shapes that can be easily reproduced. Since now the shape can be controlled, also the desired properties can be implemented with no major difficulties. In this way, in fields like Thermodynamics, low-weight insulation can be developed; in the field of optics, materials with negative refractive indexes have been achieved, among others [5] (see figure 1.2). One of the biggest advantages of the lattice-like design is the use of homogenization, which allows to approximate the properties of the macro-scale material based on the properties and parameters chosen for the design at the micro-scale of the material (namely, the properties of the unit cell), which simplifies and speeds the analysis of the pattern [6].

1.1.1 Auxetic Materials

In the scope of the present work, the focus is put on a specific type of architectured material, the auxetic metamaterial, which is in structural engineering one of the most important developments in the field. The properties of an auxetic structure are particularly dependent on the spatial shape, organization and geometry of the unit cell. In this case the shape is chosen to have a negative Poisson's coefficient at the macro-scale. That means, when a uniaxial tensile load is applied to a material, it also expands perpendicularly (as exemplified in figure 1.3). The opposite phenomenon is also observed: when the material is uniaxially compressed in one of its axes, the others also display contraction, effectively increasing the density of the material.



(a) **(b)**

Figure 1.2: Examples of metamaterials in different fields. In **a)**, a lattice used as an acoustic insulator is shown (Source: Wiki Commons). Copper rings and wires are mounted in fiberglass, creating local resonators that attenuate effects of waves, given the properties selected for the resonators. In **b)**, an example of a fabric with negative refractive index is displayed (Source: LENS). Unit cells are defined in sizes smaller than the wavelengths of the visible spectrum, creating an effective medium with properties that differ from the constituents.

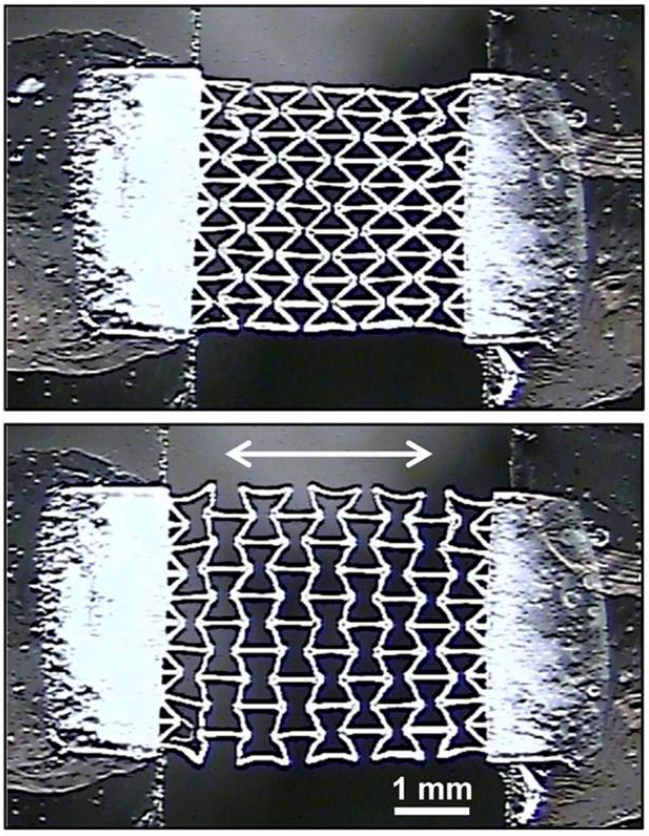


Figure 1.3: Example of auxetic behavior (Source: Smart Structures).

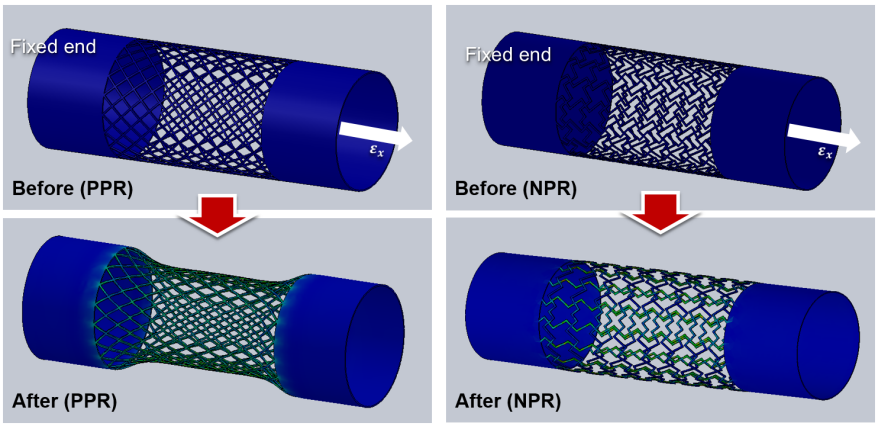


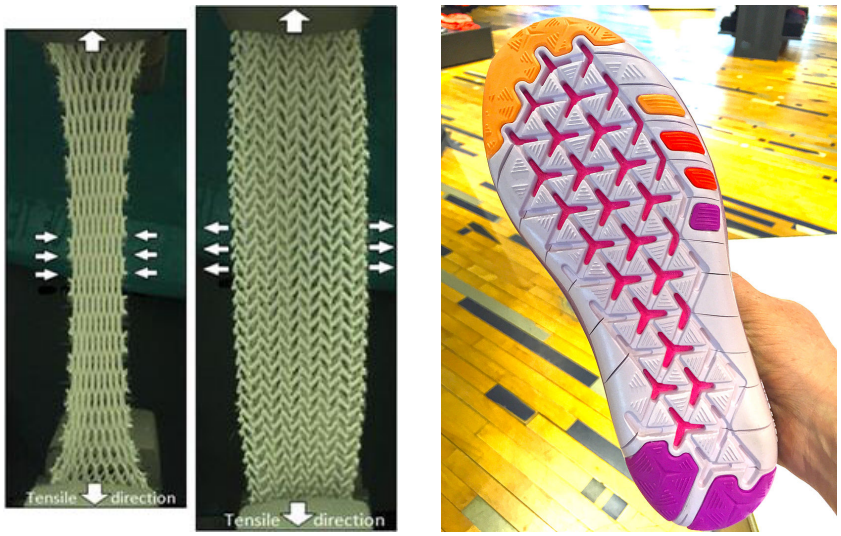
Figure 1.4: Examples of auxetic design of a blood vessel stent (Source: [10]).

Uses for auxetic materials span different fields. In materials science they prove interesting due to the possibility of enhanced mechanical properties [7]. Due to their nature given by the negative Poisson's ratio, there is a possibility of increasing indentation resistance and fracture toughness, given the increase of density when the element is compressed. This also means that the energy absorption shown by these materials is bigger than others currently used, which gives possibilities for design of protective gear and shock reducing parts [8]. It has been shown also [5] that negative Poisson's values lead to increased values of the shear modulus of the macro-structure. In the field of biomechanics, the use of auxetic materials is interesting for the design of artificial tissue, mainly blood vessels, as is exemplified in figure 1.4.

Also, other uses been proposed in the industry for common materials such as fabrics or insulation (as exemplified in figure 1.5) or the field of biomechanics [9] (referenced in figure 1.6).

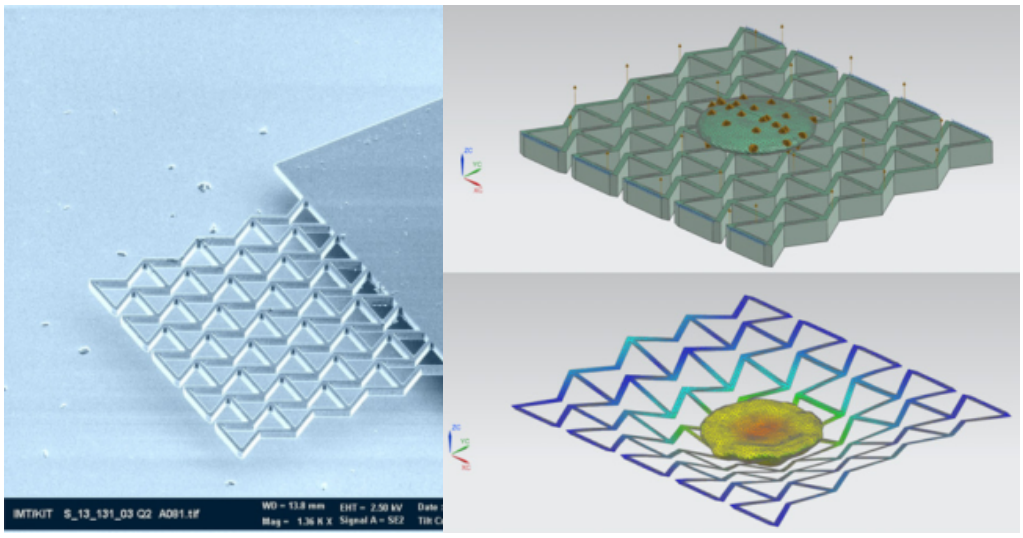
1.2 Reduced Order Models

Architectural Materials are formed in lattices that can extend in every direction. Computational modeling works as a tool to understand how these structures behave, given their different properties. These can be modified by adjusting parameters that can model changes, either in their shape or the mechanical properties of their constituent materials. Given the sizes of the lattices (in terms of elements), computing such parametric problems quickly becomes inefficient for a fair amount of parameters due to the number of equations required to solve the finite element problem, deriving in the so-called curse of dimensionality [11]. In order to address this problem, Reduced Order Models (ROM) are used to overcome this. The main goal of this methods aims to reduce the amount of data needed to solve a problem, in order to obtain a fast and efficient solution and reducing the resources needed to solve the model, which also provides the means



(a) **(b)**

Figure 1.5: Examples of auxetics in fabrics. **a)** Example of an auxetic fabric being deformed (Source: Advanced Fabric Technology). **b)** Example of a pattern applied to a shoe to provide increased flexibility (Source: Wiki Commons).



(a) **(b)**

Figure 1.6: Examples of auxetics in biomechanics (Source: [9]). In **a)**, a nanometric pattern is developed. Auxetic structures present enhanced resonance properties useful for cell culture processes. In **b)**, the vibration modes of the scaffold are analyzed with a single cell attached to it.

to easily interpret these results. These methods seek to reduce the order through the detection of redundancies, decreasing the degrees of freedom that are less relevant to the solution and giving low dimensional approximations.

There are several ROMs available in the literature [12]. There are also methods like the Singular Value Decomposition (SVD) or Proper Orthogonal Decomposition (POD) [13], among others. In the scope of this work, the Proper Generalized Decomposition (PGD) method [14] is the one used for approximating the solutions of the lattice. This method is particularly useful since it provides a fast way of diplaying solutions through the tabulation of explicit parametric results in accordance to the number of evaluation points set for each of the parameters intervals. This particular data structure provides efficient storage and easily accessible data that can be displayed in real-time [15].

Considering the stated above, PGD provides results with a particular decomposition in spatial modes and parametric functions. Each of the modes by itself can not be used directly to interpretate the complete parametric behaviour. Instead, in order to reconstruct the solution for any given value of the parameters, each of the spatial modes must be added, correspondingly weighted by their parametric functions. This leads to a tedious post-processing that may not be necessarily understandable, even more considering that the person doing it may not have knowledge about the topic. The need for a simple post-processing tool, that allows to understand the data obtained and shows the whole gamut of results available from the computation, is one of the main motivations that drives the work in this document.

1.3 Web Applications

The internet has seen an exponential growth since its inception. Rapid growth in communication technologies has led to bigger capacities such as hosting space and speed of access. With the creation of programming languages dedicated to web interactions, the development of the so-called web-based applications, which are pieces of software hosted at the server side that can be accessed on the client side, has increased in the last years.

Several advantages can be listed to support the development of Web-based Application (Web Apps from now on). The most relevant is the accessibility and cross-platform interoperability. The applications are typically developed in a dedicated framework, which means that the application is coded only once on the server side. Common web browsers are designed to interpret these frameworks, which then allows to access the same software virtually everywhere and on any platform that has access to the internet. This has a double benefit from the developer side: universality of the application and faster development. The application as such is developed properly in only one group, erasing the need

of programming for different platforms, reducing costs and deploying time, since only secondary parts, such as the graphical interface, should be adapted for different purposes. This also helps to extend the popularity and usability of the application: since no special platform is required, the users have the freedom of choosing their preferred access method, and the portability that this offers when operated in mobile smart devices. Another relevant advantage from the development taking part on the server side is the handling of updates, since the code can be modified in real time and committed without interfering on the user side. Once updates are ready, they are loaded into the server, effectively updating the software for every user accessing the site, and not requiring them to make this process locally, as it would have to be done with typical software. This offers great benefits in security and bug fixing, which can be complemented with feedback from the users for quick detection and fixing. Finally, it is worth mentioning as an advantage the easiness of use. Since the application is hosted on the cloud, the access is made easy and no installations are required, allowing for usage in devices with small storage space, as well as simultaneous access from different platforms, which can be useful for multi-user work or for resuming previous work in a different device.

Although the previously mentioned advantages are a compelling argument for developing the post-processing application, there are some drawbacks to Web Apps that need to be taken into account and are worth mentioning. The most relevant downside regarding to the work presented here is concerning "performance". Web Apps rely on scripting in order to communicate from the user side to the server, and most of the languages currently in use are optimized for this kind of transactions. In the particular case of this work, the framework used for graphics is optimized to work in a similar way as a native application. While this is allowing good performance in desktop devices, however, it shows poorer performances in mobile devices, which handle graphics in slightly different ways. Another disadvantage to be aware of is the availability and stability of internet connection. Even though measures can be taken to make offline working possible, it defies the purpose of a server hosted application. Also, this poses a dedicated and reliable hosting provider as a requirement since it is essential for the site to work that the server is operable at any moment. This is a problem that cannot be handled by the developing team, effectively externalizing a risk. An additional problem that is not dependent on the development is the variety of available web browsers. Even though efforts have been made to unify the way browsers handle the code, some differences can still be expected. These have to be taken into consideration on the design in order to avoid unexpected errors. Finally, since the application works on the cloud, security and data protection are always present issues that need to be considered in order to guard any critical information.

1.4 Project outline

With all the aforementioned regarding Architectured Materials, Reduced Order Modelling, parametric design and web applications; in the present project, the development of a web-based PGD parametric tool for real-time simulation and design of architectured materials with auxetic behavior is proposed.

With this objective in mind, this document is organized as follows: in chapter *Problem statement*, a unit cell with an inverted honeycomb pattern is presented. The design is based on the previous work of Sibileau et. al. (2018) [16], where a 2D unit cell was studied. The cell is analyzed under the Euler-Bernoulli beam theory in two cases: one unit cell with homogenization, and a lattice of cells subjected to uniaxial strain. The first case is implemented in order to obtain the effective properties of a cell as if it were acting inside of a lattice. In other words, the values of the constitutive effective tensor are computed, from which the effective values of Poisson's coefficients can be obtained. The second case is computed in a full lattice that has a prescribed amount of unit cells tesselated in each direction of periodicity. The simulations on a full structure allows to capture behaviour that is not present in the homogenized model, such as boundary effects or non-periodic loads. In addition, we use the solutions of our full model to compare some of its macro-scale properties against the effective ones computed through homogenization, which represents the ideal case of tesselation of an infinite amount of cells in the directions of periodicity.

Next, chapter *Proper Generalized Decomposition* will present to the reader the algorithm behind the PGD method, which provides a complete set of results for different ranges of parameters, output in a tabulated fashion that allows real-time handling. Following, chapter *Post-processing through web application* introduces the reader to the use of a post-process tool developed as a web application, taking advantage of the real-time properties offered by PGD. The application is developed as graphical interface for showing the possible solutions to a problem while its defined parameters can be manipulated. The capabilities of the WebGL framework in combination with the versatility of JavaScript makes web development an appropriate tool which renders a fast solution available transversely through any platform that supports the use of JavaScript and WebGL engine (i.e. any modern web browser such as Chrome, Edge or Firefox).

In the following chapter, solutions to the different cases defined in the problem statement are presented and described. Finally, in the chapter *Conclusions*, the highlights and advantages of the proposed tool are shown together with its limitations and the possibilities that exist for future works.

Chapter 2

Problem statement

The interest in this project is studying the mechanical properties at the macro-scale of architectured materials, formed by repeating a structure at the micro-scale, called the *unit cell*. Additionally, we focus on materials that can exhibit auxetic behavior, in particular, a type of lattice structure that is known as *inverted* or *re-entrant honeycomb*. The mechanical model chosen for this structure is the standard Euler-Bernoulli linear beam theory. This model, typically solved by Finite Element Method will be addressed in a parametric form and solved taking advantage of the PGD method, in order to reduce computational times. Although it is out of the scope of this thesis, the parametric solutions of these structures can be used in optimization or inverse problems, improving the performance of algorithms that design materials with taylored properties. In combination with 3D printing technologies, this can certainly boost the efficiency of the engineering design and manufacturing processes.

2.1 3D structural problem

Auxetic materials generally profit from the properties of the connected elements, which allows the full structure at the macro-scale problem to behave as it had a negative Poisson's coefficient [1]. This suggests the use of elastic elements for the modeling of the constituents of the material at the micro-scale problem, in order to capture the mentioned properties. In the case of this work, since we are dealing with a lattice material, a unit cell is defined as a set of beam elements connected in a specific pattern. In addition, this mechanical model allows us to introduce a parameterization of the unit cell structure by means of the beams geometrical properties (i.e: lengths, thickness and orientations).

2.1.1 The 3D beam element

Euler-Bernoulli beam theory proposes a continuous element with small deformations and only lateral loads [17]. The basic diagram is presented in figure 2.1

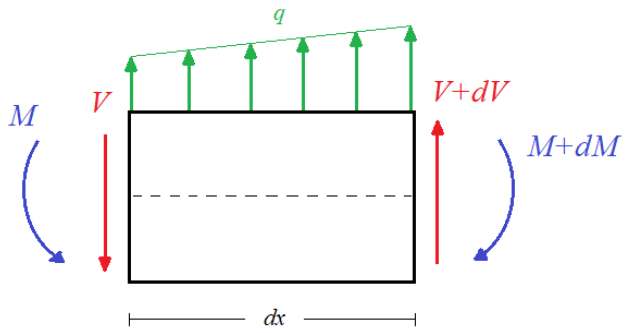


Figure 2.1: Diagram of Euler-Bernoulli beam theory

The lateral loads generate moments and shears in both transverse axes of the beam, while the extension or contraction of the element causes an axial strain. These forces and moments generate a displacement field inside the beam, which can be accurately represented by volume elements in the FEM method. As an approximation, the whole beam can be represented as a one dimensional element that captures the essence of the displacement field on the beam, following the basic Navier hypothesis that cross sections remain undeformed, which creates a curvature in the element and allows to reduce a face into a point in the line. Given that there are displacements and curvatures, a six d.o.f. element is proposed in 3D, defined as

$$\boldsymbol{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \\ \theta_x \\ \theta_y \\ \theta_z \end{pmatrix},$$

where the (x, y, z) subindices represent the local axes in the beam as defined in figure 2.2, being x the one that starts at node 1 of the element and points to node 2.

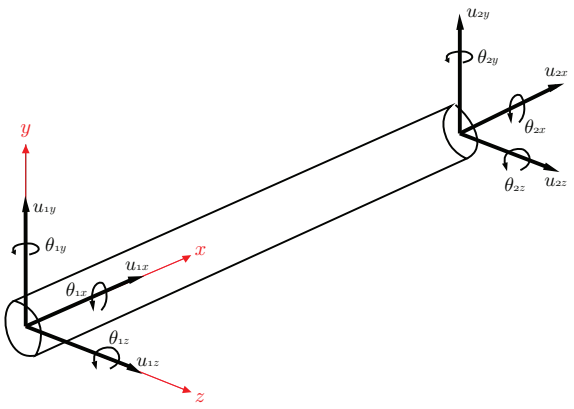


Figure 2.2: Diagram of the 3D beam element.

2.2 Structural problem parametrization

Solutions to structural problems are often of numerical nature, meaning that values are considered for the different parameters related to the problem and a unique solution is obtained for this particular set of values. This is practical, since in engineering the goal of is obtaining a particular solution for a determined problem. Nevertheless, for a study case this becomes cumbersome, since many times the interest is put in the study of the variation of parameters and the effects they have in the problem. As a way to overcome this issue, parametrization is one of the most common solutions.

A parametrized problem is a normal engineering study case in which the previously defined properties are now left as variables, leaving the solution to be an expression in terms of them. One example of this are analytical expressions, in which the result is a mathematical function which variables can be evaluated with different values, giving a valid result for that set of parameters, as it would yield a normal problem with the fixed values. This proves advantageous since now all the possible solutions to the problem are written as mathematical expressions that can be evaluated when needed, without consuming time and resources solving many cases. However, drawbacks are evident in the case of analytical expressions. The time and effort required for obtaining these kind of solution is big, since for average sized problems the amount of variables and equations to solve beforehand can be considerable. Symbolic computation can be used as an aid for this kind of calculations, but the amount of processor power required can be unaffordable in big problems.

Nonetheless, parametrizing a problem can bring several advantages too. First, it allows to "organize" the problem and reduce computation times. A set of parameters can be chosen and then decide if some parameters can be fixed or if they are irrelevant to the study, which could reduce the size of the problem. Secondly, it might not be possible to find an analytical solution for the problem, but parametrization still can be performed by selecting a range of admissible values for the parameters to be studied. In this case, if the parameterization can be posed in a separated fashion, a PGD solver can be applied, as it will be shown in chapter 3. In addition to the reduced order modeling, the particular PGD separated solution is well suited to be post-processed in real-time as it will shown in chapter 4 of this document.

The parametric problem will be presented both in 2D and 3D for a unit cell with an inverted honeycomb shape. The 2D shape serves as an introduction to the parametric design and is the basis of the work performed on the 3D case.

2.2.1 The 2D unit cell

It was explained in the introduction that the problem from previous work [16] is used as basis, so it is presented in detail here. A 2D unit cell was proposed

with an inverted honeycomb shape which allows the macrostructure to display an auxetic behavior. The structure was parametrized with 4 variables:

- a: the length of the oblique beam element.
- b: the length of the horizontal beam element.
- t: thickness of the beam elements.
- α: the angle of inclination of the oblique beam elements.

The unit cell and its parametrization can be seen in figure 2.3.

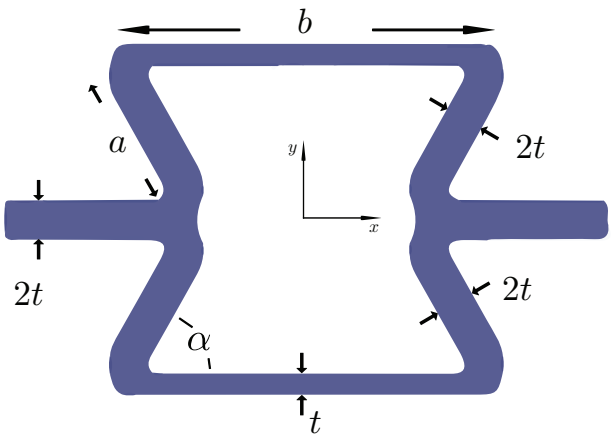


Figure 2.3: 2D cell parametrization of the inverted honeycomb.

Using beam elements, the cell can be modelled as shown in image 2.4:

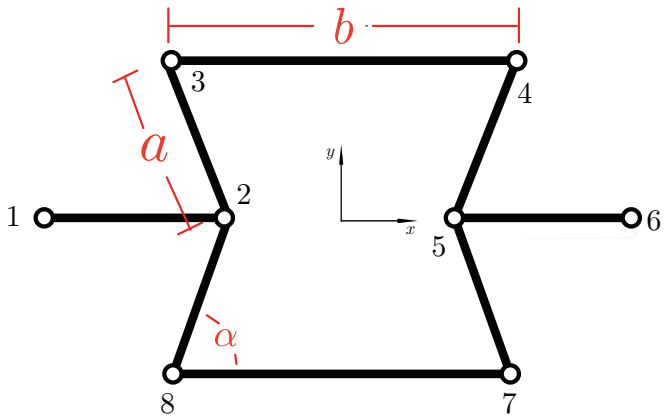


Figure 2.4: 2D cell: discretization with beam elements.

In figure 2.4 can be seen better the application of the elements, as they combine to create a simple structure formed by 1D elements that can withstand the

effects of forces and moments. The structure is modeled with eight elements and eight nodes, which capture in a simplified manner the behavior of the union of elements. The intermediate values between nodes can be estimated through beam theory and depend on the displacements and curvatures measured on the nodes. It is worth mentioning that, since this is a 2D problem, the number of d.o.f.'s per node is reduced to three, since only two displacements and one curvature is measured. Aside from this fact, all other postulates remain the same.

2.2.2 The 3D unit cell

The 3D unit cell to solve is the one shown in figure 2.5. It is defined as a tessellation of planes of 2D lattices, intersected in the orthogonal direction by a tessellation of planes of similar design, giving an inverted honeycomb shape in three dimensions.

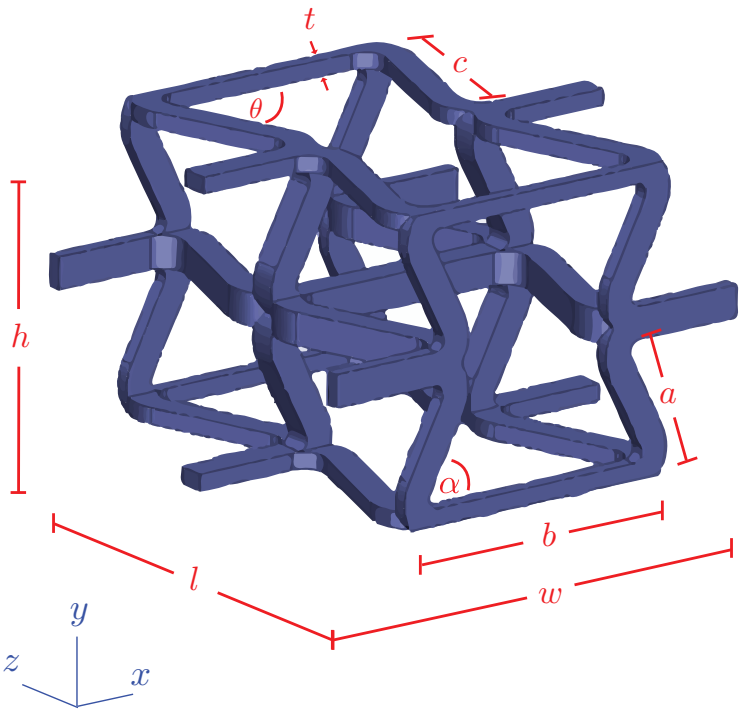


Figure 2.5: Unit Cell, 3D case.

In here, the structure is initially described by $n_p = 6$ parameters, with the first four being equivalent to the 2D case, defined for a face of the cell in the XY plane. The other parameters involved are:

- Φ: multiplication factor to define the length of oblique elements in plane XZ. In this plane, the length is defined as $c = \Phi \cdot a$.
- θ: the angle of inclination of the oblique elements in the plane XZ.

The parameter Φ is introduced as a way to ease the definition of coordinates and matrices for the cell. Considering the value c as the length of oblique beams in the plane XZ, the cell is represented as shown in figure 2.6:

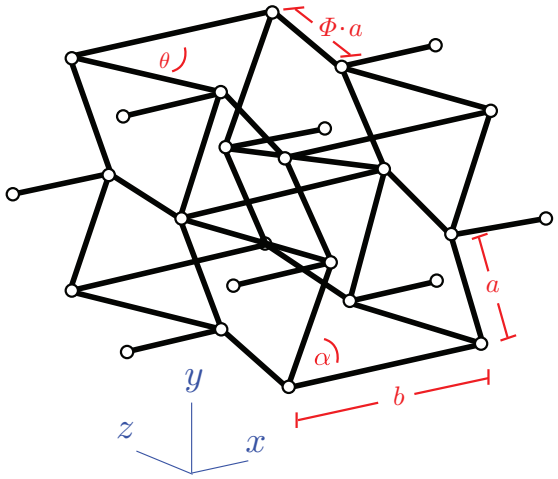


Figure 2.6: Unit Cell, 3D case with alternative formulation.

from here, the physical restrictions of the cell are described as

$$\begin{aligned} b &\geq 2a\cos\alpha \\ b &\geq 2c\cos\theta. \end{aligned} \tag{2.1}$$

which is basically the imposition of no overlapping between oblique elements, which discards all non plausible cases. Due to the definition of the cells and the dependence of the values of a and c respecting b, they are constricted with the relation

$$a\cos\alpha = c\cos\theta, \tag{2.2}$$

that allows to remove c as a parameter and express it in terms of a, α and θ. Using the parameter Φ allows to rewrite the previous expression in order to express θ as a parameter in function of the other parameters:

$$a\cos\alpha = \Phi a\cos\theta \tag{2.3}$$

This limits the scope of the problem, since now it is not only the values of a and c the ones that are linked, but also those of α and θ. Considering all previously mentioned, the parameters vector $\boldsymbol{\mu}$ is now of dimension $n_p = 5$ and is defined as

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \end{bmatrix} = \begin{bmatrix} a \\ b \\ \alpha \\ \Phi \\ t \end{bmatrix}.$$

The parameters have intervals defined to range between acceptable cases due to the already mentioned physical restrictions. The ranges are

$$\begin{cases} a \in I_1 = [0.3, 0.7] \\ b \in I_2 = [1.0, 1.5] \\ \alpha \in I_3 = [45, 135] \\ \Phi \in I_4 = [0.75, 2.00] \\ t \in I_5 = [0.01, 0.10] \end{cases}$$

All the parameters are discretized with 51 steps except for α, where 91 steps were taken. The parameter θ should still be considered since it serves the purpose of orienting the oblique beam elements. From equation 2.3 it can be seen that the term $\cos\theta$ can be replaced by a relation dependent on Φ and α (i.e., $\cos\theta = \cos\alpha/\Phi$). However, the replacement of $\sin\theta$ is not trivial, since there is no approximation that can be done in separable terms, which is a requirement for the computation using the PGD scheme. The approach taken in this work with respect to $\sin\theta$ is to approximate the dependence of this term in parameters α and Φ in a separated form using the Singular Value Decomposition (SVD) method. The procedure is shown in appendix A, and allows to represent $\sin\theta$ using only three modes dependent in a separated way only on Φ and α.

2.3 Structural problem resolution

Since the problem is being solved through the use of the finite element method, it is required the definition of a stiffness matrix for each element, as well as a force term. The definition of the force vector is done analogously to the displacements one in order to fit the principal forces and moments, and is given as

$$\boldsymbol{f} = \begin{pmatrix} N_x \\ V_y \\ V_z \\ T_x \\ M_y \\ M_z \end{pmatrix},$$

where N represents axial forces, V shear forces, T is the torsional moment and M representing the bending moments in the beam.

The computation of the stiffness matrix is already known and obtainable through different methods such as the direct stiffness method or following the

derivation of the finite element method through the use of the principle of virtual work, and its not on the scope of this docment. By application of the direct stiffness method [18], the stiffness matrix obtained for the 12 d.o.f beam element is

$$[K]^e = \begin{bmatrix} X & 0 & 0 & 0 & 0 & 0 & -X & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_1 & 0 & 0 & 0 & Y_2 & 0 & -Y_1 & 0 & 0 & 0 & Y_2 \\ 0 & 0 & Z_1 & 0 & -Z_2 & 0 & 0 & 0 & -Z_1 & 0 & -Z_2 & 0 \\ 0 & 0 & 0 & S & 0 & 0 & 0 & 0 & 0 & -S & 0 & 0 \\ 0 & 0 & -Z_2 & 0 & Z_3 & 0 & 0 & 0 & Z_2 & 0 & Z_4 & 0 \\ 0 & Y_2 & 0 & 0 & 0 & Y_3 & 0 & -Y_2 & 0 & 0 & 0 & Y_4 \\ -X & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & 0 \\ 0 & -Y_1 & 0 & 0 & 0 & -Y_2 & 0 & Y_1 & 0 & 0 & 0 & -Y_2 \\ 0 & 0 & -Z_1 & 0 & Z_2 & 0 & 0 & 0 & Z_1 & 0 & Z_2 & 0 \\ 0 & 0 & 0 & -S & 0 & 0 & 0 & 0 & 0 & S & 0 & 0 \\ 0 & 0 & -Z_2 & 0 & Z_4 & 0 & 0 & 0 & Z_2 & 0 & Z_3 & 0 \\ 0 & Y_2 & 0 & 0 & 0 & Y_4 & 0 & -Y_2 & 0 & 0 & 0 & Y_3 \end{bmatrix}$$

where

$$X = \frac{EA}{L} \qquad Y_1 = \frac{12EI_z}{L^3} \qquad Y_2 = \frac{6EI_z}{L^2}$$
$$Y_3 = \frac{4EI_z}{L} \qquad Y_4 = \frac{2EI_z}{L} \qquad S = GJ/L$$
$$Z_1 = \frac{12EI_y}{L^3} \qquad Z_2 = \frac{6EI_y}{L^2} \qquad Z_3 = \frac{4EI_y}{L}$$
$$Z_4 = \frac{2EI_y}{L}$$

In these equations, A is the sectional area, L is the length of the element, E correspond to the Young modulus of the material, G is the shear modulus defined as $G = E/2(1+\nu)$, ν is the Poisson's ratio of the material, I is the moment of inertia of the element in each axis and J is the torsional inertia.

The stiffness matrix defined corresponds to only one element, which axes are oriented in the same direction as the global reference axes. In order to represent different orientations, the stiffness matrix is pre and post multiplied by a transformation matrix, expressed mathematically as

$$[\boldsymbol{K}]_t^e = [\boldsymbol{\Lambda}]^T [\boldsymbol{K}^e][\boldsymbol{\Lambda}],$$

where $\boldsymbol{\Lambda}$ is defined as

$$\Lambda = \begin{bmatrix} \boldsymbol{T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T} \end{bmatrix}$$

and

$$\boldsymbol{T} = \begin{bmatrix} \cos\alpha_x & \cos\alpha_y & \cos\alpha_z \\ \cos\beta_x & \cos\beta_y & \cos\beta_z \\ \cos\gamma_x & \cos\gamma_y & \cos\gamma_z \end{bmatrix}$$

where α_i, for $i = x, y, z$, represents the angle between the local axis x and the corresponding global axis, β_i is the same relationship between local axis y and the global axes and γ_i the relationship for local axis z.

Finally, since the material is not composed by only one element, the stiffness matrices of all the beam elements should be combined into a global stiffness matrix, and this is done using the standard assembly operator defined as

$$[\boldsymbol{K}] = \mathop{\mathsf{A}}_{e}[\boldsymbol{K}]^e \tag{2.4}$$

which distributes the local stiffness matrices into their corresponding nodes in the global numbering. After computing this term, the equation takes the usual matrix form

$$\boldsymbol{K} \cdot \boldsymbol{U} = \boldsymbol{F}.$$

Typically, the unknowns of the problem are the nodal displacements and rotations given by the vector $\boldsymbol{U}$. Since in the 3D problem each node contains 6 degrees of freedom, the size of the solution vector is $n_{dof} = 6N$, with N being the number of nodes of the problem (meaning $\boldsymbol{U} \in \mathbb{R}^{n_{dof}}$). To obtain the nodal displacements, the equation is solved given the values of system stiffness matrix and nodal forces and moments, or $\boldsymbol{K} \in \mathbb{R}^{n_{dof} \times n_{dof}}$ and $\boldsymbol{F} \in \mathbb{R}^{n_{dof}}$, respectively.

The problem will be solved by the Proper Generalized Decomposition (PGD) method. The main idea of PGD consist on the reduction of a multidimensional domain via an approximate solution considering separated continuous functions, each one taking only one of the parameters μ_i that define the problem. This parametric equation is expressed as

$$\boldsymbol{K}(\boldsymbol{\mu}) \cdot \boldsymbol{U}(\boldsymbol{\mu}) = \boldsymbol{F}(\boldsymbol{\mu}), \tag{2.5}$$

where the dependence on $\boldsymbol{\mu}$ is explicitly written in order to express the separability of the terms. The process and details of the method are explained in section 3.1.

2.4 Solving the parametric auxetic material

The main objective of this work is to solve an architechured material with the proposed 3D unit cell, in order to understand the behavior of auxetic materials. These type of materials are designed as a repetition that extends in every direction, so periodicity is the property that links the behavior between the unit cell and the macro-scale. Two cases are presented here: the first one is the homogenized unit cell, which consists in considering only one entity in a lattice, constrained by periodic boundary conditions in order to simulate the behavior that is observed in the interior of the lattice. The second case is the computation of a lattice material whose pattern is conformed by a prescribed amount of unit cells repeated in the periodic directions. This will show the behavior of a full-scale problem, which captures some effects that cannot be appreciated in the unitary case, such as boundary deformation.

2.4.1 Homogenization of the unit cell

The generalization of the results at a macro-scale can be done by taking advantage of the periodicity of the structure. Since the problem dealt is of a linear fashion, the deformation of the problem can be obtained as a linear combination of known terms. The homogenization theory defines six independent load cases at the macro-scale in order to study the behavior of the cell: three uniaxial and three shear strains, shown in figure 2.7.

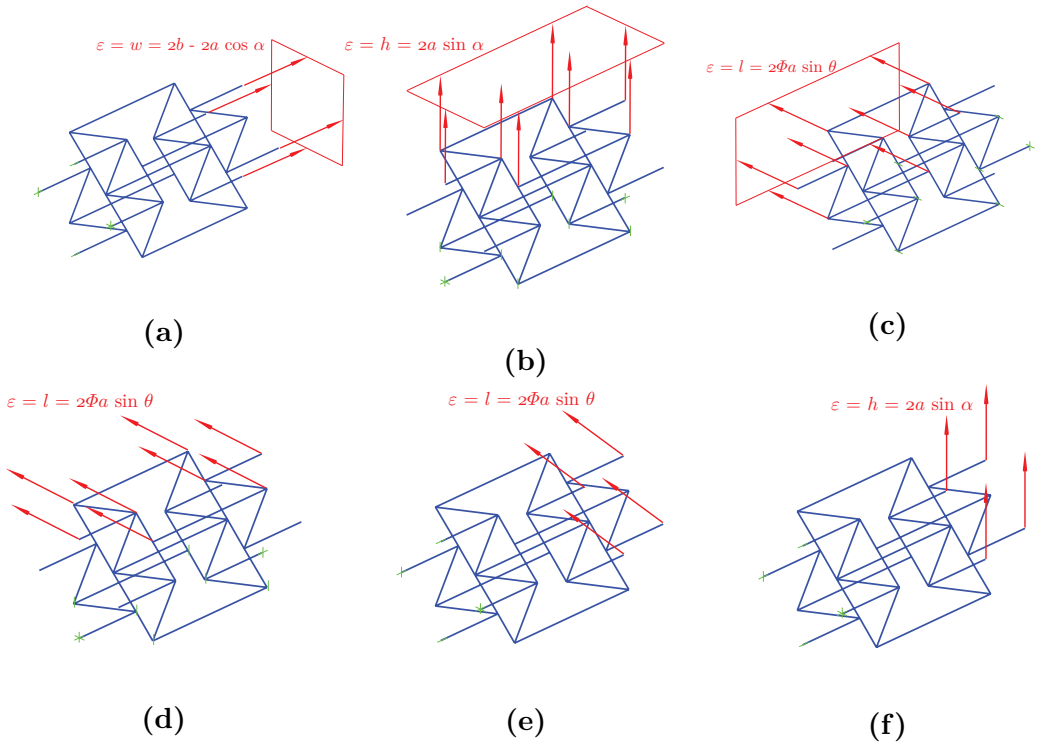


Figure 2.7: Load cases for the homogenized cell. **a)** Load case XX. **b)** Load case YY. **c)** Load case ZZ. **d)** Load case YZ. **e)** Load case XZ. **f)** Load case XY.

To obtain the homogenized solutions, the strains are applied to the unit-cell at the micro-scale problem, which is restricted through boundary conditions of periodicity in order to account for the transmission of effect in the lattice. The applied deformations on the unit cell will give as results $\boldsymbol{U}^i$ for $i = 1, 2, \ldots, 6$, which account for the displacements arising from the loading cases of the three uniaxial strains and the three shears. These in turn will provide the effective constitutive Hooke tensor for the homogenized material. For an orthotropic material, the tensor is expressed as [19]

$$\boldsymbol{C}^{\mathtt{eff}} = \begin{bmatrix} C_{11}^{\mathtt{eff}} & C_{12}^{\mathtt{eff}} & C_{13}^{\mathtt{eff}} & 0 & 0 & 0 \\ C_{21}^{\mathtt{eff}} & C_{22}^{\mathtt{eff}} & C_{23}^{\mathtt{eff}} & 0 & 0 & 0 \\ C_{31}^{\mathtt{eff}} & C_{32}^{\mathtt{eff}} & C_{33}^{\mathtt{eff}} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44}^{\mathtt{eff}} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55}^{\mathtt{eff}} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66}^{\mathtt{eff}} \end{bmatrix}. \tag{2.6}$$

where

$$C_{IJ}^{\mathtt{eff}}(\boldsymbol{\mu}) = \frac{1}{whl}(\boldsymbol{U}^I(\boldsymbol{\mu}))^T \boldsymbol{K}(\boldsymbol{\mu}) \boldsymbol{U}^J(\boldsymbol{\mu}), \quad \text{for } I, J = 1, \ldots, 6. \tag{2.7}$$

In 2.7, whl corresponds to the volume of the cube circumscribed to the unit cell, where w is the length in the X direction, h the one in Y direction and l the corresponding to the Z direction (as shown in figure 2.5), and it is used as a way of averaging the tensor. The effective Hooke tensor provides the macro-scale properties of the architectured material in the ideal case of an infinite repetition of unit cells in all directions of periodicity, without the need of computing the full extent of the lattice. In addition, using certain elements of this tensor allows to obtain effective values of the Poisson's coefficient of the homogenized cell, which is of particular interest in the case of auxetic materials, and are defined as

$$\begin{aligned} \nu_{12}^{\mathtt{eff}} &= \frac{C_{12}^{\mathtt{eff}}}{C_{22}^{\mathtt{eff}}}, \quad \nu_{21}^{\mathtt{eff}} = \frac{C_{12}^{\mathtt{eff}}}{C_{11}^{\mathtt{eff}}}, \quad \nu_{13}^{\mathtt{eff}} = \frac{C_{13}^{\mathtt{eff}}}{C_{33}^{\mathtt{eff}}}, \\ \nu_{31}^{\mathtt{eff}} &= \frac{C_{13}^{\mathtt{eff}}}{C_{11}^{\mathtt{eff}}}, \quad \nu_{23}^{\mathtt{eff}} = \frac{C_{23}^{\mathtt{eff}}}{C_{33}^{\mathtt{eff}}}, \quad \nu_{32}^{\mathtt{eff}} = \frac{C_{23}^{\mathtt{eff}}}{C_{22}^{\mathtt{eff}}}. \end{aligned} \tag{2.8}$$

The main property of auxetic materials was the fact that they have a negative Poisson's coefficient value, which makes these results the goal of performing the homogenization, since they allow to study how these ratios change given the ranges of parameters chosen for the simulation.

Loading and boundary conditions

Homogenization of the properties of the cell to be extended to the macro-scale problem depends integrally on the periodicity of the structure analyzed. This

implies that, in the study case of a single cell, the boundary conditions should be applied in order to account for this periodicity inside a structure. A unit cell immersed in a lattice can be seen in figure 2.8, as well as a schematic cell with the nodal numbering in figure 2.9.

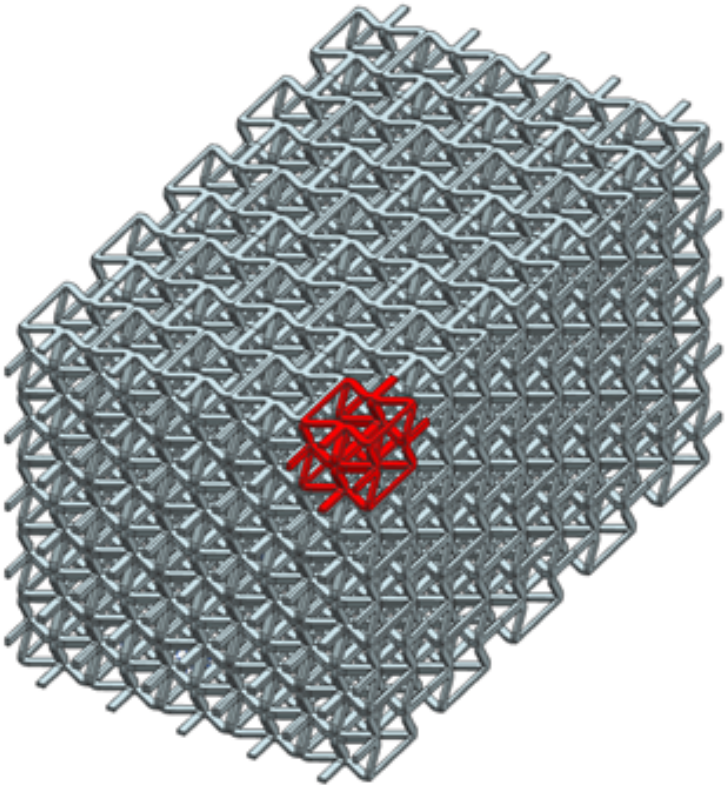


Figure 2.8: Unit cell inside a periodic lattice.

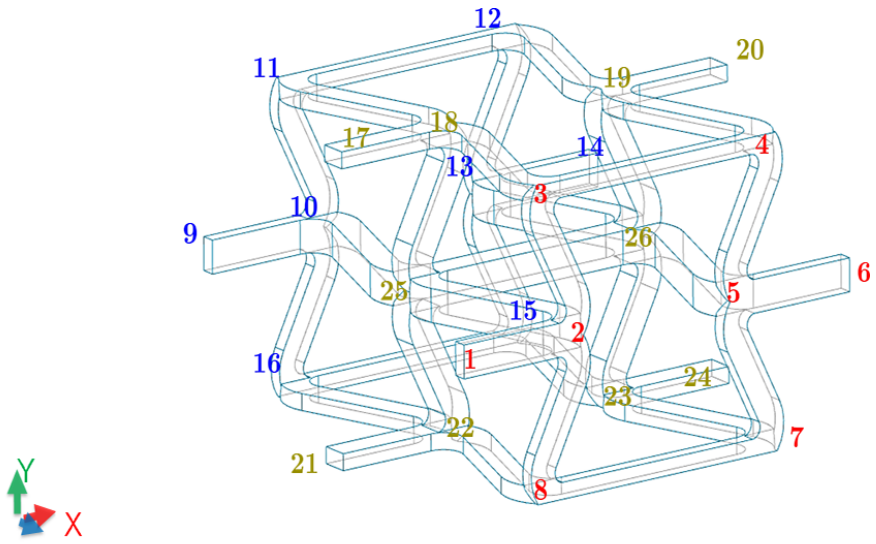


Figure 2.9: Numbering and orientation of the unit cell.

Following the nodal notation introduced here and the axis orientation, some observations can be made:

- Due to continuity of the structure in the X axis, nodes 1, 9, 17 and 21 connect to nodes 6, 14, 20 and 24 respectively on the previous cell.
- Applying the same logic to the Z axis, nodes 1 to 8 from one cell connect to the nodes 9 to 16 in the next cell, respectively.
- For axis Y nodes 3, 4, 11, 12 and 17 to 20 are related to the nodes 7, 8, 15, 16 and 21 to 24 respectively for two contiguous cells.
- The only nodes not affected by periodicity are nodes 25 and 26, since the lie inside the unit cell and they are not linked to the corresponding nodes on the next cells.

To obtain homogenization it is necessary to impose strain cases on the cell relating the nodes as mentioned. A diagram for the infinitesimal strain theory is shown in the following figure:

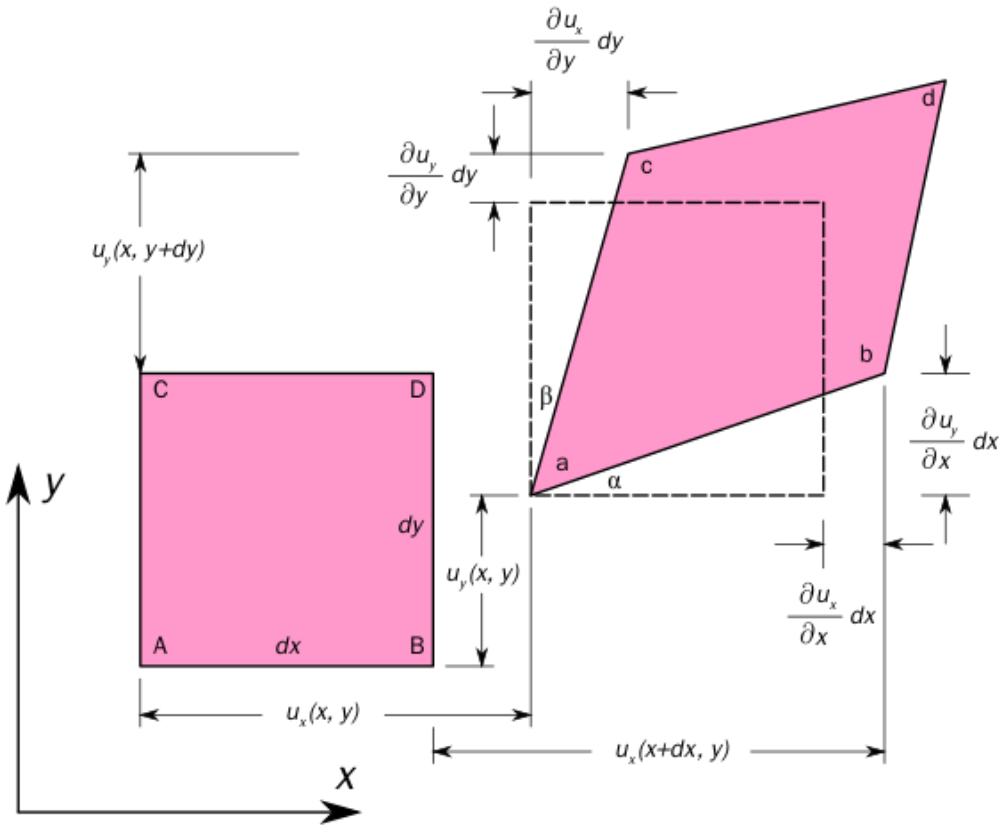


Figure 2.10: Infinitesimal strain theory (Source: Wiki Commons).

As it can be seen, imposing a deformation in one side leads to an infinitesimal increment on the deformation. Knowing that $\varepsilon_{ij} = \partial u_i/\partial j$ for $j = x, y, z$, and

accounting that the nodes in both cells should be related, the relation between these can be written in a general way as

$$\boldsymbol{u}_{n_2} - \boldsymbol{u}_{m_1} = \ell\boldsymbol{\varepsilon},$$

where $\boldsymbol{u}$ is the deformation, with $n_2 = x, y, z$ and $m_1 = x, y, z$ representing nodes in two consecutive unit cells, $\boldsymbol{\varepsilon}^{IJ}$ is the uniaxial strain imposed on the nodes for $I, J = X, Y, Z$ and ℓ being the distance between the two consecutive nodes (in the direction of interest). Knowing this, the application of the strain in the cell for the six load cases proposed is written as

$$\begin{aligned}
\boldsymbol{u}_{i_2} - \boldsymbol{u}_{i_1} &= w\boldsymbol{\varepsilon}^{XX} \quad \text{for load case XX,} \\
\boldsymbol{u}_{j_2} - \boldsymbol{u}_{j_1} &= h\boldsymbol{\varepsilon}^{YY} \quad \text{for load case YY,} \\
\boldsymbol{u}_{k_2} - \boldsymbol{u}_{k_1} &= l\boldsymbol{\varepsilon}^{ZZ} \quad \text{for load case ZZ,} \\
\boldsymbol{u}_{j_2} - \boldsymbol{u}_{j_1} &= l\boldsymbol{\varepsilon}^{YZ} \quad \text{for load case YZ,} \\
\boldsymbol{u}_{i_2} - \boldsymbol{u}_{i_1} &= l\boldsymbol{\varepsilon}^{XZ} \quad \text{for load case XZ,} \\
\boldsymbol{u}_{i_2} - \boldsymbol{u}_{i_1} &= h\boldsymbol{\varepsilon}^{XY} \quad \text{for load case XY,}
\end{aligned} \tag{2.9}$$

with w, l and h the lengths of the cell, explained in figure 2.5 and

$$\begin{aligned}
i_2 &= 1, 9, 17, 21 \\
i_1 &= 6, 14, 20, 24 \\
j_2 &= 7, 8, 15, 16, 21, 22, 23, 24 \\
j_1 &= 3, 4, 11, 12, 17, 18, 19, 20 \\
k_2 &= 1, 2, 3, 4, 5, 6, 7, 8 \\
k_1 &= 9, 10, 11, 12, 13, 14, 15, 16
\end{aligned}$$

For the constraints of the problem, the application is not trivial. Given that some nodes are shared between faces (i.e. node 3 between $Y+$ and $Z+$) the system can easily become overconstrained when applying boundary conditions through lagrange multipliers or direct methods, leading to ill-conditioned matrices, solutions with no physical meaning or the requirement of penalty methods. In order avoid this, the conditions need to be applied following an ordering strategy. In this work it is followed the approach shown in [20], which methodically establishes relationships between the nodes in order to avoid repetition of boundary conditions.

2.4.2 Full structure pattern design

Homogenization of the unit cell works as a fast solution to represent the behavior of a periodic material and to estimate its properties, but it has some drawbacks. From the structural point of view, the homogenized case only represents the behavior of the average cell in the pattern, representing only the interior parts. In this case, the unit cells positioned in the borders of the material are not accounted

for, which may be of interest when designing lattices. From a user point of view, the homogenized solution is presented as a highly technical concept that aims to people with knowledge on the subject. For any common user this does not say much, since a full scale problem is being represented by one single entity, defying the notion of a periodic structure. Therefore, a parametric solution of a full pattern, is presented also in this document in order to model the complete behavior of an auxetic material in the global scale. In addition, it is also used to measure the resulting Poisson's ratios and compare them against the effective ones computed by homogenization.

For this analysis, a full pattern structure of $3 \times 3 \times 3$ unit cells with three uniaxial load cases are presented, similar to the ones shown for the single unit cell, as shown in the figure below:

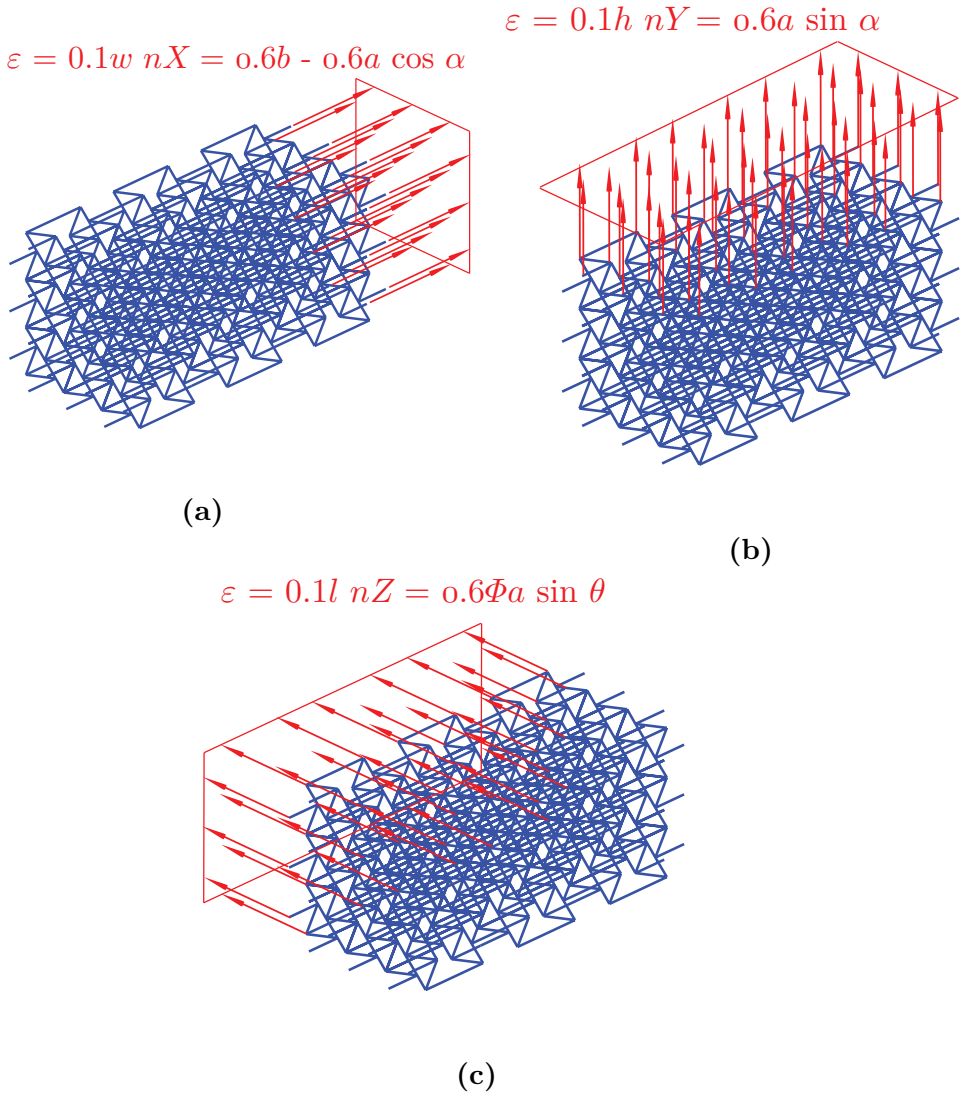


Figure 2.11: Load cases for the full pattern in 3D. **a)** Load case XX. **b)** Load case YY. **c)** Load case ZZ.

which are an extension of the solutions previously developed for the 2D case, as shown in the figure below [16]:

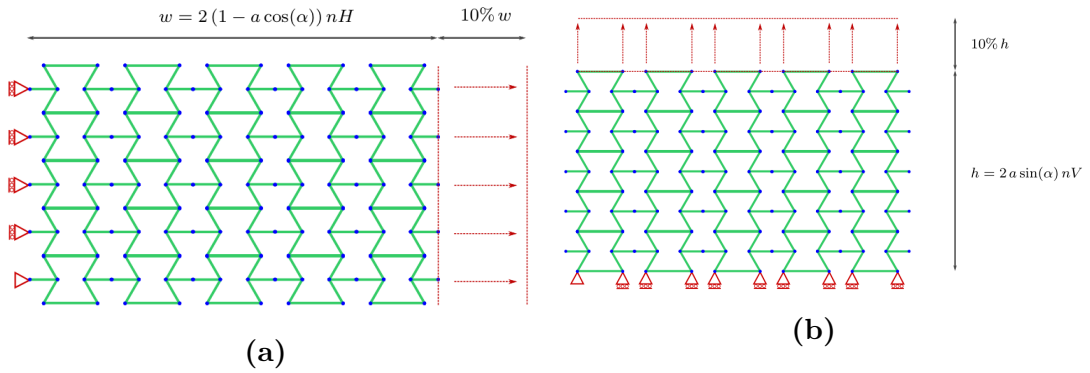


Figure 2.12: Load cases for the full pattern in 2D (Source: Sibileau, 2018). **a)** Load case XX. **b)** Load case YY.

The strains are applied as shown in the figure 2.11, equally applied to all thenodes on one of the faces of the scaffold. The boundary conditions are applied in the opposite face of the strain, where all the nodes are restricted in the same direction as the strain. In order to avoid having a hyperstatic problem, one of the nodes (chosen in a corner) is completely restrained to movement (with only rotations allowed). The images for the 2D case serve as a reference. The application is analogous in the 3D problem and directly applied.

Effective Poisson's coefficients

The same exercise as in the homogenized case is repeated in this section. The main difference is that an effective value of the strain tensor is not obtainable for this case, so an approximation of the Poisson's ratio is calculated. To perform this approximation, a ratio between the imposed strain and the averaged ones on the orthogonal directions is computed, namely

$$\nu_{ij} = -\frac{\varepsilon_{jj}}{\varepsilon_{ii}}, \tag{2.10}$$

with $i = X, Y, Z$ the imposed strain, which is a known value, and $j = X, Y, Z$ the corresponding orthogonal axes strain, which is taken as

$$\varepsilon_{jj} = \frac{\ell_f(\boldsymbol{\mu}) - \ell_0(\boldsymbol{\mu})}{\ell_0(\boldsymbol{\mu})}.$$

In this expression, $\ell_0(\boldsymbol{\mu})$ is the length of the undeformed scaffold in the axis of interest, while $\ell_f(\boldsymbol{\mu})$ is the deformed state after imposing the strain, depending on the parameters chosen. An example of how the values are taken is shown in the next figure [16]:

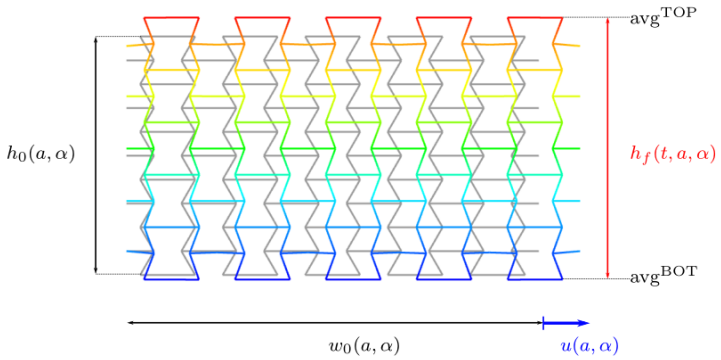


Figure 2.13: Effective Poisson's coefficients for a full pattern (Source: Sibileau, 2018).

The figure represents how it was performed in the 2D full pattern example, but the application to the 3D case is analogous, performed in any of the three axis of interest.

Chapter 3

Proper Generalized Decomposition: solution of the parametric problem

Proper Generalized Decomposition (PGD) is a good strategy in order to circumvent the efficiency difficulties of a big sized parametric problem. It is because of this that the formulation of the problem looks different from the usual Finite Element problem, since the parametric dependence must be expressed in a separated way.

PGD relies on a parametric set of variables to represent the geometry and properties of the problem to be analyzed. The amount of parameters to be used depend on the required freedom for the system and it will have a impact on the efficiency and speed of the problem solving. In a general way of speaking, the problem is defined by μ_i parameters, with $i = 1, 2, \ldots, n_{\mathtt{p}}$, and each one in an real interval $I_i \subset \mathbb{R}$. The parameters are expressed as a vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_i]^T$, in order to simplify the notation of the equations to come. This vector of course ranges in a multidimensional interval given by $\mathcal{D} = I_1 \times I_2 \times \cdots \times I_{n_{\mathtt{p}}} \subset \mathbb{R}^{n_{\mathtt{p}}}$.

As was introduced in equation 2.5, the problem input terms are dependent on $\boldsymbol{\mu}$, so the solution is also dependent on it, with every term belonging to the space interval $\mathcal{D}$ already mentioned. For the sake of completeness and solvability, it is provided that the dependence of the terms with respect to the parameters are regular enough in order to provide square integrability. In technical fashion, it is provided that $\boldsymbol{F}(\boldsymbol{\mu}) \in [\mathcal{L}_2(\mathcal{D})]^{n_{\mathtt{dof}}}$ and $\boldsymbol{K}(\boldsymbol{\mu}) \in [\mathcal{L}_2(\mathcal{D})]^{n_{\mathtt{dof}} \times n_{\mathtt{dof}}}$. It is also given that the functional space $\mathcal{L}_2(\mathcal{D})$ is expressed in term of the sectional spaces $\mathcal{L}_2(I_i)$, so even setting some parameters to a given value would still provide a square integrable function. As usual, matrix $\boldsymbol{K}$ is formed as an assembly of the different stiffness matrices associated to every element of the unit cell. It is worth mentioning that the presented parametric formulation works for every kind of beam element, and it is not limited to the Euler-Bernoulli formulation used in this work.

Since the formulation of the Finite Element Method is used, equation 2.5 can be written in an integral form. Writing the residual gives

$$\boldsymbol{R}(\boldsymbol{U}(\boldsymbol{\mu})) := \boldsymbol{F}(\boldsymbol{\mu}) - \boldsymbol{K}(\boldsymbol{\mu})\boldsymbol{U}(\boldsymbol{\mu}). \tag{3.1}$$

Using weighted residuals, it is known that $\boldsymbol{U}(\boldsymbol{\mu})$ is a solution of the problem if it minimizes the following expression:

$$\int_{I_1}\int_{I_2}\cdots\int_{I_{n_\mathtt{p}}} \delta\boldsymbol{U}(\boldsymbol{\mu})^T \boldsymbol{R}(\boldsymbol{U}(\boldsymbol{\mu}))\; d\mu_{n_\mathtt{p}}\ldots d\mu_2 d\mu_1 = 0, \tag{3.2}$$

for all $\delta\boldsymbol{U}(\boldsymbol{\mu}) \in [\mathcal{L}_2(\mathcal{D})]^{n_\mathtt{dof}}$.

As it can be seen from the previous equations, the integration is performed in the parametric space $\mathcal{D}$ and never on the physical space, due to the algebraic nature of equation 2.5, which can be seen as an already discretized space. The space integration in here is replaced by the product of the residual and the test function, so the energy product is represented by a product of forces and virtual displacements.

To obtain an equivalent problem to the ones described in equations 2.5 and 3.2 in parametric form, sectional spaces are discretized in finite-dimensional spaces written as $V_i \subset \mathcal{L}_2(I_i)$, with $i = 1, 2, \ldots, n_\mathtt{p}$. Each one of these spaces is of dimension $n_{d,i}$, which means that the solution $\boldsymbol{U}(\boldsymbol{\mu})$ lies in the space $[V_1 \otimes V_2 \otimes \cdots \otimes V_{n_\mathtt{p}}]^{n_\mathtt{total}}$, where

$$n_\mathtt{total} = n_\mathtt{dof} \prod_{i=1}^{n_\mathtt{p}} n_{d,i}. \tag{3.3}$$

As can be seen from this relation, the problem grows exponentially with $n_\mathtt{p}$, so the choosing of parameters needs to be carefully done in order to avoid increasing the size of the problem unnecessarily. The solution obtained from the PGD problem is written as a tabulated result depending on the parameters chosen and the intervals selected for each one of them. Having these results allows to obtain a real-time solution from the combination of the specific values needed. The great advantage from this method is the fact that the approximation to the solution is tabulated by dimension (and not the tensor product of them), which proves to be efficient in memory allocation and provides fast access to the values.

3.1 Solving the PGD problem

The previous section set the basic information about parametric equations and how the problem will be handled. In order to solve it, the process for obtaining

the previous terms is explained.

3.1.1 Separable approximations

In order to follow the parametric form, $\boldsymbol{U}(\boldsymbol{\mu})$ is approximated with a separable $\boldsymbol{U}^n_{\texttt{PGD}}(\boldsymbol{\mu})$ given as

$$\begin{aligned}\boldsymbol{U}(\boldsymbol{\mu}) \approx \boldsymbol{U}^n_{\texttt{PGD}}(\boldsymbol{\mu}) &= \sum_{m=1}^{n} \boldsymbol{u}^m G_1^m(\mu_1) G_2^m(\mu_2) \cdots G_{n_\texttt{p}}^m(\mu_{n_\texttt{p}}) \\ &= \sum_{m=1}^{n} \boldsymbol{u}^m \prod_{i=1}^{n_\texttt{p}} G_i^m(\mu_i) \\ &= \boldsymbol{U}^{n-1}_{\texttt{PGD}}(\boldsymbol{\mu}) + \boldsymbol{u}^n G_1^n(\mu_1) G_2^n(\mu_2) \cdots G_{n_\texttt{p}}^n(\mu_{n_\texttt{p}}),\end{aligned} \tag{3.4}$$

with n being the number of terms in the approximation $\boldsymbol{U}^n_{\texttt{PGD}}(\boldsymbol{\mu})$ and $m = 1, 2, \ldots, n$. Each term is determined by a displacement vector $\boldsymbol{u}^m$ that describes the spatial mode; and also described by a set of parametric modes ($G_i^m(\mu_i)$, from now on G_i^m) that ranges from $i = 1, 2, \ldots, n_\texttt{p}$ and represent each one of the parameters chosen for the problem. For PGD, and as stated before, $\boldsymbol{K}$ and $\boldsymbol{F}$ also should be written in a separable fashion, given as

$$\boldsymbol{K}(\boldsymbol{\mu}) = \sum_{k=1}^{n_\texttt{k}} \boldsymbol{K}^k \prod_{i=1}^{n_\texttt{p}} B_i^k(\mu_i), \tag{3.5}$$

$$\boldsymbol{F}(\boldsymbol{\mu}) = \sum_{j=1}^{n_\texttt{f}} \boldsymbol{f}^j \prod_{i=1}^{n_\texttt{p}} S_i^j(\mu_i), \tag{3.6}$$

where $n_\texttt{k}$ and $n_\texttt{f}$ are the number of terms in which the original $\boldsymbol{K}$ and $\boldsymbol{F}$ are separated respectively, $\boldsymbol{K}^k$ and $\boldsymbol{f}^j$ correspond to the spatial modes and finally B_i^k and S_i^j (following the same notation adopted for G_i^m) correspond to the parametric modes associated to the parameters chosen.

3.1.2 Sectional norms

Solving a PGD problem focuses in discretizing the integral form shown in equation 3.2 with the separable approximations. This in turn requires the introduction of the called *sectional norms*, which are the norm of the parametric functions in each parametric dimension, which allows to measure the modes.

For the parametric modes describing the approximated solution $\boldsymbol{U}_{\texttt{PGD}}$ (i.e.,

$G_i^m \in \mathcal{L}_2(I_i)$), the standard $\mathcal{L}_2$ norm is used:

$$||G_i^m||^2 = \int_{I_i} (G_i^m)^2 \, d\mu_i. \tag{3.7}$$

For the spatial mode the choice is not as trivial as the parametric ones. Using an Euclidean norm approach (i.e., $||\boldsymbol{u}^m||^2 = (\boldsymbol{u}^m)^T \boldsymbol{u}^m$) does not account for the nature of the problem and mixes displacements and rotations, so the resulting measure lacks of physical meaning. The suitable choice for this problem is using a structural mass matrix $\boldsymbol{M}_u$, which results in a norm given by

$$||\boldsymbol{u}^m||^2 = [\boldsymbol{u}^m]^T \boldsymbol{M}_u \boldsymbol{u}^m. \tag{3.8}$$

Having obtained the norms, each mode can be normalized as follows

$$\tilde{\boldsymbol{u}}^m = \frac{\boldsymbol{u}^m}{||\boldsymbol{u}^m||} \quad \text{and} \quad \tilde{G}_i^m = \frac{G_i^m}{||G_i^m||},$$

allowing to rewrite $\boldsymbol{U}_{\texttt{PGD}}^n$ as

$$\boldsymbol{U}_{\texttt{PGD}}^n(\boldsymbol{\mu}) = \sum_{m=1}^{n} \beta^m \tilde{\boldsymbol{u}}^m \prod_{i=1}^{n_\texttt{p}} \tilde{G}_i^m. \tag{3.9}$$

In the previous equation, $\beta^m = ||\boldsymbol{u}^m|| \prod_{i=1}^{n_\texttt{p}} ||G_i^m||$ which corresponds to the amplitude of the term m of the complete sum. β^m provides information of the different modes, mainly on the importance of each mode in the separable approximation. It is because of this that it is used as one of the stopping criteria in order to decide the amount of terms to consider.

Finally, a global norm is also introduced in $[\mathcal{L}_2(\mathcal{D})]^{n_\texttt{dof}}$ such that

$$||\boldsymbol{U}(\boldsymbol{\mu})||^2_\texttt{glob} = \int_{I_1} \int_{I_2} \cdots \int_{I_{n_\texttt{p}}} \boldsymbol{U}(\boldsymbol{\mu})^T \boldsymbol{M}_u \boldsymbol{U}(\boldsymbol{\mu}) \, d\mu_{n_\texttt{p}} \ldots d\mu_2 d\mu_1, \tag{3.10}$$

for any $\boldsymbol{U}(\boldsymbol{\mu}) \in [\mathcal{L}_2(\mathcal{D})]^{n_\texttt{dof}}$.

3.1.3 Methodology and rank-1 approximation

In order to obtain a solution for the PGD problem it is necessary to obtain a solution of the form of equation 3.4. To do this, a greedy strategy is used, which consist in starting with a solution for $\boldsymbol{U}_{\texttt{PGD}}^1(\boldsymbol{\mu})$, and once this is computed, continue with $\boldsymbol{U}_{\texttt{PGD}}^2(\boldsymbol{\mu})$ and so on, until reaching $\boldsymbol{U}_{\texttt{PGD}}^n(\boldsymbol{\mu})$, for a desired amount of iterations or after achieving the desired amounts of modes given by the chosen

tolerance. Each step considers solving a rank-1 approximation problem. The previous statement can be written in equation form as

$$\boldsymbol{U}^n_{\texttt{PGD}}(\boldsymbol{\mu}) = \boldsymbol{U}^{n-1}_{\texttt{PGD}}(\boldsymbol{\mu}) + \boldsymbol{u}^n \prod_{i=1}^{n_\texttt{p}} G^n_i. \tag{3.11}$$

The problem is then reduced to finding the value of $\boldsymbol{u}^n$ and G_i for $i = 1, 2, \ldots, n_\texttt{p}$, such that $\boldsymbol{U}^n_{\texttt{PGD}}$ is a solution of equation 3.2. The complete unknown term $(\boldsymbol{u}^n \prod_{i=1}^{n_\texttt{p}} G^n_i)$ is considered as rank-1, since it is built as the product of sectional functions. This yields a non-linear problem, but with the advantage of having the degrees of freedom reduced from the original problem posed in eq. 3.2. With a PGD strategy, the number of degrees of freedom is given by an additive fashion, rather than a multiplicative one as seen in eq. 3.3. For PGD, the number of DOFs is given as

$$n_{\texttt{Rank1}} = n_{\texttt{dof}} + \sum_{i=1}^{n_\texttt{p}} n_{d,i}, \tag{3.12}$$

which for bigger problems (namely, more parameters) is $n_{\texttt{total}} \ll n_{RankOne}$.

Since the solution is now obtained in a recursive manner, the residuals can also be expressed in an explicit form, such that

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{U}^n_{\texttt{PGD}}) &= \boldsymbol{R}(\boldsymbol{U}^{n-1}_{\texttt{PGD}}) - \left[\sum_{k=1}^{n_\texttt{k}} \boldsymbol{K}^k \prod_{i=1}^{n_\texttt{p}} B^k_i\right] \boldsymbol{u} \prod_{i=1}^{n_\texttt{p}} G_i \\ &= \boldsymbol{R}(\boldsymbol{U}^{n-1}_{\texttt{PGD}}) - \sum_{k=1}^{n_\texttt{k}} \left[\prod_{i=1}^{n_\texttt{p}} B^k_i G_i\right] \boldsymbol{K}^k \boldsymbol{u}. \end{aligned} \tag{3.13}$$

Given the previous expression, the test function $\delta\boldsymbol{U}(\boldsymbol{\mu})$ from eq. 3.2 has to be chosen now in accordance with the unknown of the rank-1 problem, which is obtained by introducing the residual as written in eq. 3.13. The test function is written as a variation of the unknown term $(\boldsymbol{u}^n \prod_{i=1}^{n_\texttt{p}} G^n_i)$, and this yields

$$\delta\boldsymbol{U} = \delta\boldsymbol{u} \prod_{i=1}^{n_\texttt{p}} G^n_i + \sum_{j=1}^{n_\texttt{p}} \boldsymbol{u}^n \delta G_j \prod_{i=1, i\neq j}^{n_\texttt{p}} G^n_i. \tag{3.14}$$

The previous equation tells that in order to obtain a solution, a test solution $\delta\boldsymbol{U}$ is not taken, but rather an arbitrary $\delta\boldsymbol{u} \in \mathbb{R}^{n_\texttt{dof}}$ and $\delta G_i \in \mathcal{L}_2(I_i)$ for $i = 1, 2, \ldots, n_\texttt{p}$, with V_i replacing $\mathcal{L}_2(I_i)$ in the numerical version.

3.1.4 Alternated directions strategy

The obtained approximated problem is of a non-linear fashion, which complicates the solution of it. The strategy used to deal with this issue is the use of the alternated directions strategy, which solves successively for each one of the unknown parameters, assuming that the others are known.

Given this information, the first step is to compute $\boldsymbol{u}$, assuming that G_i is known for $i = 1, 2, \ldots, n_{\mathtt{p}}$. The consequence of G_i being known is the fact that δG_i is zero, so $\delta \boldsymbol{U}$ reduces to

$$\delta \boldsymbol{U} = \delta \boldsymbol{u} \prod_{i=1}^{n_{\mathtt{p}}} G_i,$$

which when replaced in 3.2 leads to

$$\int_{I_1} \int_{I_2} \cdots \int_{n_{\mathtt{p}}} \delta \boldsymbol{u}^T \boldsymbol{R}(\boldsymbol{U}^n_{\mathtt{PGD}}) \prod_{i=1}^{n_{\mathtt{p}}} G_i \; d\mu_{n_{\mathtt{p}}} \cdots d\mu_2 d\mu_1 = 0,$$

for all $\delta \boldsymbol{u} \in \mathbb{R}^{n_{\mathtt{dof}}}$. This equation is equivalent to

$$\int_{I_1} \int_{I_2} \cdots \int_{n_{\mathtt{p}}} \boldsymbol{R}(\boldsymbol{U}^n_{\mathtt{PGD}}) \prod_{i=1}^{n_{\mathtt{p}}} G_i \; d\mu_{n_{\mathtt{p}}} \cdots d\mu_2 d\mu_1 = \boldsymbol{0}.$$

Substituting $\boldsymbol{R}(\boldsymbol{U}^n_{\mathtt{PGD}})$ by equation 3.13 it is obtained

$$\int_{I_1} \int_{I_2} \cdots \int_{n_{\mathtt{p}}} \left(\boldsymbol{R}(\boldsymbol{U}^{n-1}_{\mathtt{PGD}}) - \sum_{k=1}^{n_{\mathtt{k}}} \left[\prod_{i=1}^{n_{\mathtt{p}}} B_i^k G_i \right] \boldsymbol{K}^k \boldsymbol{u} \right) \prod_{i=1}^{n_{\mathtt{p}}} G_i \; d\mu_{n_{\mathtt{p}}} \cdots d\mu_2 d\mu_1 = \boldsymbol{0}.$$

Replacing with the expressions of $\boldsymbol{K}(\boldsymbol{\mu})$, $\boldsymbol{F}(\boldsymbol{\mu})$ and $\boldsymbol{U}(\boldsymbol{\mu})$ given in 3.5, 3.6 and 3.4 in the definition of the residual in 3.1 it is obtained

$$\left[\sum_{k=1}^{n_{\mathtt{k}}} \boldsymbol{K}^k \underbrace{\prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} B_i^k (G_i)^2 \; d\mu_i}_{c^k} \right] \boldsymbol{u} = \sum_{j=1}^{n_{\mathtt{f}}} \boldsymbol{f}^j \overbrace{\prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} S_i^j G_i \; d\mu_i}^{\hat{c}^j} - \sum_{m=1}^{n-1} \left[\sum_{k=1}^{n_{\mathtt{k}}} \boldsymbol{K}^k \underbrace{\prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} B_i^k G_i^m G_i \; d\mu_i}_{c^{k,m}} \right] \boldsymbol{u}^m, \quad (3.15)$$

which is a linear system of equations for $\boldsymbol{u}$. Written in the short form:

$$\left[\sum_{k=1}^{n_{\mathtt{k}}} \boldsymbol{K}^k c^k\right] \boldsymbol{u} = \sum_{j=1}^{n_{\mathtt{f}}} \boldsymbol{f}^j \hat{c}^j - \sum_{m=1}^{n-1} \left[\sum_{k=1}^{n_{\mathtt{k}}} \boldsymbol{K}^k c^{k,m}\right] \boldsymbol{u}^m \tag{3.16}$$

with the previously defined scalars

$$c^k := \prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} B_i^k (G_i)^2 \; d\mu_i, \tag{3.17}$$

$$\hat{c}^j := \prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} S_i^j G_i \; d\mu_i, \tag{3.18}$$

$$c^{k,m} := \prod_{i=1}^{n_{\mathtt{p}}} \int_{I_i} B_i^k G_i^m G_i \; d\mu_i. \tag{3.19}$$

Once $\boldsymbol{u}$ is obtained, the subsequent stages are to obtain the values of G_i for $i = 1, 2, \ldots, n_{\mathtt{p}}$, again setting all the parameters as known, except for the G_i being determined. For a given i, this renders again the values of $\delta\boldsymbol{u}$ and $\delta G_{j\neq i}$ as zero, so the test function becomes

$$\delta \boldsymbol{U} = \boldsymbol{u} \delta G_i \prod_{j=1, j\neq i}^{n_{\mathtt{p}}} G_j.$$

Repeating the process as before, considering the equation 3.2 for $\boldsymbol{U}^n_{\mathtt{PGD}}$ it is obtained

$$\int_{I_i} \delta G_i \left[\int_{I_1}\int_{I_2}\cdots\int_{n_{\mathtt{p}}} \boldsymbol{u}^T \boldsymbol{R}(\boldsymbol{U}^n_{\mathtt{PGD}}) \prod_{j\neq i}^{n_{\mathtt{p}}} G_j \; d\mu_j \; d\mu_{n_{\mathtt{p}}} \cdots d\mu_2 d\mu_1\right] \; d\mu_i = 0. \tag{3.20}$$

In the previous equation, the series of $\boldsymbol{\mu}$ terms consider all spaces except the one corresponding to $I_i = I_j$. The previous relation also holds for all $\delta G_i \in \mathcal{L}_2(I_i)$. Equation 3.20 solves for the unknown function G_i and appears explicitly in 3.2, so the equation can be rewritten as

$$\int_{I_1}\int_{I_2}\cdots\int_{n_{\mathtt{p}}} \boldsymbol{u}^T \left[\boldsymbol{R}(\boldsymbol{U}^{n-1}_{\mathtt{PGD}}) - G_i \boldsymbol{K}(\boldsymbol{\mu}) \boldsymbol{u} \prod_{j\neq i} G_j\right] \prod_{j\neq i} G_j \; d\mu_{n_{\mathtt{p}}} \cdots d\mu_2 d\mu_1 = 0,$$

again, integrating all spaces except for $i = j$. This is also written as

$$G_i \left[\int_{I_1} \int_{I_2} \cdots \int_{n_\mathtt{p}} \boldsymbol{u}^T \boldsymbol{K}(\boldsymbol{\mu}) \boldsymbol{u} \prod_{j \neq i} (G_j)^2 \; d\mu_{n_\mathtt{p}} \cdots d\mu_2 d\mu_1 \right] = \int_{I_1} \int_{I_2} \cdots \int_{n_\mathtt{p}} \boldsymbol{u}^T \boldsymbol{R}(\boldsymbol{U}_{\mathtt{PGD}}^{n-1}) \prod_{j \neq i} G_j \; d\mu_{n_\mathtt{p}} \cdots d\mu_2 d\mu_1, \quad (3.21)$$

following the rule of $i = j$ as before. The terms in both left and right sides of 3.21 are computable functions in I_i. Using again the equations 3.5, 3.6 and 3.4 the expression can be written as

$$G_i = \frac{\sum_{j=1}^{n_\mathtt{f}} (\boldsymbol{u}^T \boldsymbol{f}^j) \hat{d}_i^j - \sum_{m=1}^{n-1} \sum_{k=1}^{n_\mathtt{k}} (\boldsymbol{u}^T \boldsymbol{K}^k \boldsymbol{u}^m) d_i^{k,m}}{\sum_{k=1}^{n_\mathtt{k}} (\boldsymbol{u}^T \boldsymbol{K}^k \boldsymbol{u}) d_i^k}, \quad (3.22)$$

where the terms d_i^k, $\hat{d}_i^j$ and $d_i^{k,m}$ are defined by

$$d_i^k := \left(\prod_{j \neq i = 1}^{n_\mathtt{p}} \int_{I_j} B_j^k (G_j)^2 \; d\mu_j \right) B_i^k, \quad (3.23)$$

$$\hat{d}_i^j := \left(\prod_{l \neq i = 1}^{n_\mathtt{p}} \int_{I_l} S_l^j (G_l)^2 \; d\mu_l \right) S_i^j, \quad (3.24)$$

$$d_i^{k,m} := \left(\prod_{j \neq i = 1}^{n_\mathtt{p}} \int_{I_j} B_j^k G_j^m G_j \; d\mu_j \right) B_i^k G_i^m. \quad (3.25)$$

For each value of n in the PGD solution, the alternated directions scheme is expected to converge to the optimal rank-1 approximation of $\boldsymbol{U}_{\mathtt{PGD}}^n - \boldsymbol{U}_{\mathtt{PGD}}^{n-1}$, with $\boldsymbol{U}_{\mathtt{PGD}}^n$ being the unknown term. Since the algorithm is iterative, a stopping criterion is defined in order to check if the solution is accurate enough. In this case, the selected criterion is based on the difference between steps, meaning that the algorithm should stop if the difference between solutions for $\boldsymbol{U}_{\mathtt{PGD}}^n$ and $\boldsymbol{U}_{\mathtt{PGD}}^{n-1}$ is small enough. If a solution for a previous step is considered with $\boldsymbol{u}$ and G_i for $i = 1, 2, \ldots, n_\mathtt{p}$, and also a solution for the newer step with $\boldsymbol{u}^{\mathtt{new}}$ and $G_i^{\mathtt{new}}$ for $i = 1, 2, \ldots, n_\mathtt{p}$, then the stopping criterion can be expressed considering the norm of both solutions, namely

$$\left|\left| \boldsymbol{u}^{\mathtt{new}} \prod_{i=1}^{n_\mathtt{p}} G_i^{\mathtt{new}} - \boldsymbol{u} \prod_{i=1}^{n_\mathtt{p}} G_i \right|\right|_{\mathtt{glob}} < \eta_{\mathtt{tol}} \left|\left| \boldsymbol{u}^{\mathtt{new}} \prod_{i=1}^{n_\mathtt{p}} G_i^{\mathtt{new}} \right|\right|_{\mathtt{glob}}, \quad (3.26)$$

with $\eta_{\mathtt{tol}}$ being a prescribed tolerance. This expression can be simplified using the normalized modes, namely $\tilde{\boldsymbol{u}}$ and $\tilde{G}_i$, including the amplitude β, such as

$$\left|\left|\boldsymbol{u}^{\texttt{new}}\prod_{i=1}^{n_\texttt{p}}G_i^{\texttt{new}}-\boldsymbol{u}\prod_{i=1}^{n_\texttt{p}}G_i\right|\right|^2_{\texttt{glob}}= \left|\left|\boldsymbol{u}^{\texttt{new}}\prod_{i=1}^{n_\texttt{p}}G_i^{\texttt{new}}\right|\right|^2_{\texttt{glob}}+\left|\left|\boldsymbol{u}\prod_{i=1}^{n_\texttt{p}}G_i\right|\right|^2_{\texttt{glob}}-2(\boldsymbol{u}^{\texttt{new}},\boldsymbol{u})\prod_{i=1}^{n_\texttt{p}}(G_i^{\texttt{new}},G_i) = (\beta^{\texttt{new}})^2+\beta^2 - 2\beta^{\texttt{new}}\beta(\tilde{\boldsymbol{u}}^T\boldsymbol{M}_u\tilde{\boldsymbol{u}}^{\texttt{new}})\prod_{i=1}^{n_\texttt{p}}\int_{I_i}\tilde{G}_i^{\texttt{new}}\tilde{G}_i\,d\mu_i, \tag{3.27}$$

where $(\cdot,\cdot)$ represents the scalar product. In this expression, when the new iteration is equivalent to the previous one, then the expression goes to zero due to the scalar product of identical normalized modes being equal to one.

3.1.5 PGD compression

After obtaining n terms in the PGD solution, there is a possibility of having redundant information, associated to the greedy strategy due to the fact that there is no orthogonality imposed between the successive computed modes. This means that the optimal number of required modes to have a solution with the desired accuracy may be less than n.

In this work, the employed solution to lower this effect is the so called PGD compression, which consists in a least-squares projection of the mode solution $\boldsymbol{U}^n_{\texttt{PGD}}$ into the same approximation space. This is computed with the same PGD strategy combining a greedy algorithm for the terms and an alternated direction scheme for the modes.

For the problem concerning this work, the compression is obtained by minimizing the least-squares function $\mathcal{J}$ defined as

$$\mathcal{J}(\boldsymbol{U}_{\texttt{comp}})=||\boldsymbol{U}_{\texttt{comp}}-\boldsymbol{U}^n_{\texttt{PGD}}||^2_{\texttt{glob}}, \tag{3.28}$$

where $\boldsymbol{U}_{\texttt{comp}}$ is the compressed solution of the problem. The aim is obtaining a separate approximation, $\boldsymbol{U}^{\hat{n}}_{\texttt{comp}}$ which contain $\hat{n}$ terms and is expressed in an similar way as the normal solution:

$$\boldsymbol{U}^{\hat{n}}_{\texttt{comp}}=\sum_{\hat{m}=1}^{\hat{n}}\hat{\boldsymbol{u}}^{\hat{m}}\prod_{i=1}^{n_\texttt{p}}\hat{G}_i^{\hat{m}}. \tag{3.29}$$

When using a PGD approach to minimize 3.29, the idea is to obtain an accurate enough approximation of $\boldsymbol{U}^{n}_{\texttt{PGD}}$ where $\hat{n} \ll n$. The first term of $\boldsymbol{U}^{\hat{n}}_{\texttt{comp}}$ given the PGD strategy is

$$\boldsymbol{U}_{\texttt{comp}} = \hat{\boldsymbol{u}} \prod_{i=1}^{n_{\texttt{p}}} \hat{G}_i.$$

The expansion in 3.27 is reduced to its dependence on the unknowns $\hat{\boldsymbol{u}}, \hat{G}_1, \ldots, \hat{G}_{n_{\texttt{p}}}$, as follows

$$\mathcal{J}(\hat{\boldsymbol{u}}, \hat{G}_1, \ldots, \hat{G}_{n_{\texttt{p}}}) = \left|\left| \hat{\boldsymbol{u}} \prod_{i=1}^{n_{\texttt{p}}} \hat{G}_i \right|\right|^2_{\texttt{glob}} + \left|\left| \sum_{m=1}^{n} \boldsymbol{u}^m \prod_{i=1}^{n_{\texttt{p}}} G_i^m \right|\right|^2_{\texttt{glob}} - 2 \sum_{m=1}^{n} (\hat{\boldsymbol{u}}, \boldsymbol{u}^m) \prod_{i=1}^{n_{\texttt{p}}} \left(\hat{G}_i, G_i^m \right),$$

where again an alternated direction scheme is used. In here it is assumed that for the first mode, the values of $\hat{G}_i$ are known for all $i = 1, 2, \ldots, n_{\texttt{p}}$. Since the functional is quadratic due to its least-squares nature, minimizing the problem leads to a linear solution for $\hat{\boldsymbol{u}}$. The dependence on the remaining unknowns for the functional is

$$\mathcal{J}(\hat{\boldsymbol{u}}) = \overbrace{(\hat{\boldsymbol{u}}^T \boldsymbol{M}_u \hat{\boldsymbol{u}}) \prod_{i=1}^{n_{\texttt{p}}} \left(\hat{G}_i, \hat{G}_i \right)}^{\lambda} + \left|\left| \sum_{m=1}^{n} \boldsymbol{u}^m \prod_{i=1}^{n_{\texttt{p}}} G_i^m \right|\right|^2_{\texttt{glob}} - 2 \sum_{m=1}^{n} (\hat{\boldsymbol{u}} \boldsymbol{M}_u \boldsymbol{u}^m) \underbrace{\prod_{i=1}^{n_{\texttt{p}}} \left(\hat{G}_i, G_i^m \right)}_{\gamma^m}.$$

Taking the derivative of the functional (i.e., $\frac{\partial \mathcal{J}(\hat{\boldsymbol{u}})}{\partial \hat{\boldsymbol{u}}} = 0$) it is obtained

$$\boldsymbol{M}_u \hat{\boldsymbol{u}} = \frac{1}{\lambda} \sum_{m=1}^{n} \gamma^m \boldsymbol{M}_u \boldsymbol{u}^m,$$

and since the mass matrix appears in both sides, it can be taken out. This leaves as a solution for the system the following expression

$$\hat{\boldsymbol{u}} = \frac{1}{\lambda} \sum_{m=1}^{n} \gamma^m \boldsymbol{u}^m. \tag{3.30}$$

The algorithm implemented in MATLAB to solve the problem, as well as the PGD compression process, are represented schematically in appendix B.

The solution to the problem, presented in a separated fashion, is given by equation 3.9. In this expression, the product from $i = 1$ to $n_{\mathtt{p}}$ is the multiplication of all the modal functions for a particular mode, while the sum from $m = 1$ to n represents the combination of all the modes obtained. This result can be expressed in a tabulated form, which is one of the advantages considered for this method. In this way, all the parts of the solution are presented in matrices of varying amounts of rows, with a number of columns equivalent to the number of computed modes. This can be written as

$$\begin{aligned}
\boldsymbol{\beta} &= \left[\beta^1 \ \beta^2 \ \cdots \ \beta^m\right], \\
\tilde{\boldsymbol{G}}_i &= \left[\tilde{\boldsymbol{G}}_i^1 \ \tilde{\boldsymbol{G}}_i^2 \ \cdots \ \tilde{\boldsymbol{G}}_i^m\right], \\
\tilde{\boldsymbol{u}} &= \left[\tilde{\boldsymbol{u}}^1 \ \tilde{\boldsymbol{u}}^2 \ \cdots \ \tilde{\boldsymbol{u}}^m\right].
\end{aligned}$$

In here, $\boldsymbol{\beta}$ is just a row vector of scalar values for each mode, while $\tilde{\boldsymbol{G}}_i$ is a matrix formed by column vectors $\tilde{\boldsymbol{G}}_i^m$, each one corresponding to a mode, with the length of the interval I_i defined for each parameter. The matrix $\tilde{\boldsymbol{u}}$ is a matrix of size $6N \times n_{modes}$ with the solution for each one of the modes computed in each column.

Given that the output is in a matricial form, its importation for post-processing can be done with a script in order to be interpreted by any software with array handling capabilities. In the next chapter, the web aplication tool that was designed for this work will be explained, since it takes advantages of this output style in order to display the real-time the results.

Chapter 4

Post-processing through web application

In the previous chapters it was discussed the advantages of the PGD method for computing solutions on tessellated lattices. Given the fact that this method provides tabulated solutions stored in a vademecum that can be accessed in a fast way, this enables in practice real-time solutions given the proper value of the parameters. This works as an inspiration to have a proper post-processing tool that allows the user to see the solutions for different changes in the parameters in an interactive, graphical way that could ease the understanding of the phenomena occurring in the given structure.

In this chapter a post-processing solution is proposed and explained, as well as providing different computed cases to demonstrate the capabilities of the tool as an aid for design and study of auxetic metamaterials. The post-process tool of this work is chosen to be developed in a web-based fashion. The main goal of this program is to show, in an interactive graphic interface, how auxetic materials behave under different settings. Web development offers a fast solution, which makes it appropriate for presenting the results. Having the tool implemented simplifies the way in that the problem is seen. A PGD problem gives copious amount of data as result, which is not ready to be analyzed until it is post-processed, which can be a tedious task if there is no way to display all the results at the same time. In the same manner, someone who is not used to work with the numerical method could not be able to interpret this results without studying the math behind it. In this sense, this tools points to those kind of users, since they can visualize how the material behaves given certain parameters, without the need of looking at background information. The tool also points to see results for specific designs. Any kind of solution in PGD form could be implemented to the graphic interface, which would allow users with particular set of solutions to implement their own in a similar way, not necessarily limited to auxetic materials.

The post-processing application is developed using the web *de facto* standards. A graphical interface is developed using a combination of HTML and

CSS languages, which provide operability in a user-friendly environment. This is powered with the scripting language JavaScript, which has been used for years as a tool to provide functionality and dynamic features in websites. The graphical part of the tool is handled through WebGL (Web Graphics Library), which is included virtually in all popular web browsers available. In order to use the WebGL API (Application Programming Interface), a JavaScript library called `Three.js` is used, which is designed to take advantage of the WebGL framework to provide 2D and 3D graphics without the need of adding third party add-ons.

The inception of the web application is a code, developed by the AMB group in the University of Zaragoza, for displaying the solutions of a cantilever beam obtained with the PGD method [21]. This original file allowed to watch the deformation of the beam being loaded in real-time, which inspired the work developed here.

The full code to be presented in this chapter can be found in [22], extensively commented in order to be understood, and readily available to be modified in case of need.

4.1 Structure of the application

The web application is designed in a typical software structure, with the webpage being the graphic interface. This structure is chosen in order to ease the development and maintenance of the software, separating critical elements from different aspects of the application into their corresponding folders, for modularized organization. In this section the parts of the folder structure of the project will be explained. The root folder of the project is:

```
|
|_ cases/
|_ css/
|_ docs/
|_ js/
|_ resources/
|
|_ index.html
```

In this structure it is intended that the `cases` folder be the only one modified, which is where new solutions can be added following the coding style explained in this chapter. The `css` and `js` folders act as the core of the application, while the `resources` and *docs* folders provide additional data and information about the application.

4.1.1 Basic interface

The interface of the project is defined mainly by two elements: `index.html`, which defines the elements to show on the interface, and the `css` folder, that contains the CSS (Cascade Style Sheets) file that define how the elements are arranged and displayed in the html. The main interface is shown in figure 4.1, and is divided in two parts:

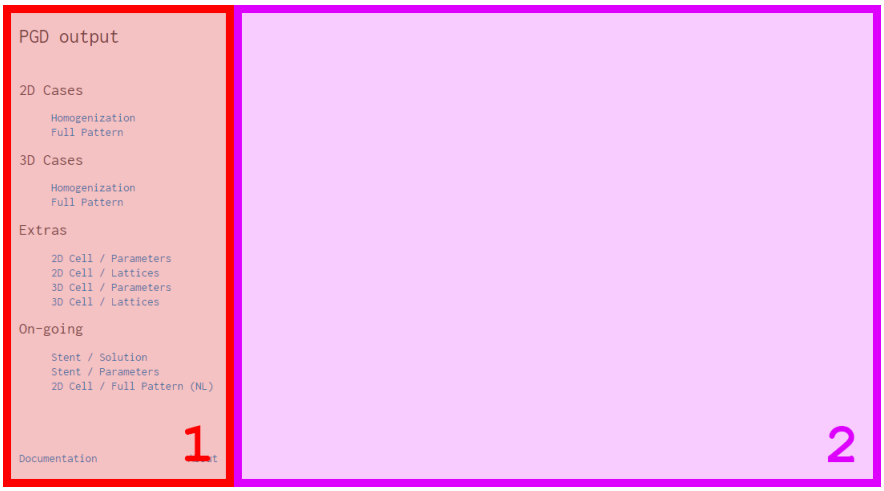


Figure 4.1: Interface and its constituents.

Panel 1. Side panel: Fixed on the screen, contains the different cases implemented in the code that can be shown. Clicking into any of the different solutions loads it into the main working area (*Screen 2*) for the user to interact with it. It is thought as the main menu and starting point to use the application.

Panel 2. Working area: It is defined in code as an iframe, which is an HTML element that allows embedded content in the interface. Clicking in the elements of the side panel "injects" the html code inside the iframe, instead of loading a web-page on its own. This was done in such way in order to reduce the amount of files needed and to simplify the implementation of future cases, since a generic interface is already implemented and loaded for each case, eliminating the need of coding a new interface for each new solution.

Once a case is selected in *Screen 1*, it is loaded into the interface, which is depicted in the following figure:

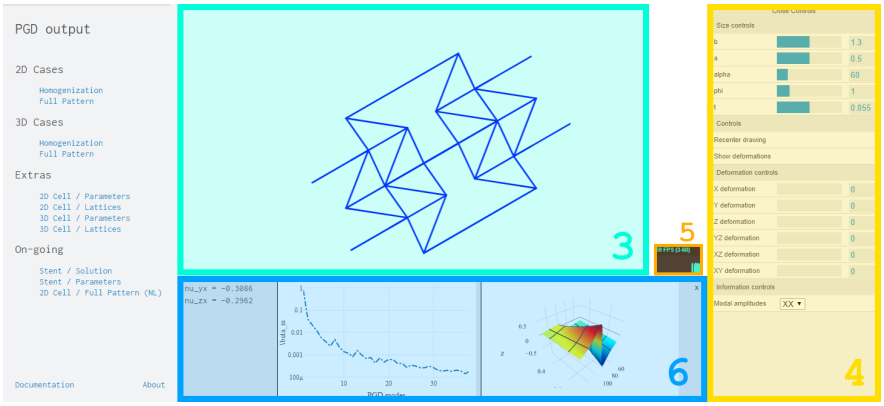


Figure 4.2: Main interface and its parts.

After a solution is loaded into *Screen 2*, the working area is filled with different elements that play different roles in the interaction. In this case, the panels are defined as:

Screen 3. Graphic result: The graphic solution of the structure is loaded in the middle of the working area. The screen is provided with functionality to visualize the solution using a mouse in computers or the fingers in a touch-screen capable device, allowing for rotation of the figure, as well as translation and zooming, enhancing the experience. The problem is now observable from different perspectives that simple image plotting restricts. *Screen 3* works in close relation to the control bar. Every change made in the parameters defined in *Screen 4* is visible immediately in here, allowing user interaction and visualization of different solutions obtained from PGD computation.

Screen 4. Control bar: One of the main elements of the web application, it enables the modification of parameters without the need of reloading the page or the need to input values, in order to provide interaction with the image in *Screen 3*. In figure 4.3 a detail of the control bar for the solution of a scaffold in 2D is shown as an example.

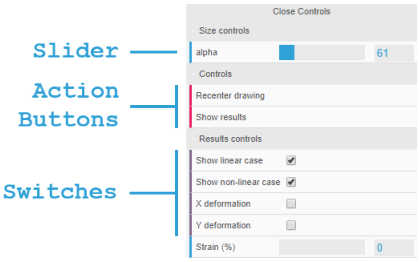


Figure 4.3: Detail of the control bar.

The interaction with the application is mainly done in this two panels. For every one of the implemented cases, the control bar shows the appropriate parameters that are used in the particular problem shown in *Screen 3*. Some possible actions that can be added are described:

- Sliders: They control the range between values in a certain amount of steps defined. In this project they model the ranges of the parameters, since it allows to display in an intuitive graphical fashion the values selected for the parameters. In the image it can be seen how it is being used to set the values of b, a, α and t to some desired set. It is also used to range the values of the deformation to show on the model in *Screen 3*.
- Switches: Allow the use of true/false statements. In our case, it is used to show or hide cases on the working screen, displaying only the deformations marked.
- Action buttons: Used to trigger actions. They are programmed for two things: recentering the image and show the results controls. The first one only renders the image again in the center of the working area, resetting the rotations or translations made. The latter one opens a new drawer in the control bar that contains elements to modify the solution, such as changing the deformation of the structure or enabling actions associated to them, like opening the information panel to see "live" numerical values.
- Inputs: Give the ability to add desired numerical information without being limited to ranges. It is used, for example, to create lattices, where the number of repetitions per axis is set arbitrarily.
- Lists: Allow the user select one case from a collection of the database, such as the different loading cases and which one is to be displayed in the screen.

Screen 5. Stats bar: A piece of code added to track the performance of the graphic engine. It displays in real-time the frames per second at which the image is moving, so it allows to see if the CPU/GPU usage is at high levels due to slowness of the renderization. It relies in a external library named `stats.js`, developed by the same group that mantains *Three.js*.

Screen 6. Results panel: Space reserved for showing live results of the image displayed in *Screen 3*. It is available to show results related to the simulation being displayed, and can accept any type of information that is written in text format. In the case of this work, the panel is used when displaying the solutions of the homogenized cell, as well for the results of the full pattern deformation. In both cases, *Screen 6* is used to display live values of the Poisson's ratio, computed in real-time given the parameters that are chosen in the control bar. Its functioning is triggered by the parameters, as it was explained, by the use of the "Show results" button, which opens the panel in the cases that is required. As an additional information, there are also two spaces reserved for plot display, where different information such as the modal amplitudes, the plots for the values of the orthotropic Poisson's ratios or the modal functions can be displayed, to complement the information given in the main screen.

4.1.2 Application core

The main functionality of this project is achieved through the use of code written in JavaScript, which is compatible with every modern web browser. The scripts are arranged according to the function that they provide in the project: The ones that have a global scope are arranged in the `js` folder, which provides the general functions that are used by all cases. Scripts that are only valid for a certain case are placed in the corresponding case folder, since the code related to them does not affect the functioning of the rest of the application.

The `js` folder has the next structure:

```
|_ js/
      |_ core/
      |_ misc/
```

`core` folder

The following files can be found here:

- `three.js`: Is the main file for the JavaScript library and API of the same name, which is used to display computer graphics using the capabilities of the WebGL library, which is handled by the browser. The library is published under the MIT license, which allows complete modification and redistribution free of charge as long as credit is given. The documentation of the library can be seen in [23]. The main uses of the library are to provide an easy definition of geometries and materials in order to define shapes and properties associated to them.

- `CanvasRenderer.js`: Module of `three.js` mainly designed for drawing 2D graphics using the *Canvas 2D Context* API. Its use can be extended to 3D graphics, with a slower implementation than the WebGL renderer. However, it circumvents some limitations that the WebGL renderer has, like displaying different line widths on Windows.

- `OrbitControls.js`: Module of `three.js` that enables the controlling of the space where the elements are displayed. In desktop computers allows the usage of the mouse and its buttons to control the images, using the left button for orbiting the image, the right one for panning, and the scrolling wheel for zooming. In the case of portable devices, defines the orbiting with one finger scrolling, the panning with three fingers and the zooming with two.

- `Projector.js`: Module of `three.js` that enables the projection of geometries in the space defined. It enables the visibility of the defined geometries and its handling during spacial transformations.

- `jQuery.js`: JavaScript library developed for handling of the HTML elements in an easy way. It is widely used since it simplifies the selection of

elements of the code, as well as handling events and animations, effectively reducing the amount of coding needed to obtain the same results. It is also distributed under the MIT license.

- `plotly.min.js`: JavaScript library made for easy implementation of plots. It is compatible with different frameworks, allowing a simple adaptation from MATLAB files. It works by taking data and instructions, externalizing the of coding the vector graphics associated with the plots.

- `math.min.js`: Library developed for implementing basic mathematical capacities to JavaScript. The language can natively handle basic instructions like sum or division, but lacks the format to work with higher entities like vectors or matrices. The library extends the array system, giving the ability to use linear algebra and efficient indexing.

- `stats.js`: Library that handles the statistics bar shown in *Screen 5*.

- `dat.gui.min.js`: Graphical user interface developed for changing variables in JavaScript using a visual representation instead of having to modify the variables in code or as an input. It is used in combination with the tabulated results obtained from the PGD computations in order to have the real-time solutions in an interactive way. This is achieved using the following process:

  - Modification of the parameters in the control bar. This changes the value of the variable associated to them, which is then saved.
  - When the parameters are modified, a function is called to refresh the image being shown. When this happens, the new value of the variable is used to access the different arrays coming from the PGD solution, with the values associated to the chosen parameter.
  - Having this values updated, they are used to recompute everything dependent on them, such as coordinate matrices or deformations.

  Examples of how the values are tabulated and addressed in the application are shown in section 4.1.3.

- `loader.js`: Set of functions provided to load the cases into the interface. As it was explained before, the cases are not a webpage on itself, but content that is "injected" into the iframe element, which displays the loaded case. The following functions are implemented in here:

  - `getLinkbyId()`: Each link in the side panel is defined with a unique ID, which is also stored in a JSON file[1]. When the links are clicked, this function is called and injects the basic html structure into the

[1]JSON, or JavaScript Object Notation, is a file format that allows to parse data objects through the use of tabulated attribute-value pairs, including the use of arrays. Its use is extended due to the fact that many programming language include tools to parse or generate JSON formatted data natively.

iframe, as well as loading the functions needed for the case to work. The files that are needed are defined in the JSON file under the unique ID, so each case may load different files.

- `getDocs()`: Function designed to load the documentation on top of the site. It is defined separately since the documentation is defined as a separate website, which is loaded and superimposed in the same space dedicated to the case. The documentation presented include the points explained in this work plus more particular features, in case the user needs to include or modify the code in it.

`misc` folder

Three libraries are included:

- `anim.js`: One of the most important library of the project, it contains two functions:
  - `init()`: It is the function that initiates the rendering of the solution. It works by defining the main elements that *three.js* requires to work:
    - The container: Defines the physical part of the screen where the animation will be held. It is the main HTML element and works as the link between the HTML code and the JavaScript code used for programming the graphical elements.
    - The renderer: The object in charge of transforming the programmed orders in JavaScript into visible images on the website. It is nested inside the container, generating the *scene*.
    - The scene: Represents the image held inside the container, and it holds inside the different geometries that will create the view.
    - The control bar: *Screen 4* as defined in section 4.1.1.
    - The statistics bar: *Screen 5* as defined in section 4.1.1.

    This function also defines the initial coordinates of the element to be displayed as well as the position of the camera (the view to the image) and its controls. Since the cases can be very different from each other, in order for the `init()` function to work, the cases must be defined in a standardized way in order to be compatible. This is explained with more detail in section 4.2.3.
  - `animate()`: The animation of the elements being shown on the screen is performed by an infinite background loop that provides a new image (or frame) 60 times per second, in order to create a fluid image. This function is repeated, requesting the renderer to provide a new image, given the parameters that are selected by the user at the moment.
- `events.js`: Contains the different events that are not essential for the display of the solution, but provide extended functionality to them. It is defined by six functions:

  - `onWindowResize()`: Readjusts the size of the image when the browser window is resized, in order to keep the proportions and the image on focus, without cuts or alterations.
  - `panelCreate()/panelOpen()/panelClose()/panelLoad(string)`: Functions defined to create the interaction with the information/results panel defined as *Screen 6* in section 4.2.1. As the names suggests, `panelCreate()` creates the physical panel where the information is to be shown, `panelOpen()` is used to open and display the panel, and is typically used together with a button in the control bar; `panelClose()` is defined to erase the panel from the view and works together with the "x" button on the top right corner. Finally, `panelLoad(string)` loads the information passed through the variable `string` into the panel.
  - `recenter()`: A function added for easiness of display. Its function is to recalculate the coordinates in order to display the image again in the center. This cuts the need of reloading the image to obtained the original view.

- `utils.js`: A list of utilities to ease some of the calculations performed by the program, such as the length of the bars drawn, creation of linearly spaced arrays, among others.

4.1.3 Case system

After presenting the main functionality of the web application, it is necessary to explain the case system implemented in order to define the solutions, on how they depend and interact with the core to display the graphics. Each case is represented by a unique name ID that is used by the core functions to load them into the application. For ease of explanation in this work, an explanatory fictitious case will be used by the generic name of `caseName`.

Main case file

The main file for developing each case is `caseName.js`. In this file, variables and functions defined for each one of the particular cases are declared. The variables in this file need to be named arbitrarily in order to be compatible with the core of the application. Therefore, some restrictions need to be applied for variables such as coordinates, connectivities and parameter definitions, such as deformations or other logical variables. The functions defined in this file complement the ones already presented in the core, and they are used specifically for purposes of the specific case. The contents of this file are:

- `createGUI()`: Function that defines how the control bar will be displayed in the work area. In here, the parameters and their ranges, as well as the sections displayed on the control bar, are defined. All the functionality that is given to buttons and boxes shown are also coded here.

- `result()`: Used for all the modifications related to the display of solutions in the cases. It adds elements to the control bar, loads the information panel and also performs the calculations in the code to display the results in it. This includes real-time computation of the effective Poisson's coefficient values that are displayed then on the panel, as well as their plots.
- `recreate()`: Designed for adapting the structure to the particular set of parameters chosen and modified on the control bar. When the values are changed, this function is called and performs the computation of the nodal coordinates with the new values selected, loading the new image into the screen.

Parameters files

Parameters files are typically the ones used as input data coming from the solution of the problem, and are generated by the software used to solve the problem, in this case MATLAB. These files need to be also defined with a `.js` extension. Some of the files that can be added are the separated solutions of the PGD problem, matrices, as well as any other constant that will be used for defining the problem. Precaution should be taken at the moment of defining this, since JavaScript does not handle cell structures or matrices as MATLAB does. One way to circumvent this issue is to define multidimensional arrays, which can store the information from the MATLAB cells, but may have some complicated writing and could lead to errors at the moment of using them in the functions. Using this format, separable matrices are expressed as

```
M[k][i][j] = [[[M_11^1, M_12^1, ..., M_1n^1],
                              ...
               [M_n1^1, M_n2^1, ..., M_nn^1]],
                              ...
                              ...
              [[M_11^m, M_12^m, ..., M_1n^m],
                              ...
               [M_n1^m, M_n2^m, ..., M_nn^m]]]
```

where i and j account for the subindices of the matrix, while k for the different modes from 1 to m, giving an array of size $m \times n_{rows} \times n_{cols}$. Using the same convention, vectors are represented as

```
v[k][i] = [[v_1^1, v_2^1, ..., v_n^1],
                     ...
           [v_1^m, v_2^m, ..., v_n^m]]
```

Again, with i being the vector element and k being the corresponding mode, giving a nested array of dimension $m \times n_{elems}$. In the case of this work, when it is required, the stiffness matrix is added in its own file in the shown format. The solution from the PGD method is uploaded separately for each one of the load

cases defined in section 2.4. An excerpt of one of this files is shown below:

```
uxx = [[[0.000000, 0.000000, 0.000000, 0.000000, -0.000000, ...

bxx = [[1.140523,1.151271,1.162011,1.172742,1.183466, ...

axx = [[1.520048,1.527351,1.533917,1.539823,1.545138, ...

alxx = [[0.072721,0.073384,0.074055,0.074735,0.075423, ...

phixx = [[0.861956,0.866231,0.870030,0.873402,0.876393, ...

txx = [[3.344676,3.344653,3.344624,3.344587,3.344541, ...

cxx = [0.780098,0.038787,0.021473,0.011464,0.005369, ...
```

In here, `uxx` is the deformation of the problem in a separated matricial form (of size $n_{\texttt{modes}} \times n_{\texttt{nodes}} \times n_{\texttt{dof}}$), `bxx` is the modal function associated to the parameter b, and so on for each one of the five parameters that define the problem. The last array, `cxx`, corresponds to the modal amplitudes.

Mesh file

The mesh file is where the coordinates and connectivities of the structure are defined, in order to be loaded. It is required both by the core and local problem functions, so it is necessary to define a name convention for the variables and functions defined in here to be compatible with the rest of the application. The core variables that are defined for using together with the JavaScript libraries are

- `NUM_NODES`: number of nodes of the structure.
- `NUM_ELEMS`: number of elements of the structure.
- `DIM3`: boolean variable to define if the problem is 2D or 3D.
- `CONNECTIVITY`: connectivity matrix, defined as a 2D array of `NUM_NODES` $\times$ 2 elements.
- `COORDINATES`: coordinates matrix , defined as a 2D array of `NUM_NODES` $\times n_{\texttt{ndim}}$ elements, with $n_{\texttt{ndim}}$ being the dimensionality of the problem (2D or 3D). If the problem is defined in a separated fashion, the matrix is defined as a 3D array of $N \times$ `NUM_NODES` $\times n_{\texttt{ndim}}$, where N is the amount of matrices in which the full matrix is separated.

If the coordinates matrix is defined in a separated fashion, a loop needs to be defined in the file in order to assembly the matrix with the initial set of parameters. It is not required, but recommended, to define the parameters in this file, with the separated arrays and the initial values in order to create the proper

coordinates matrices for each problem.

The idea of this file is to take out information from the main case file, in order to keep it clean and maintainable, so any variable or data required by the functions defined earlier should be defined in here. Both files need to be coded side by side, since the main case file requires the data from the mesh files, so careful attention should be put in the usage of names. In most of the cases, the file looks like the following:

```
var NUM_NODES = 26;
var NUM_ELEMS = 37;

var COORDINATES1 = [[-1.00, 0.00, 0.00],[-0.50, 0.00, 0.00], ...
var COORDINATES2 = [[1.00, 0.00, 0.00],[1.00, 0.00, 0.00], ...
var COORDINATES3 = [[0.00, 0.00, 0.00],[0.00, 0.00, 0.00], ...
var COORDINATES4 = [[0.00, 0.00, 1.00],[0.00, 0.00, 1.00], ...
var CONNECTIVITY = [[1, 2],[2, 3],[3, 4],[5, 4],[5, 6], ...

var b_pm = 25, a_pm = 25, phi_pm = 11, t_pm = 0, al_pm = 16,
   nu_pm = 51; // Initial indices of the arrays

var a_arr   = new Array(51); // Initialization of the arrays
...

// Population of parameters array
for (var i = 0; i < one_dim.length; i++) {
    a_arr[i] = 0.3  + i*(0.7-0.3)/(51-1);
    ...
}

// Population of parameters array
for (var i = 0; i < one_ang.length; i++) {
    sal_arr[i] = Math.sin(0.25*Math.PI + 0.5*i*Math.PI/(91-1));
    ...
}

// Definition of initial coordinates
for (var i = 0; i < CoorSup.length; i++) {
    CoorSup[i] = new Array(3);

    for (var j = 0; j < CoorSup[i].length; j++) {
        CoorSup[i][j] = CoorSup1[i][j]*b_arr[b_pm] + ...
    }
}

// Slider control (defines controller object properties)
var slider = {
    b:     b_arr[b_pm],
    a:     a_arr[a_pm],
    alpha: al_arr[al_pm],
    ...
};
```

The definition of the amount of nodes and elements per cell is stated in the beginning of the file. Following, the coordinates arrays are written. In here the nodal positions were separated in four matrices, which are added independently. The connectivity matrix is also added as a simple bidimensional array. Line 10 is the initialization of the indices for the parameters arrays (giving the initial position to display). After this, the arrays for the parameters are initialized, while the next `for` loops correspond to the writing of the ranges in the arrays. The last `for` loop defines the initial position matrix, taking both the indices defined and the parameter ranges.

One important element that also needs to be defined is the "control bar object". The elements displayed in the control bar writes and reads information from this structure, which is then used by other functions in order to process instructions related to the solution. For example, according to the case shown in figure 4.3, the object looks like below:

```
1  var slider = {
2      b:      b_arr[b_pm],
3      a:      a_arr[a_pm],
4      alpha:  al_arr[al_pm],
5      t:      t_arr[t_pm],
6      scale:  0,
7      xx:     false,
8      yy:     false,
9      rectr:  function() {recenter()},
10     reslt:  function() {result()},
11 };
```

The values displayed are those shown when the case is loaded into the screen. For the buttons that provide actions, a function is set as attribute, so it can be run when it is clicked.

4.2 Post-process

As it can be seen from the previous sections, full solutions can be developed when implemented following the mentioned procedures. The usage of the control bar, plus the iteration loop of the animation, provide the real-time capabilities that were looked for the PGD solutions, which is useful for doing the post-processing. In PGD, results are given separated by modes, each one represented by matrices and vectors that account for a part of the solution, but not for the whole. In order to see what is happening to the structure, combinations using the adequate parameters need to be performed, which can be tedious to verify in MATLAB, since it needs to be tested case by case. When the results are imported into the application, all the solutions are available, since the modification of the parameters is done on the screen, with the combinations of elements performed in the background, and thus giving solutions that can be readily seen (figure 4.4).

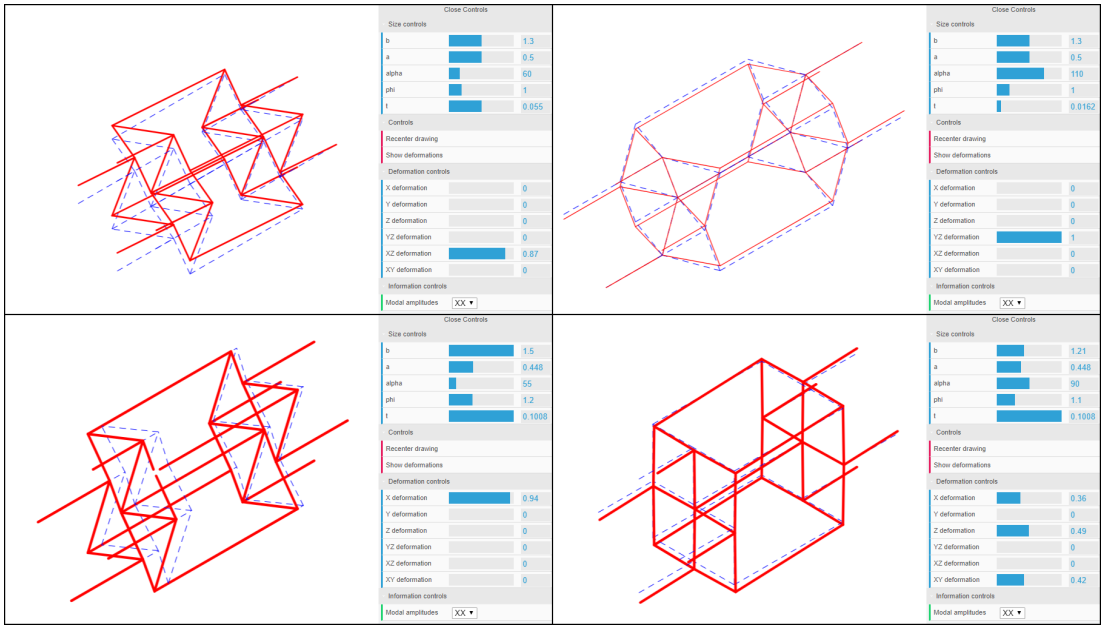


Figure 4.4: Example of a deformation solution in different parameters.

The ability to control the position of the camera in the model also helps in the post-process, since it allows to observe the problem from different angles. This gives the user the chance to see the effects of the different properties in a way that may have not been visible before from the static images.

Chapter 5

Numerical solutions

In previous chapters, the methodology for solving the problems using the PGD method was established. Also, a post-processing tool was developed in order to visualize in real-time the results, using the vademecum obtained in the solution. In this chapter, the results for different simulations are presented and explained. The implementation was done using MATLAB routines, taking advantage of its matrix handling capabilities.

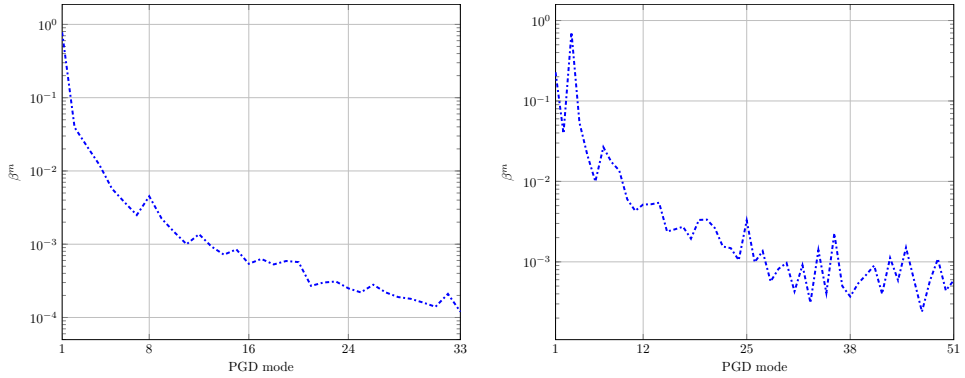
5.1 Homogenized cell

The results in this section were calculated using the methodology explained in 2.4.1. For the computation, a prescribed strain is applied according to the homogenization theory, as depicted in figure 2.7. The cases are run using the PGD solver defined in chapter 3 and a separated solution is obtained. A tolerance of 10^{-6} is set as a stopping criteria for the alternated directions loop, which is measured as a ratio between two consecutive modal amplitudes. Additionally, a maximum of 51 modes is set, in order to limit the amount of computation up to a relevant number of terms.

5.1.1 Modal amplitudes

The cases are run both with PGD compression and without, to compare the effects of reducing the amount of modes while keeping the solution accuracy. With the values obtained, the amplitudes are shown below. These correspond to the contribution of each one of the modes to the total solution. The first loading state, ε^{XX}, can be seen in figure 5.1. In here, the effects of the modal compression are evident. The computation of modes without compression is stopped for the maximum selected number of terms, which is set as 51 modes, while the compressed solution reduces this to 33. This makes the solution smaller, making memory allocation less demanding. It is also noticeable that the tendency of the data is similar for both cases, since the first modes provide values of β of the order of 10^{-1}, decreasing until values of 10^{-3} in the last ones. Compression achieves

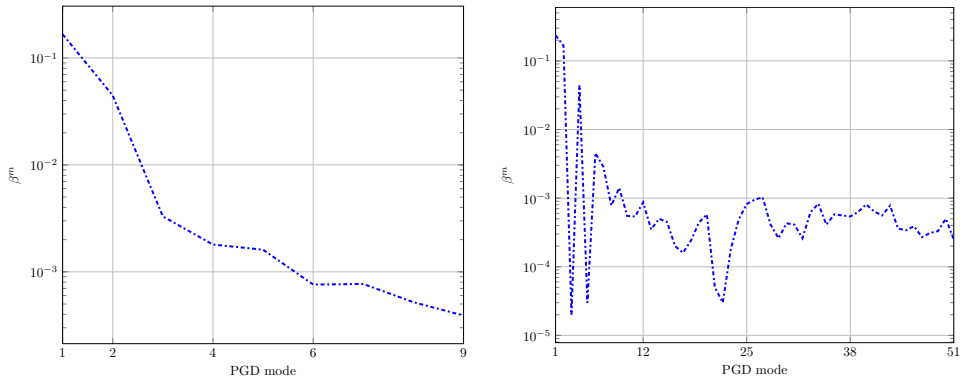
this in less modes, reducing as well the oscillation effect on modal amplitudes.



(a) PGD compression. **(b)** No PGD compression.

Figure 5.1: Modal amplitudes for XX case.

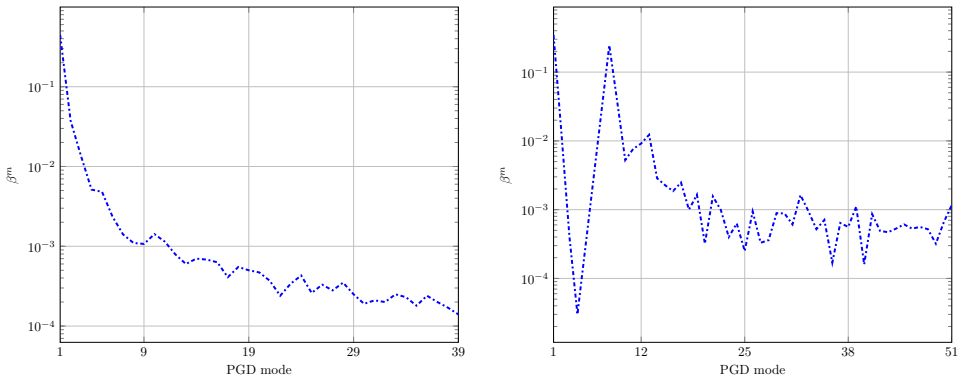
The same results are shown for the cases YY, ZZ, YZ, XZ and XY respectively in figures 5.2 to 5.6.



(a) PGD compression. **(b)** No PGD compression.

Figure 5.2: Modal amplitudes for YY case.

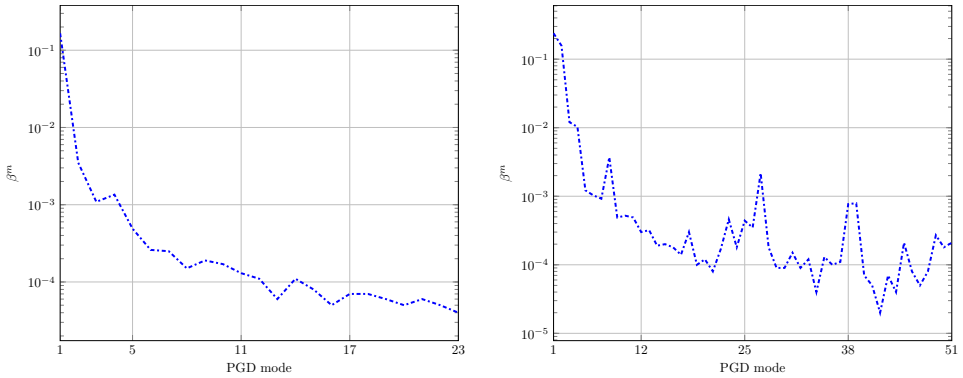
For the case YY it is seen that the compression reduces effectively the amount of modes from 51 to just 9. This amount of modes is low, and it can be explained by the fact that they are enough to represent the exact solution to an acceptable degree. As it will be seen later in the error analysis, case YY is the case with the smallest deviation when compared to the solution obtained by FEM. The oscillation of the results in the case without compression is big, alternating then between modes of very low to very high values of the amplitudes. However, when launching the PGD solver with compression, this one builds up a solution by adding modes that have a bigger contribution to the solution, thus explaining the big improvement between both compressed and uncompressed case.



(a) PGD compression. **(b)** No PGD compression.

Figure 5.3: Modal amplitudes for ZZ case.

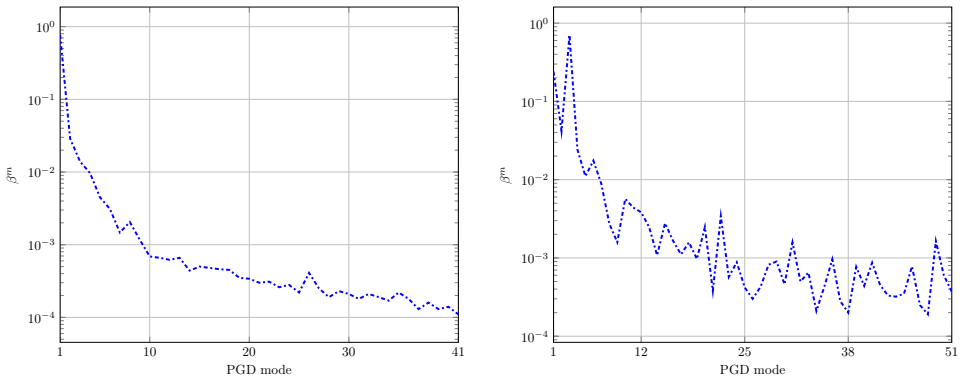
Case ZZ helps clarify the point seen in figure 5.2. It can be seen that convergence for the compressed modes is straightforward, while the solution without the compression tends to oscillate. This oscillations disappear in the compressed solution because of the fact that modes adding less relevant information can be recalculated in fewer modes with a higher contribution.



(a) PGD compression. **(b)** No PGD compression.

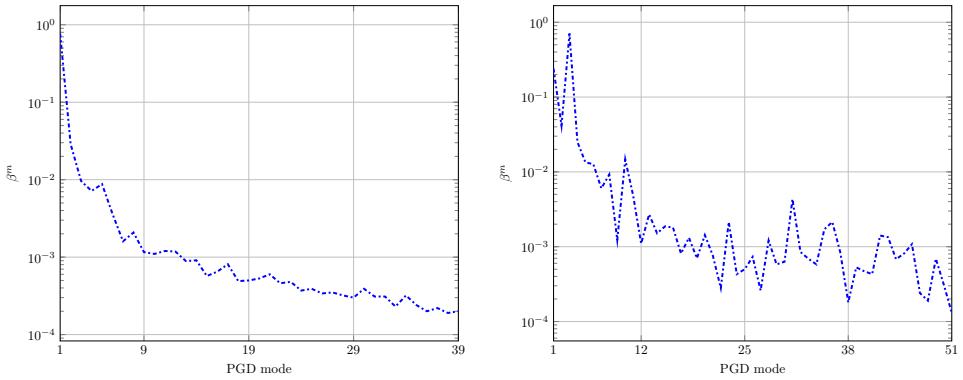
Figure 5.4: Modal amplitudes for YZ case.

In case YZ it can be seen that the tendency for both computations is similar, and the amplitudes follow the same trend as the other cases. Compressed solution can handle a solution with fewer modes, but the accuracy is mostly kept equal.



(a) PGD compression. **(b)** No PGD compression.

Figure 5.5: Modal amplitudes for XZ case.



(a) PGD compression. **(b)** No PGD compression.

Figure 5.6: Modal amplitudes for XY case.

For the last two cases the phenomena from previous cases is repeated. Oscillations are again present, but with similar tendencies for both cases, so compression works as smoothness for the curve while reducing the total amount of solution modes.

As can be seen as a general comment for all cases is that the new modes being added keep contributing to a total solution, but as a general trend they do it in a decreasing fashion, being less relevant. Total solution for both compressed and uncompressed cases can be compared in accuracy, but it is preferable the cases with compression, due to the smoothness of the solution, achieving similar results with less demand for memory.

5.1.2 Unit cell deformation and modal functions

The deformations for each of the prescribed strains are shown below. For every case, the deformation of the unit cell is shown, as well as three representative

deformations from three modes of the solution, chosen arbitrarily to display how modal solutions work. For the modal functions, the first six modes are shown in order to establish a trend for each one of the cases. Since PGD provides a set of solutions, in order to display the deformations, an arbitrary set of parameters is chosen, namely:

$$b = 1.25, \quad a = 0.50, \qquad \alpha = 60°,$$
$$\Phi = 1.00, \quad t = 0.0055.$$

For case XX:

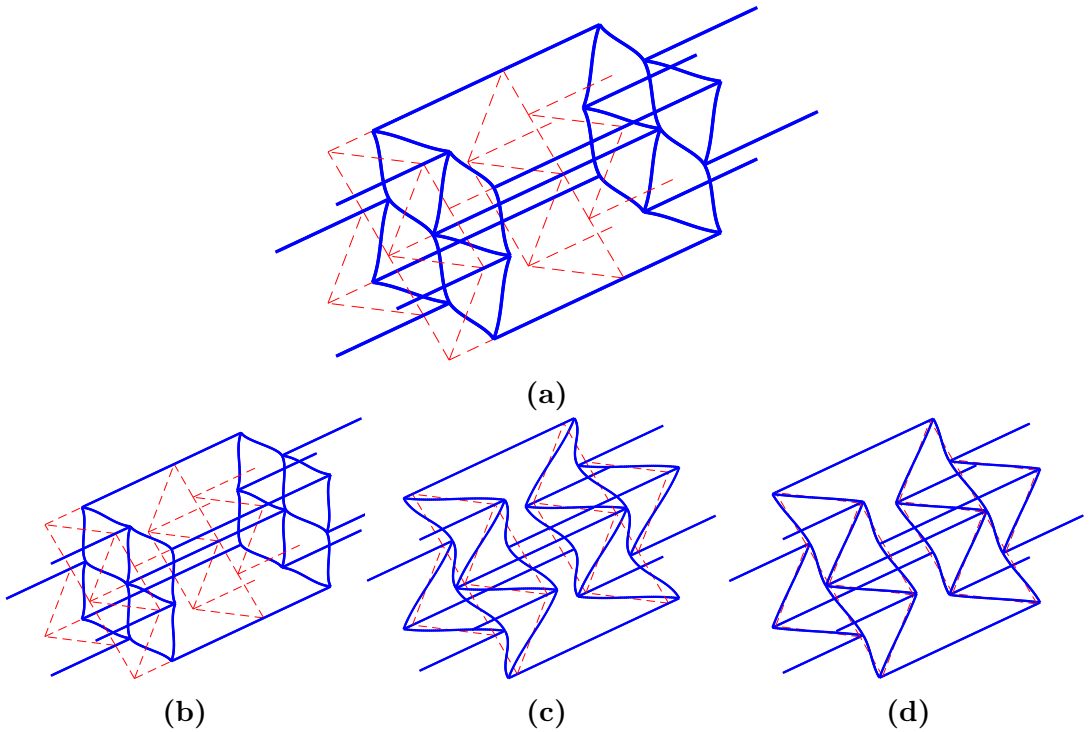


Figure 5.7: Modal deformations for case XX (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 5 (scale 10x).

Here, it can be noticed that the first mode represents the biggest contribution to the total deformation of the cell. The deformation of mode 1, compared to the total sum is similar, but it can be appreciated that the corners of the cell are not displaying similar movement. This is due to the way that the solution is calculated. In here, the first mode is not able to capture all the information of the total solution, but the sum of all of the will, up to a certain tolerance. In this case, the movement of the corners of the cell is corrected by the successive modes, as is shown for example in the deformation of the second mode. This is why having a relevant amount of modes is necessary to obtain a trusted solution, and also why the tolerance is set by how relevant are the new modes being computed.

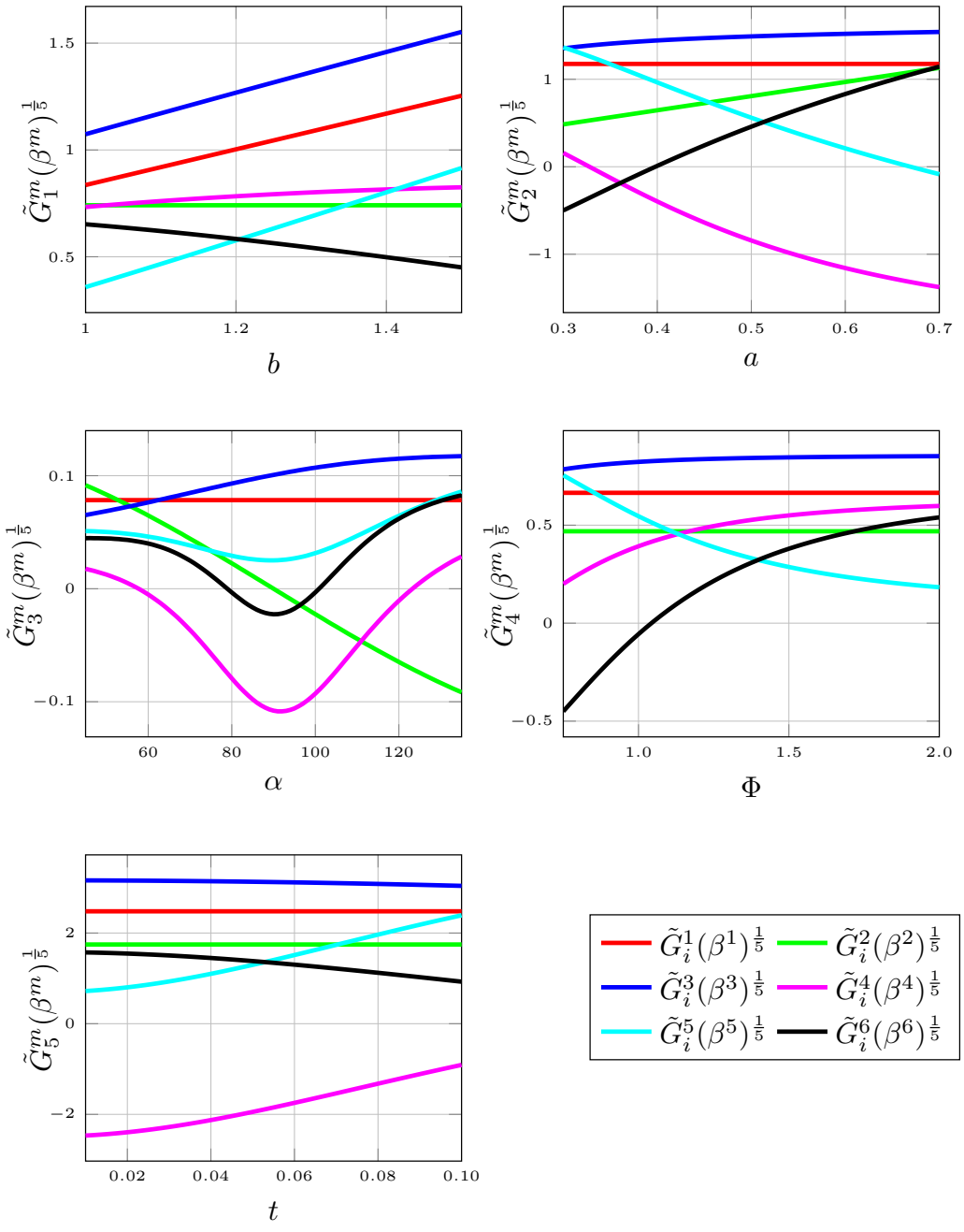


Figure 5.8: Modal functions for case XX.

In figure 5.8 the normalized functions, weighted by the modal amplitudes $(\beta^m)^{1/n_p}$, with $n_p = 5$, are plotted for the first six modes of case XX. Here, the effects of the different parameters become apparent. For the solution, every parameter has some sort of increasing or decreasing relation with the variation of the interval of the parameter (i.e., the derivative keeps its sign during the interval). However, this behavior is not present in α, which for certain modes display a change of behavior in the value of 90°, since this value represents the point in which the unit cell stops behaving as an auxetic material, given on how the shape is defined.

Repeating the exercise for the case YY:

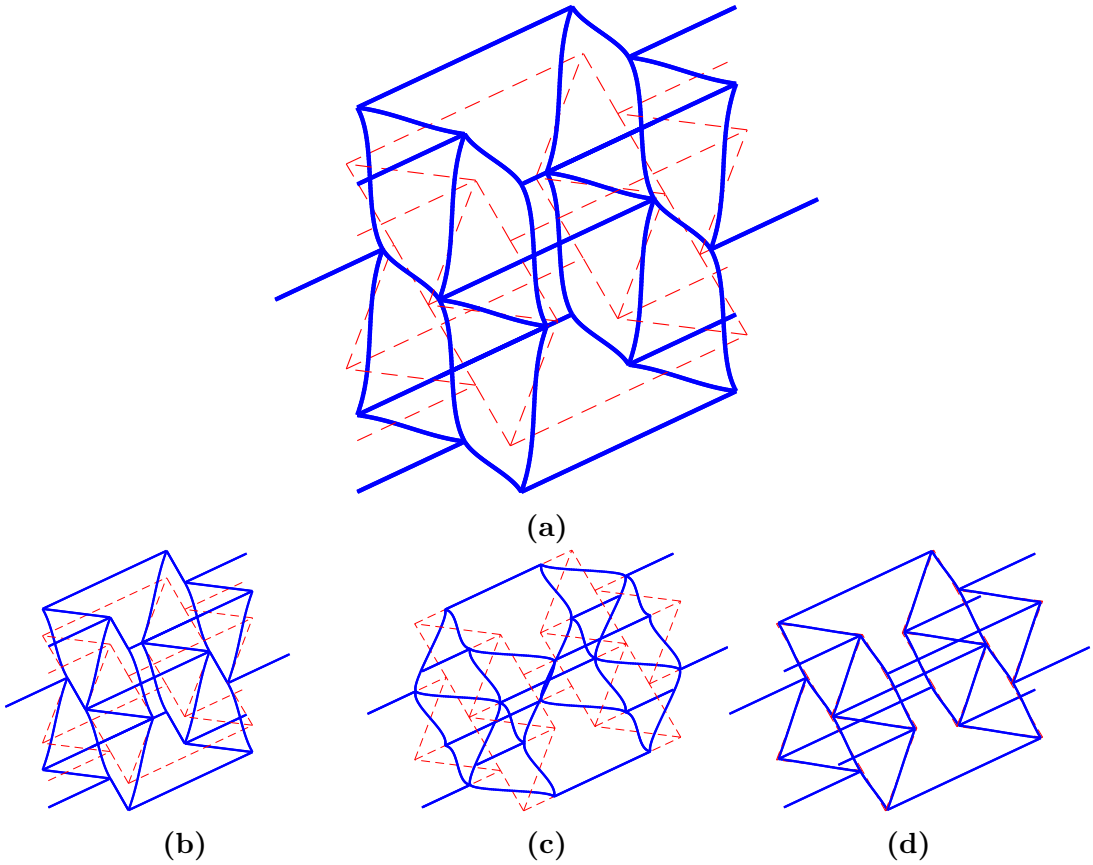


Figure 5.9: Modal deformations for case YY (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 4 (scale 10x).

Here, it can be noticed better what was mentioned in the case XX. Again the first mode contributes the most to the total solution, but still some of the effects, mostly related to the rotation of the nodes, are not fully captured by it. In mode number 2 it is seen a solution that does not behave as the total deformation, since there is not deformation in the Y axis as it is imposed, but only movement of the inner nodes of the cell, keeping fixed the extreme nodes of the X axis. These

kind of modal solutions are the refinements that allow the previous modes to be perfected, in order to obtain a more accurate solution.

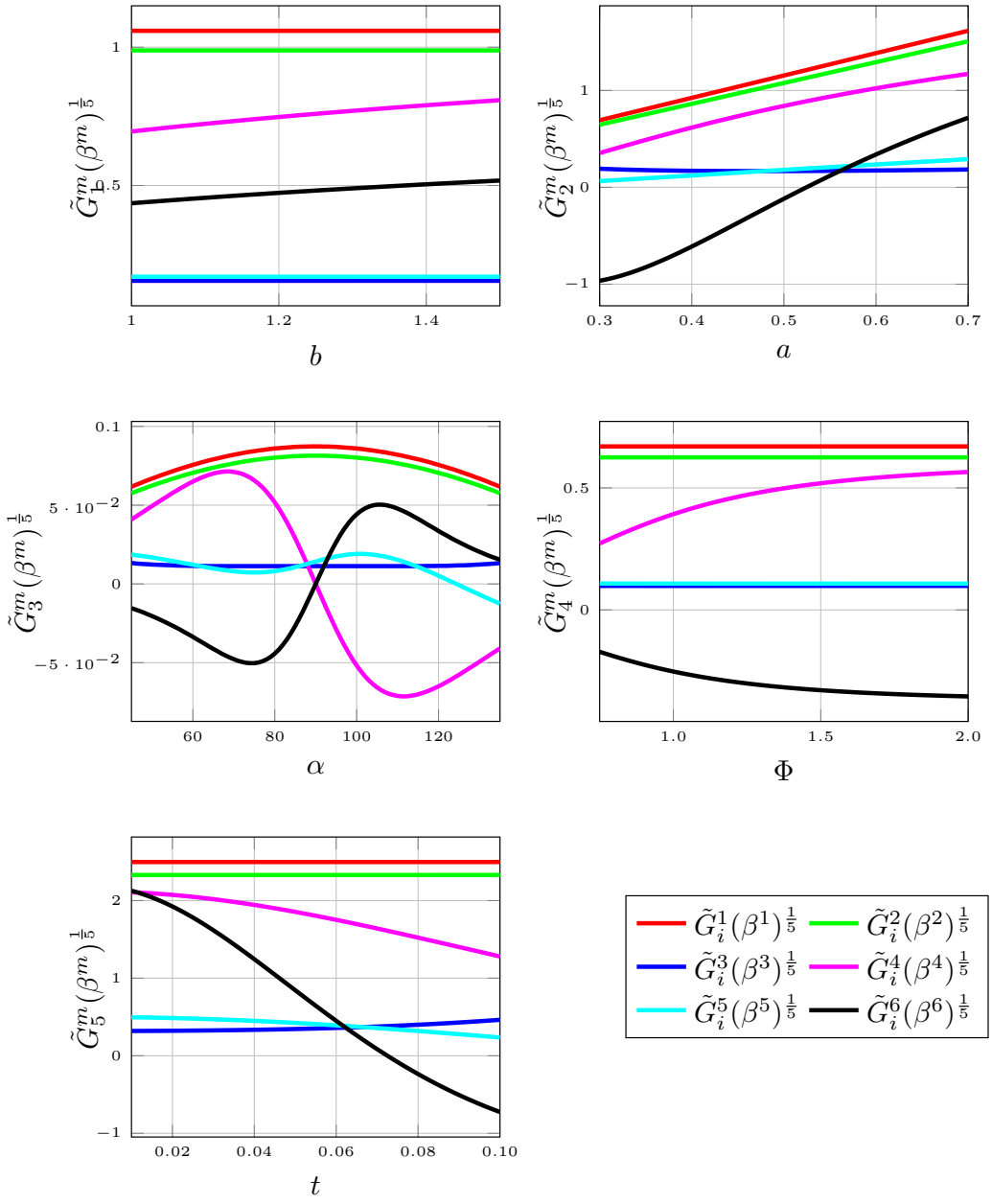


Figure 5.10: Modal functions for case YY.

For the modal functions, behavior is similar to case XX. Relations between parameters and the solution are typically monotonous with the exception of α, where for some modes the functions change while passing through $\alpha = 90°$. Particular are modes 4 and 5, which being symmetrical by the 90° axis, they have a double oscillation, opposed to mode 2 which only shows one peak.

For case ZZ:

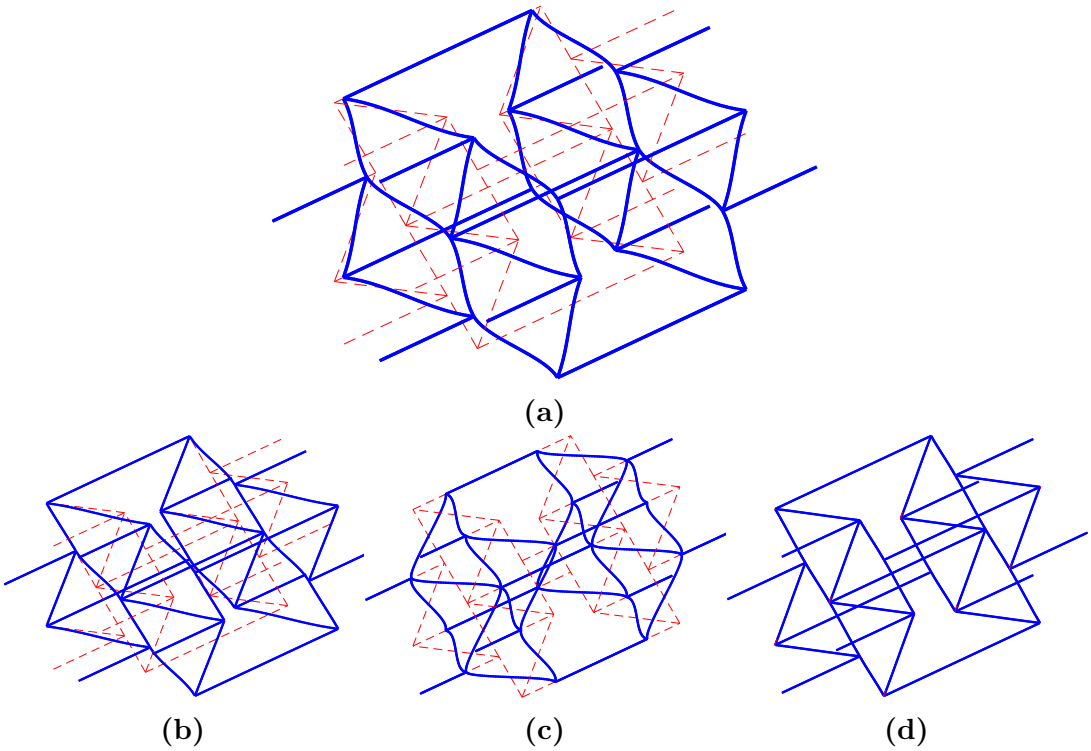


Figure 5.11: Modal deformations for case ZZ (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 6 (scale 10x).

Again, the behavior mentioned before appears here, as this case behaves similarly to the YY one. The first mode as always represent the main part of the solution, but in here it is clear that for mode 2 the rectification of the solution is not necessarily adding more to the previous ones. In this case, mode 2 shows a deformation in opposite directions of mode 1, meaning that mode 1 is not able to capture the deformation in the direction shown in mode 2.

For the case of the modal functions, it is observed a higher dependency on the Φ parameter, mainly in higher modes. Also, with respect to the YY case, the dependency on α is changed, having an oscillation pattern through the interval, mainly symmetrical on the 90° value. This dependence shows higher oscillation in the higher modes. The main difference considering previous cases is the dependence on b and t, which is mainly non existent, given by the constant value of the function for any value of the interval.

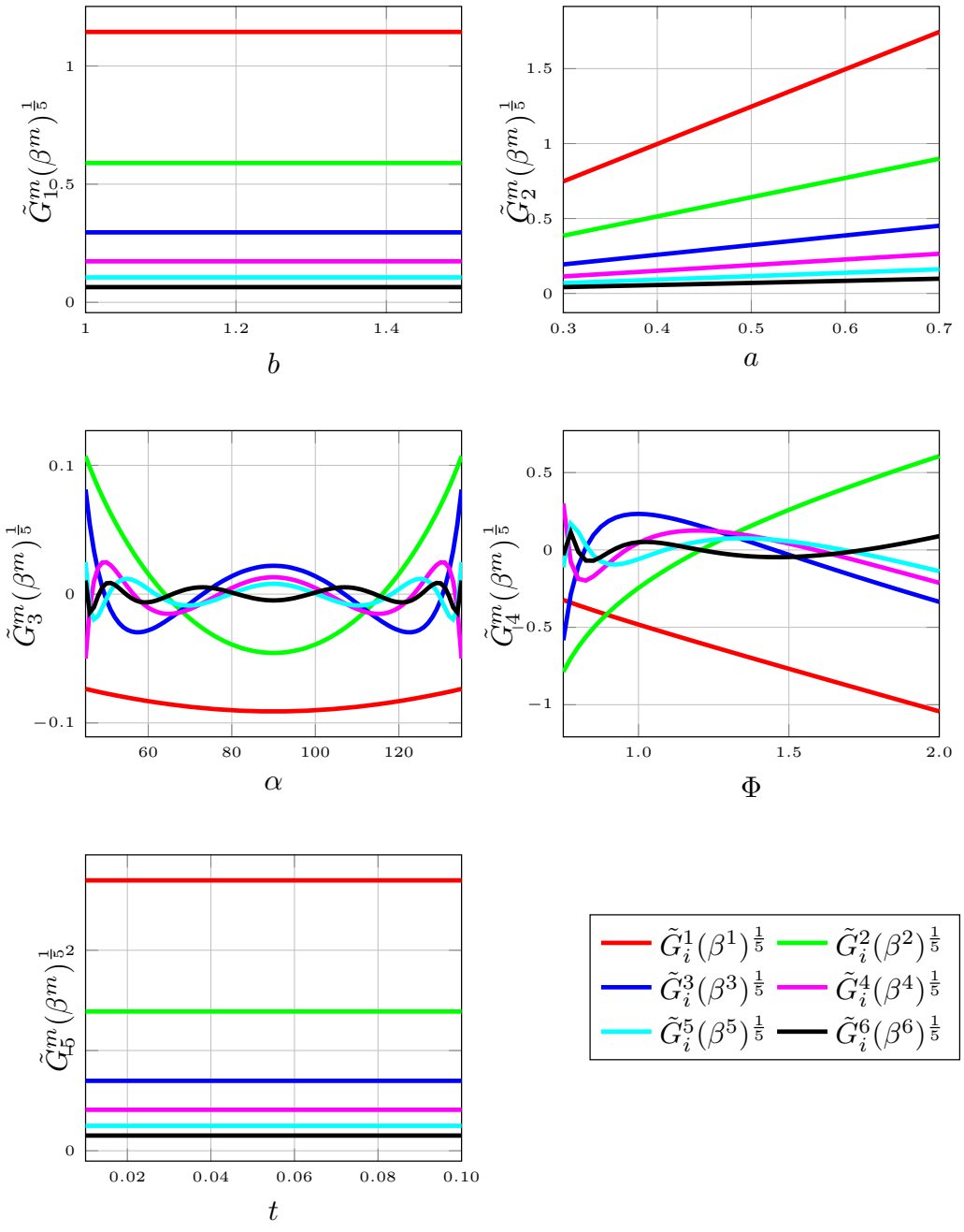


Figure 5.12: Modal functions for case ZZ.

For case YZ:

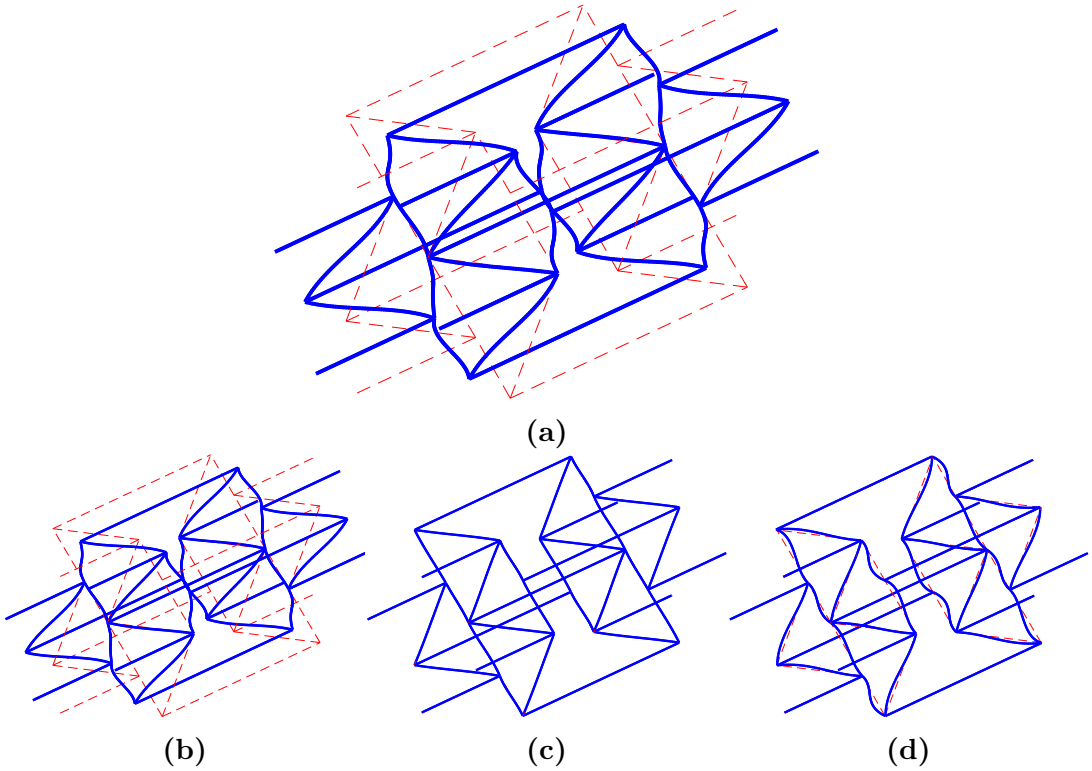


Figure 5.13: Modal deformations for case YZ (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 4 (scale 10x).

For the case of shears, deformations show similar behavior as the principal strains, but the modal functions are different. In the case YZ, it is seen how the results are not affected nor by the values of b nor the values of t, which can be explained by the form of the movement imposed. In this case, as it is seen in the in figure 5.13a, the figure is being rotated around an axis formed by the middle beam of the cell (element assembled by nodes 25 and 26, as shown in figure 2.9). This makes the effect of b non-existent since the length of the cell in the X direction does not interfere with the movement given by the solution. The same happens with the value of t, which mainly contributes to axial and bending stiffnesses, without having an effect on the forces or strains applied in the problem.

An interesting effect to notice here is also the fact that α still shows an inflection point in 90°, where derivatives of the parametric functions changes, but with different shapes as the ones seen in the uniaxial cases. Other parameters show similar relations as seen in previous cases.

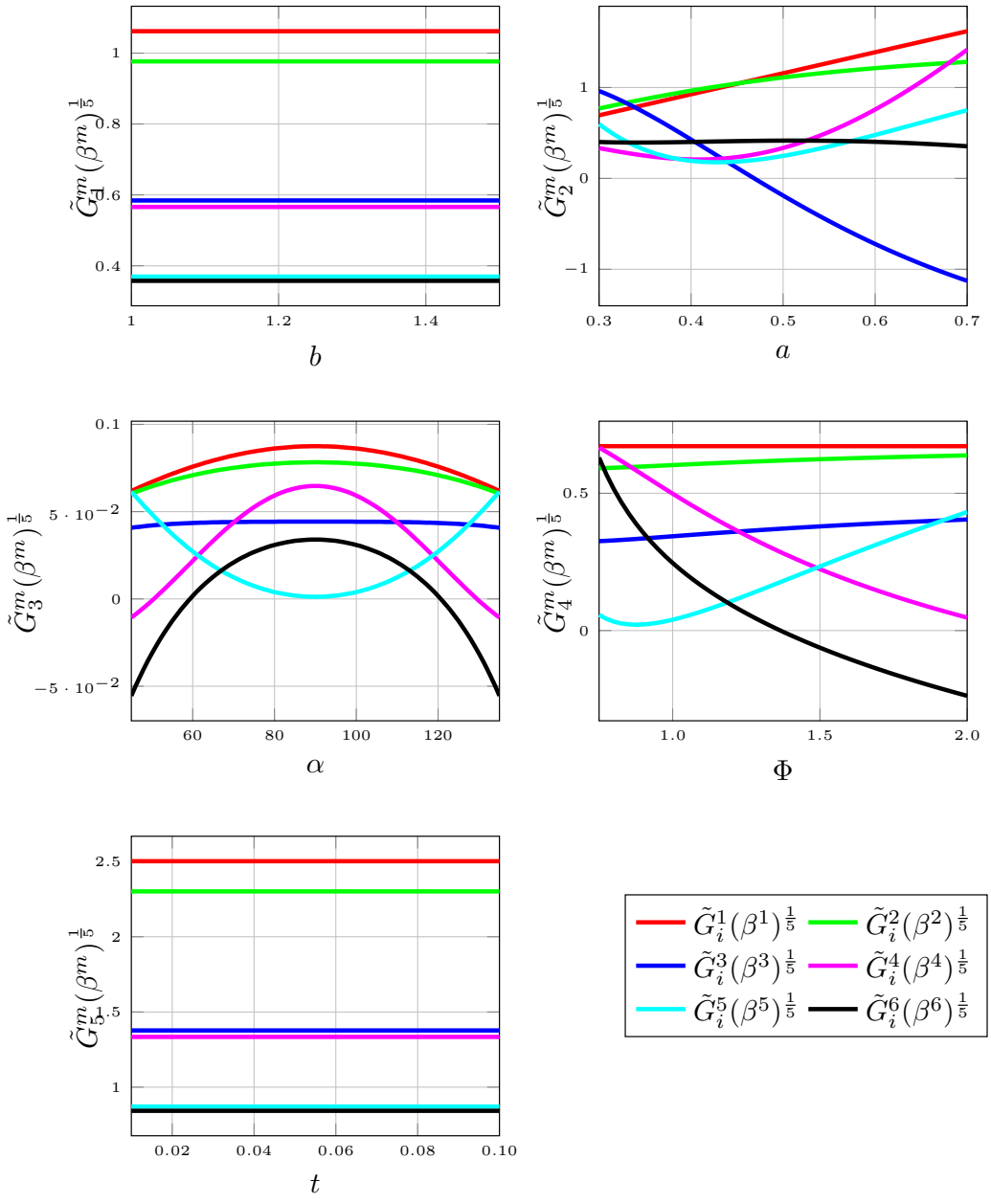


Figure 5.14: Modal functions for case YZ.

For case XZ:

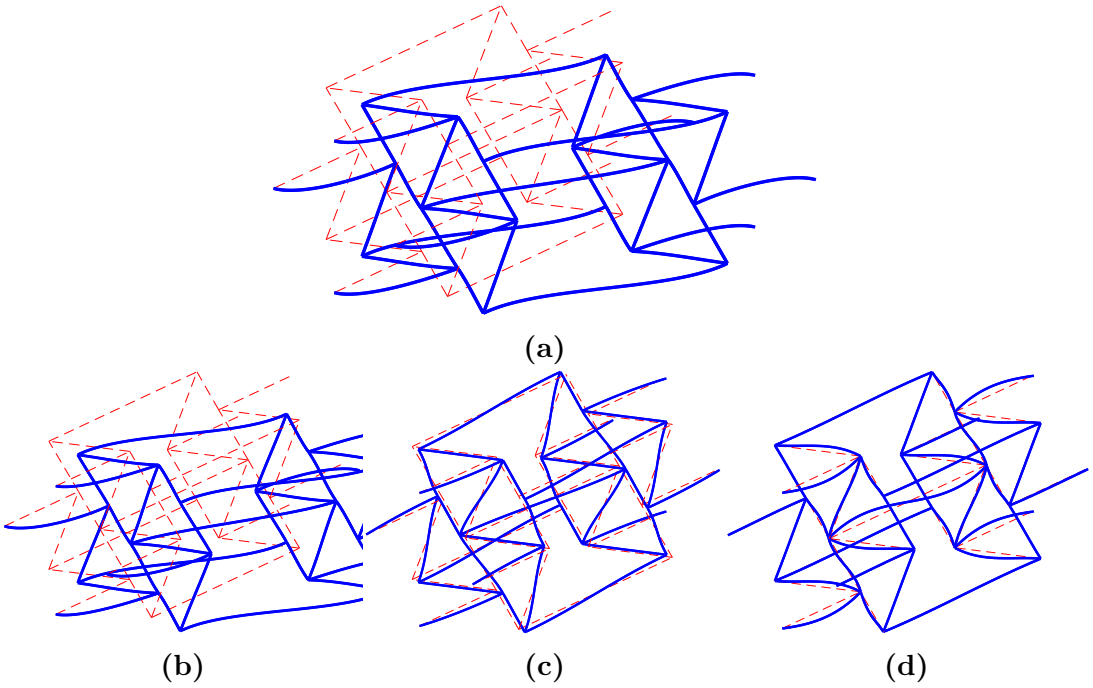


Figure 5.15: Modal deformations for case XZ (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 4 (scale 2x). **d)** Deformation for mode 7 (scale 10x).

The solution here is similar to the other cases. In the deformation of mode 4 it can be appreciated how the deformation obtained is shifted in an opposite direction to the one imposed on the problem, accounting for the modal corrections mentioned earlier. In the modal functions it is visible now the fact that α now does not have inflection points, and the parameter becomes monotonous as the other ones. This leads to the conclusion that for shear cases, α is not as relevant as a parameters as it is for the uniaxial strain cases, which is attributed to the fact that the cell is being deformed without activating the auxetic mechanism.

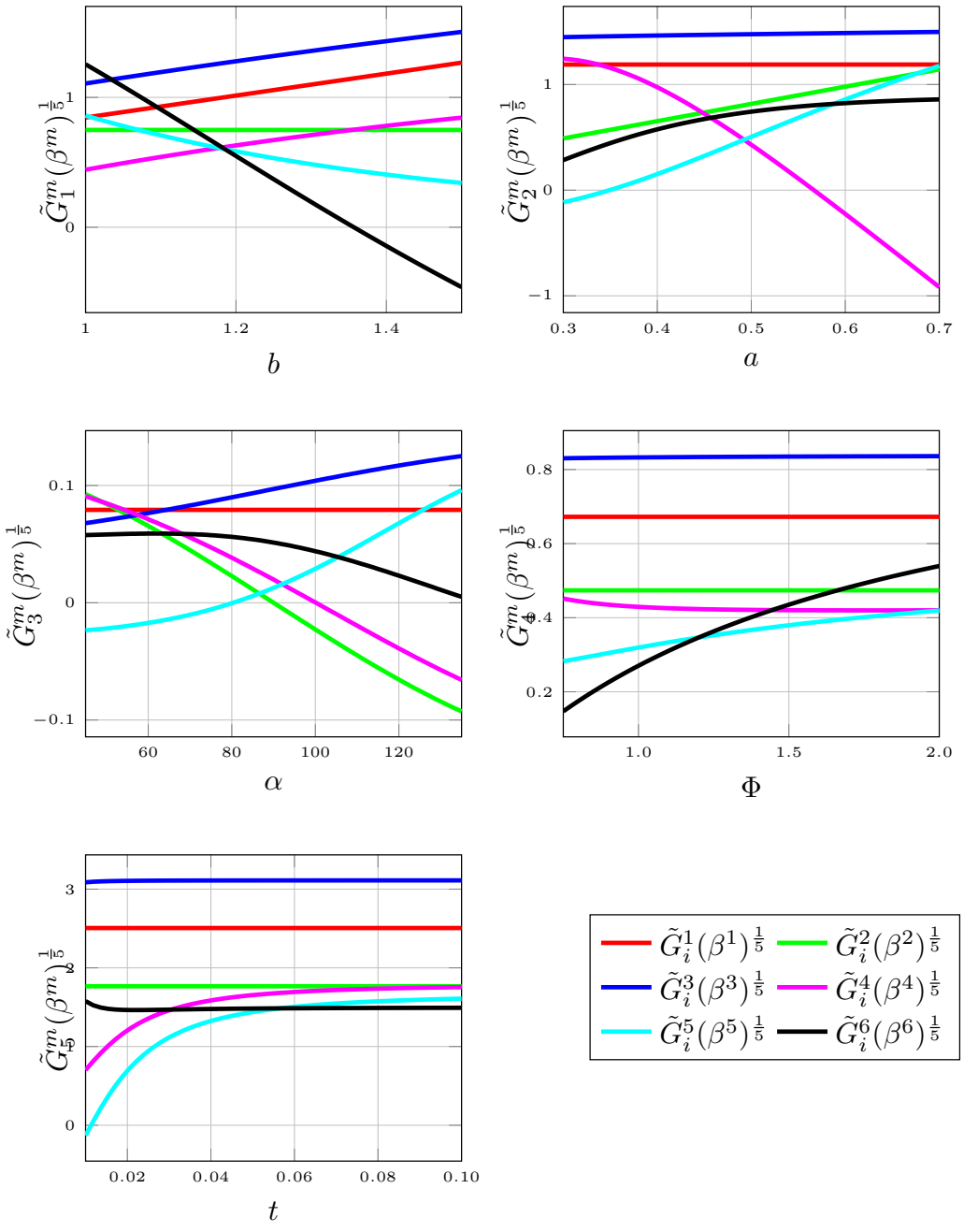


Figure 5.16: Modal functions for case XZ.

For case XY:

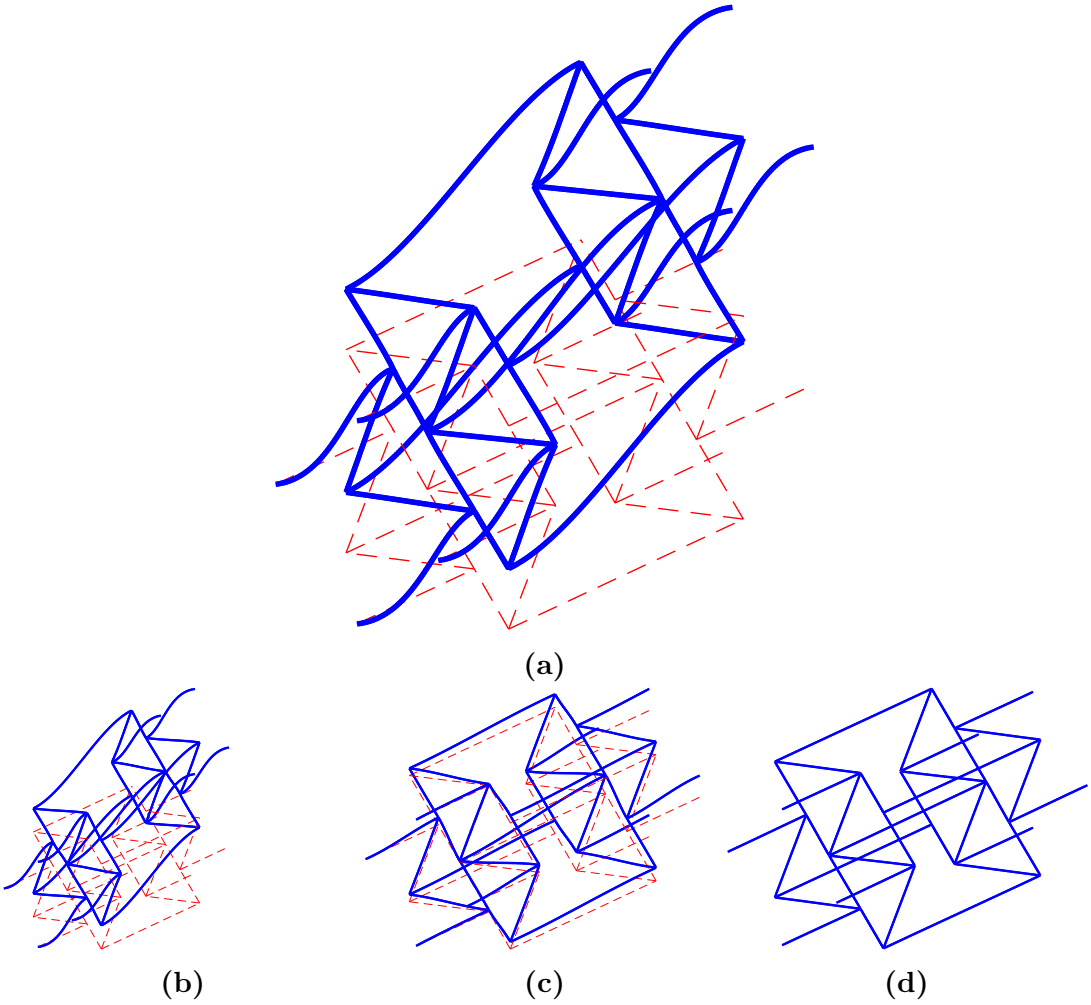


Figure 5.17: Modal deformations for case XY (scale 0.5x). **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 0.5x). **c)** Deformation for mode 3 (scale 10x). **d)** Deformation for mode 5 (scale 10x).

Lastly, in case XY the deformations do not show different behavior, but now it is seen in the modal functions that for every parameter it is shown a monotonous behavior, with the only particularities being parameters a and b, only one mode affected in each. Nonetheless, as it is seen in many cases, the effects are not visible on the first mode, where only for a few parameters there is a variable parametric function, leaving constant functions for the rest. Since mode 1 is the one that represents the majority of the expected behavior for the homogenized cell, the effects of higher modes are not completely appreciable when plotting the deformation contributed by all modes.

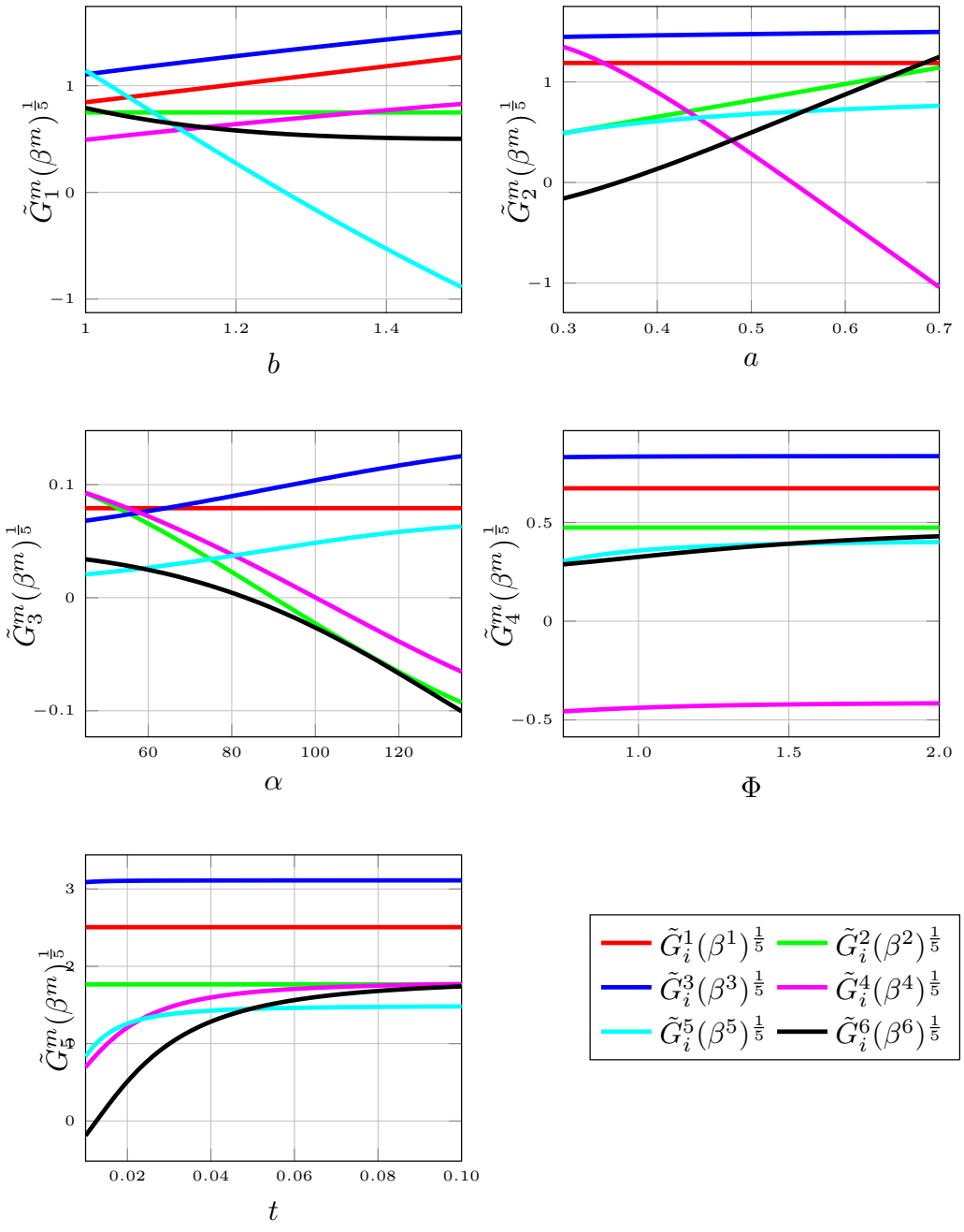


Figure 5.18: Modal functions for case XY.

5.1.3 Effective Poisson's values

The Poisson's values were computed and plotted against variations of the parameters α and a, since it was deemed that these parameters are the most relevant concerning the auxetic properties of the cell. For this, the rest of the parameters were fixed to a set of values in order to be plotted in surface graphs. The values were chosen in order to represent an average case, typically in the middle of each interval. Φ was chosen different than 1 to break the unit cell symmetry between planes XY and XZ. The chosen values are

$$\begin{aligned} b &= 1.25, \\ \Phi &= 1.5, \\ t &= 0.055. \end{aligned}$$

For these values, the different surfaces of ν are shown in figure 5.19. As it can be seen, the behavior of ν_{12} and ν_{21} is similar to their counterpart in the other axis, namely ν_{13} and ν_{31} respectively. For both cases, it is noticed that the Poisson's ratio of the unit cell is mainly dependant on the angle of the oblique elements in the cell, rather than the length of it. This can be obtained intuitively from the definition of the cell, given that the auxetic behavior is only present when the angle is less than 90°. For any higher value, once a extension is applied in the long axis, the cell will contract in the orthogonal directions. This is observed in the plots, while for values of α lower than 90°, the Poisson's ratio is negative, while turning positive in the case of α higher than 90°. This effect is seen dramatically in the ν_{21} and ν_{31} cases, were the Poisson's ratio values change forming a sine-like wave on the surface. Using the post-processing tool this effect is clearly appreciated at the transition point, observed in the sudden changes of deformation at the lattice material.

One interesting point to check is the values of ν_{23} and ν_{32}, which correspond to the ratio of both perpendicular axes. In this case, all the values of ν are negative, independently of the range of values for the angle or oblique element length. However, there is a peak point to the surface given in $\alpha = 90°$, a local maximum of the surface. This may be explained mainly by the fact that at 90° the cell presents the form of a rectangular prism, so the value of ν approaches zero, in order to increase its magnitude again for higher angles.

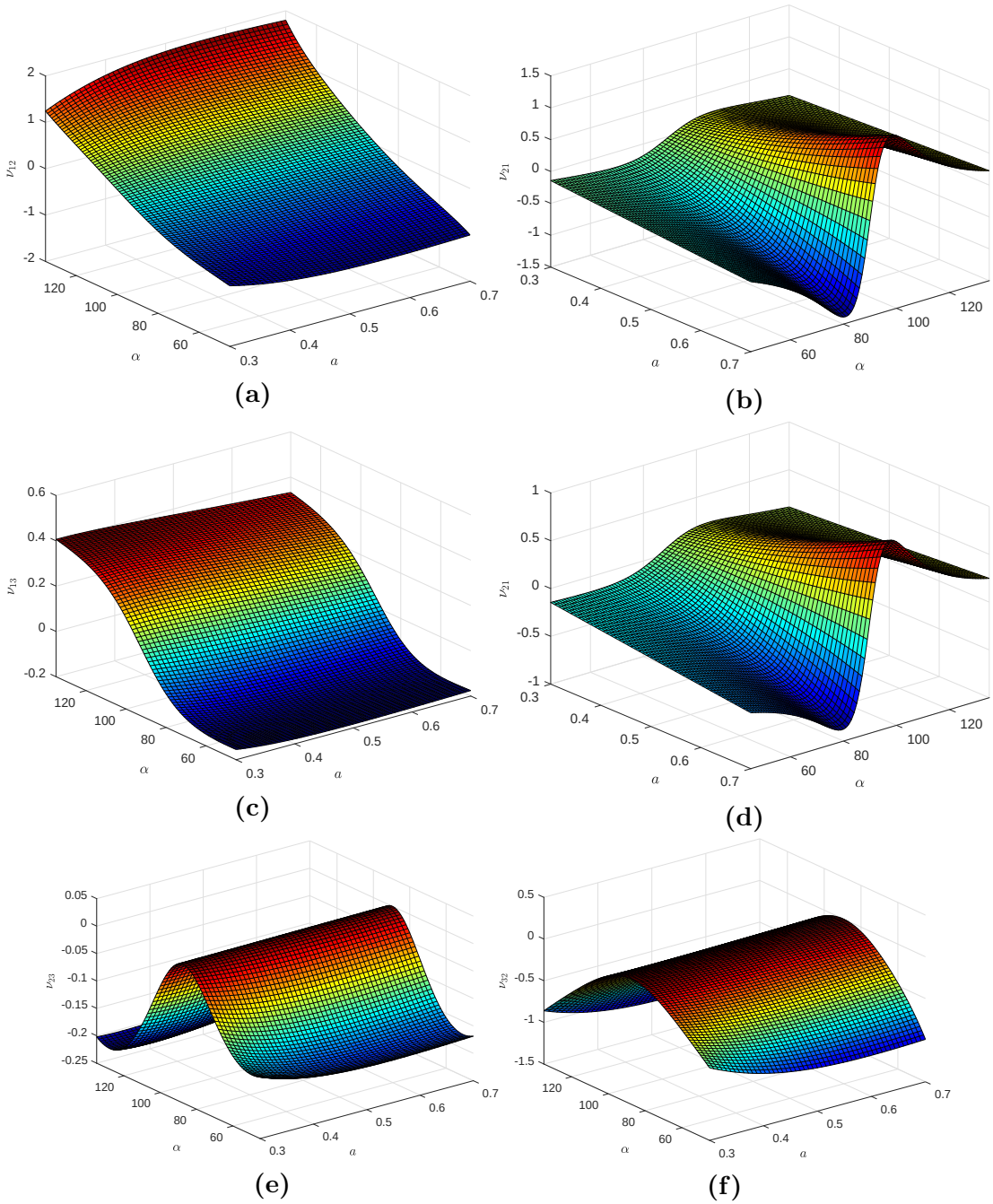


Figure 5.19: Values for different ν. **a)** Values of ν_{12}. **b)** Values of ν_{21}. **c)** Values of ν_{13}. **d)** Values of ν_{31}. **e)** Values of ν_{23}. **f)** Values of ν_{32}.

5.1.4 Computed errors

Once the problem is solved using the previous considerations and applying the method, the accuracy of the solution is tested. For computing the error, the structure resolved by PGD (in the version using compression) is compared against a procedure without separation of parameters. Since the solution of the unit cell is not known in an analytical form, the comparison is made using finite elements evaluated at prescribed values of the parameters. Three sets of errors are computed, defined as

$$\epsilon_r = \frac{||\boldsymbol{U}(\boldsymbol{\mu}_p) - \boldsymbol{U}_{PGD}(\boldsymbol{\mu}_p)||}{||\boldsymbol{U}(\boldsymbol{\mu}_p)||}, \tag{5.1}$$

$$\epsilon_{L2} = \frac{\left[\boldsymbol{U}(\boldsymbol{\mu}_p) - \boldsymbol{U}_{PGD}(\boldsymbol{\mu}_p)\right]^T \left[\boldsymbol{M}_u(\boldsymbol{\mu}_p)\right] \left[\boldsymbol{U}(\boldsymbol{\mu}_p) - \boldsymbol{U}_{PGD}(\boldsymbol{\mu}_p)\right]}{\left[\boldsymbol{U}(\boldsymbol{\mu}_p)\right]^T \left[\boldsymbol{M}_u(\boldsymbol{\mu}_p)\right] \left[\boldsymbol{U}(\boldsymbol{\mu}_p)\right]}, \tag{5.2}$$

$$\epsilon_{H1} = \frac{\left[\boldsymbol{U}(\boldsymbol{\mu}_p) - \boldsymbol{U}_{PGD}(\boldsymbol{\mu}_p)\right]^T \left[\boldsymbol{K}_{PGD}(\boldsymbol{\mu}_p)\right] \left[\boldsymbol{U}(\boldsymbol{\mu}_p) - \boldsymbol{U}_{PGD}(\boldsymbol{\mu}_p)\right]}{\left[\boldsymbol{U}(\boldsymbol{\mu}_p)\right]^T \left[\boldsymbol{K}_{PGD}(\boldsymbol{\mu}_p)\right] \left[\boldsymbol{U}(\boldsymbol{\mu}_p)\right]}. \tag{5.3}$$

The first equation corresponds to the normal relative error between the norms of vectors. The second equation is the so called $L2$ error, which computes the error in a more precise way, since it accounts for the differences between displacements and rotations, also providing an idea of how the solutions differ from the whole spatial domain and not only the nodal positions. Finally, the third equation is the $H1$ error, which also accounts for this difference, since stiffness matrices also differentiate displacements and rotations. The errors are computed with an increasing amount of modes in order to obtain a trend and analyze the effect of this relation, which allows to see the improvement in accuracy due to the extra modes. Since the FEM solution can only be obtained for particular sets of parameters, 15 random cases were chosen to be studied. These 15 cases are shown in table 5.1.

Table 5.1: Parameters used for error.

	b	a	α	Φ	t
Case 1	1.2500	0.5000	60.0000	1.0000	0.0550
Case 2	1.4100	0.6040	70.0000	1.1750	0.0244
Case 3	1.4600	0.5960	106.0000	1.0000	0.0820
Case 4	1.0600	0.4600	104.0000	1.0500	0.0370
Case 5	1.4600	0.5640	59.0000	1.5250	0.0568
Case 6	1.3200	0.3640	55.0000	1.3500	0.0244
Case 7	1.0400	0.5880	90.0000	1.1750	0.0640
Case 8	1.1400	0.3080	132.0000	1.8000	0.0334
Case 9	1.2700	0.4120	75.0000	1.4750	0.0694
Case 10	1.4800	0.3160	98.0000	1.4500	0.0730
Case 11	1.4900	0.3320	65.0000	1.9000	0.0784
Case 12	1.0800	0.6280	113.0000	1.1000	0.0496
Case 13	1.4900	0.5800	68.0000	1.7000	0.0172
Case 14	1.4800	0.4280	91.0000	1.7000	0.0298
Case 15	1.2400	0.6840	108.0000	1.2250	0.0928

The results are plotted in figures 5.20 to 5.25.

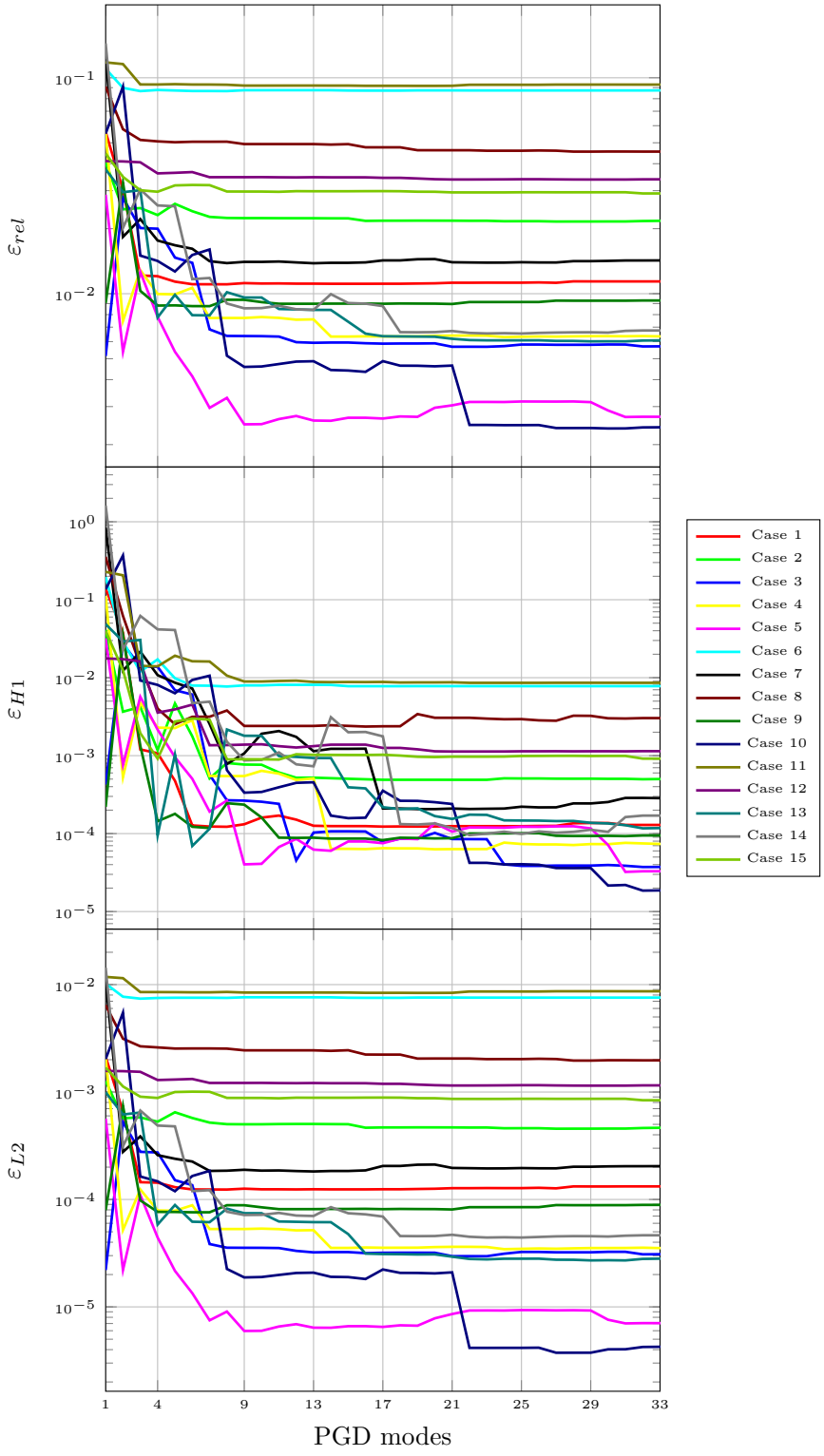


Figure 5.20: Error for all study cases in XX. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

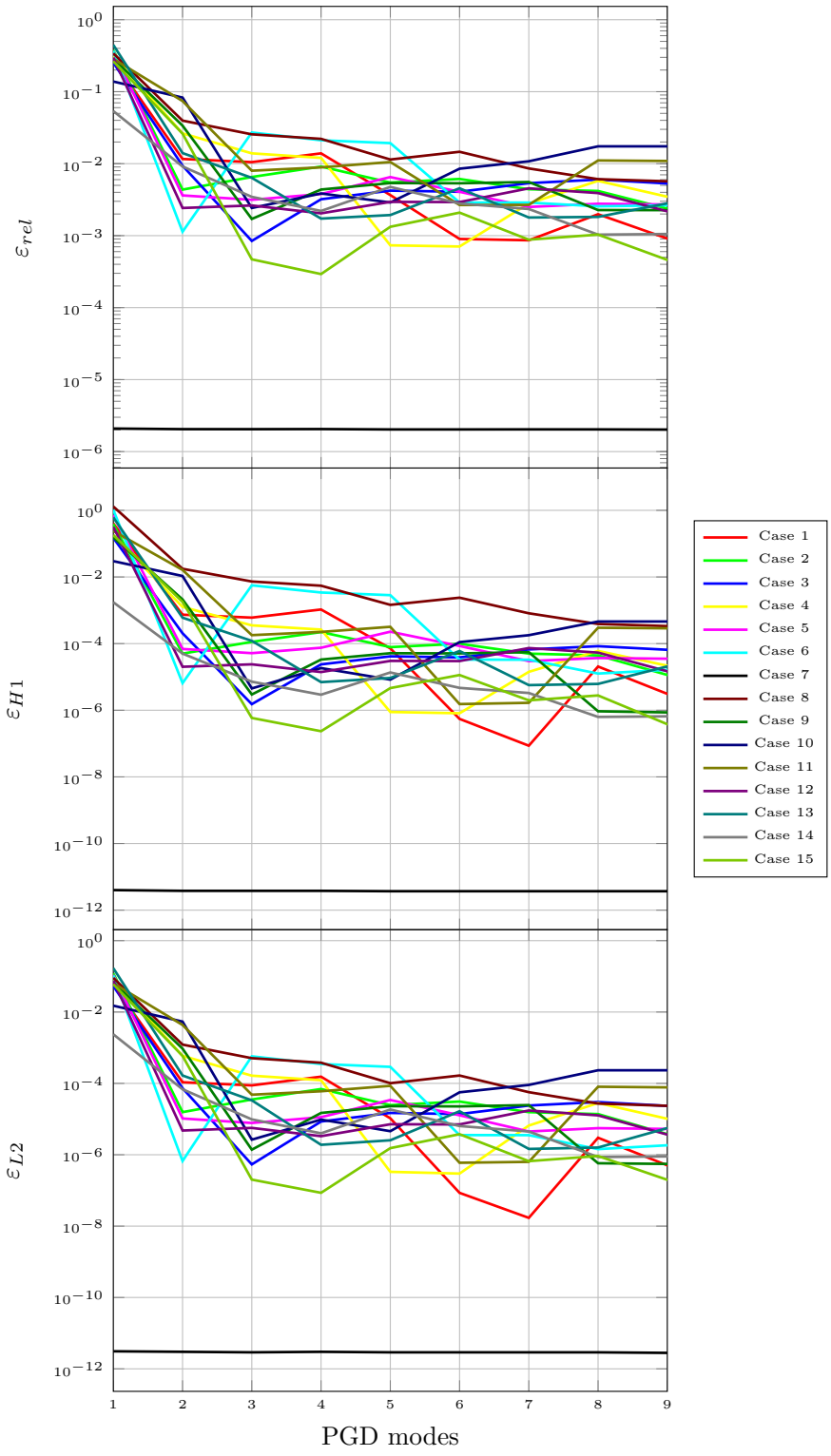


Figure 5.21: Error for all study cases in YY. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

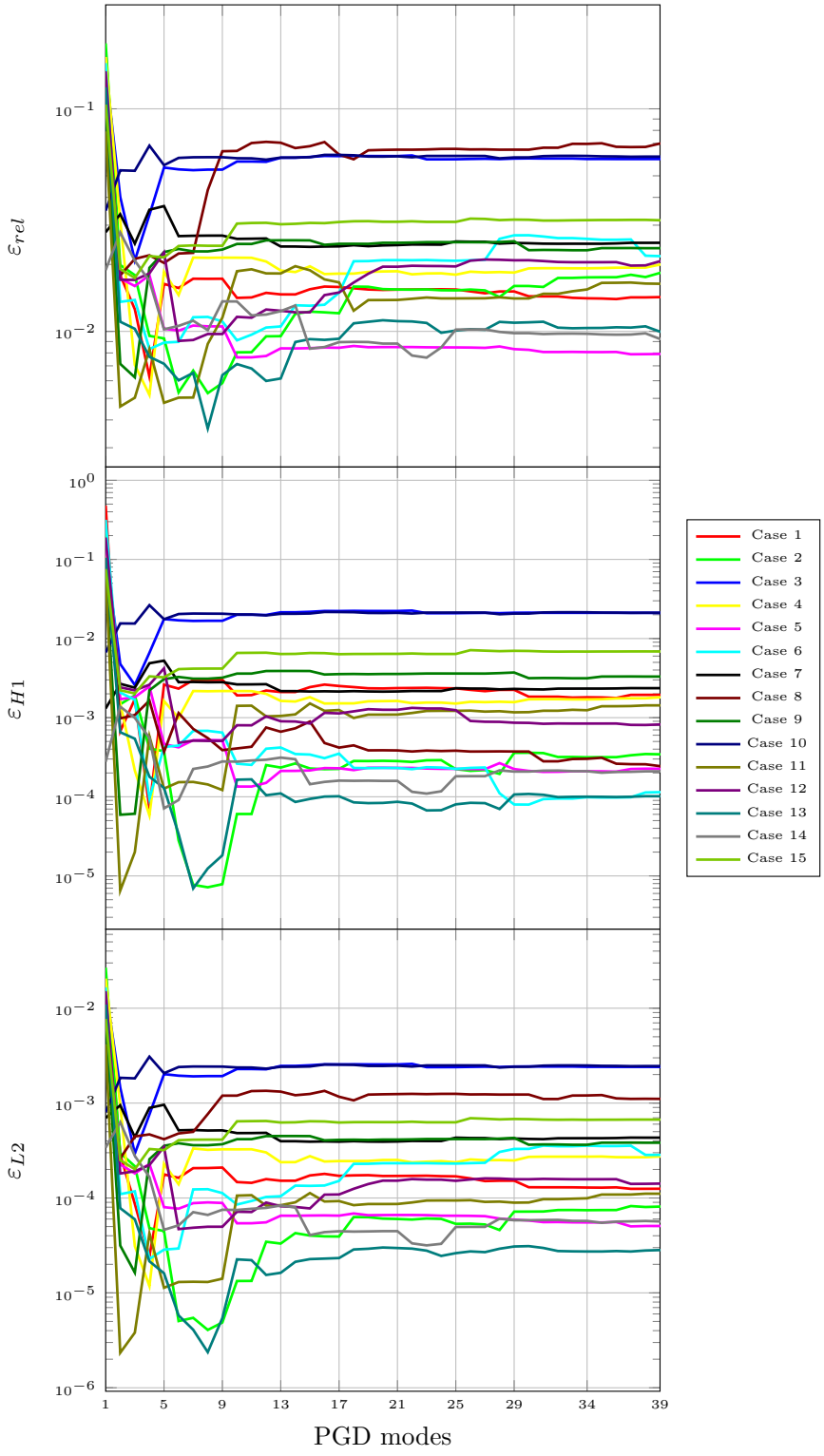


Figure 5.22: Error for all study cases in ZZ. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

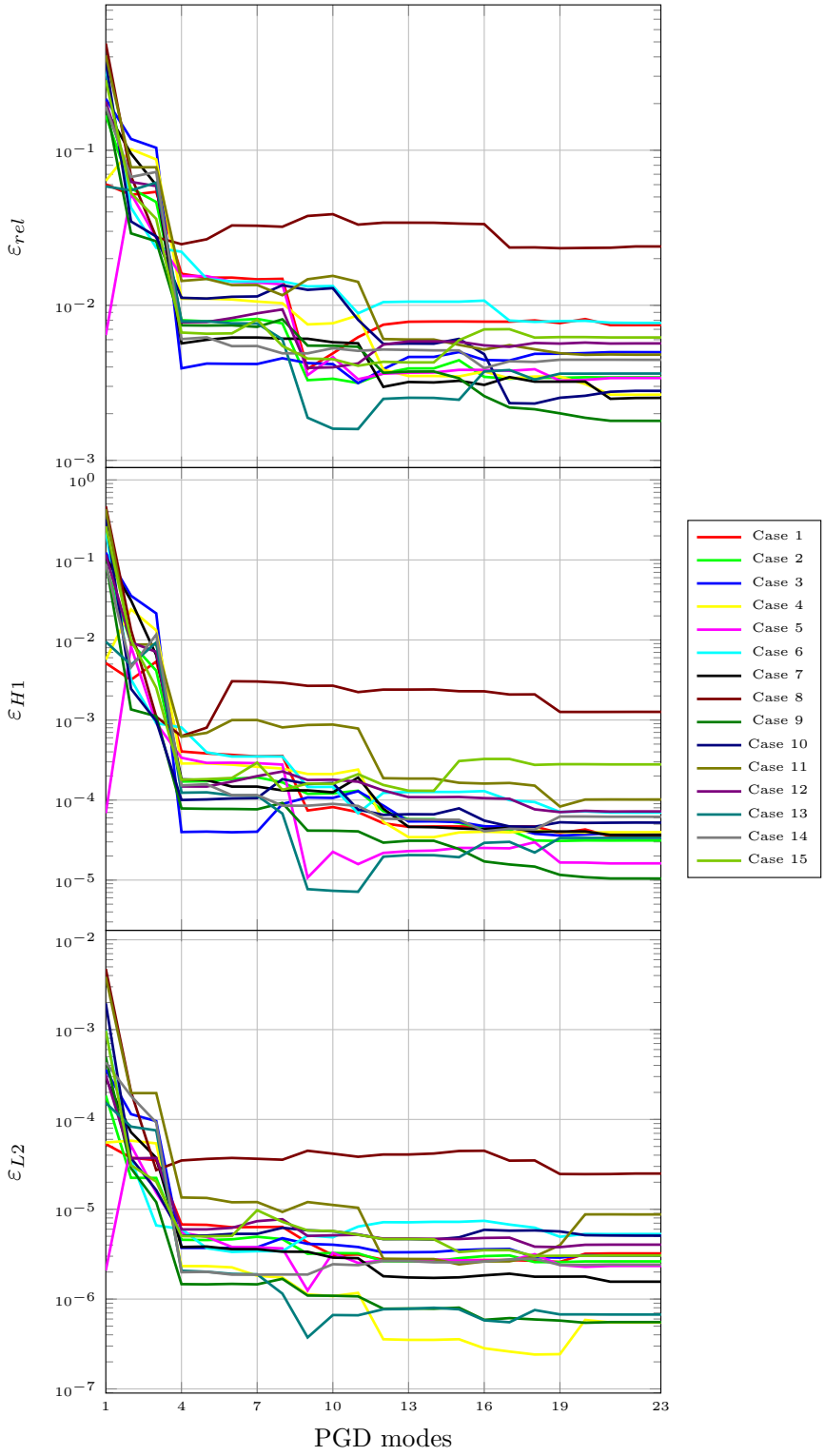


Figure 5.23: Error for all study cases in YZ. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

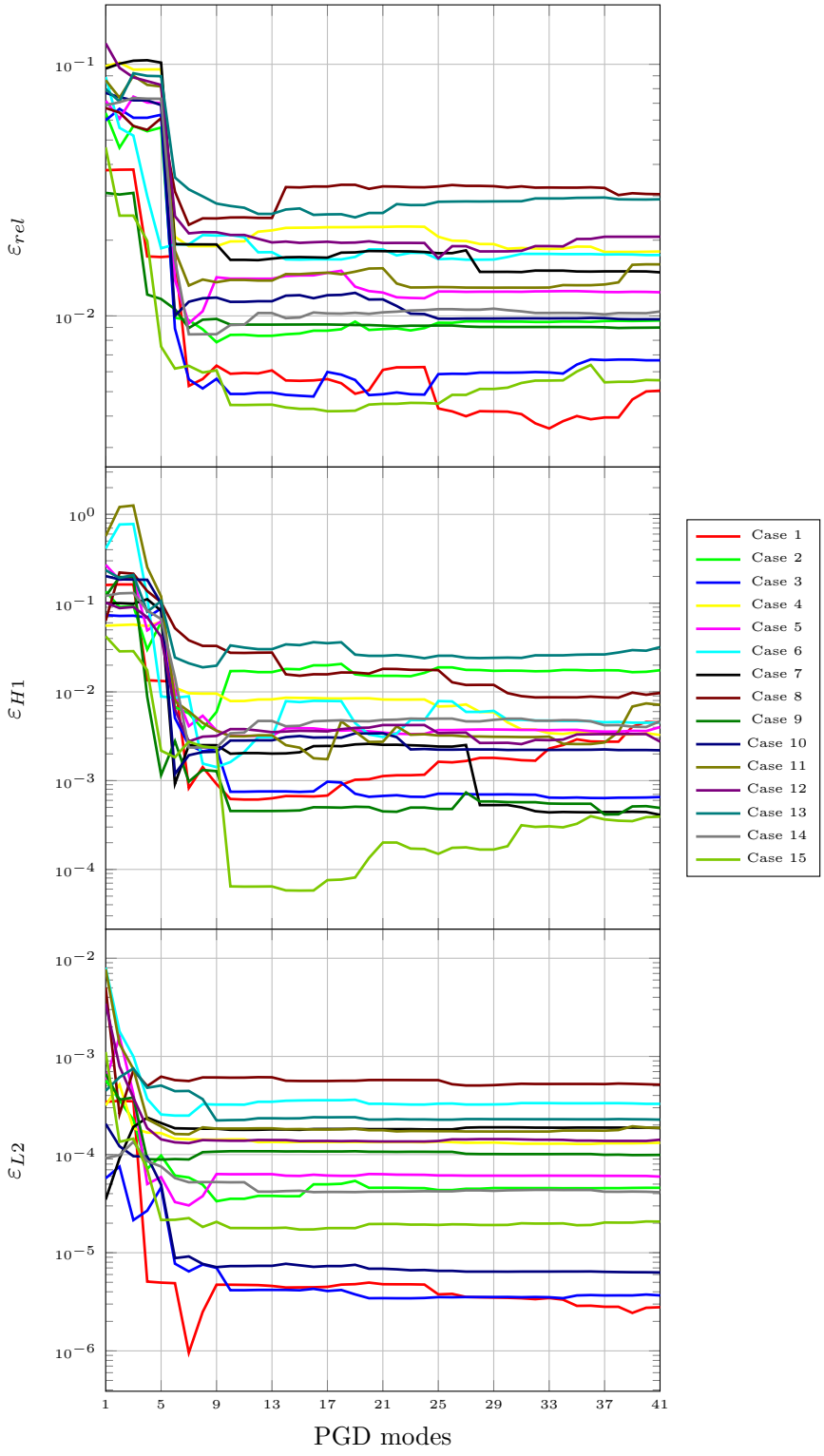


Figure 5.24: Error for all study cases in XZ. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

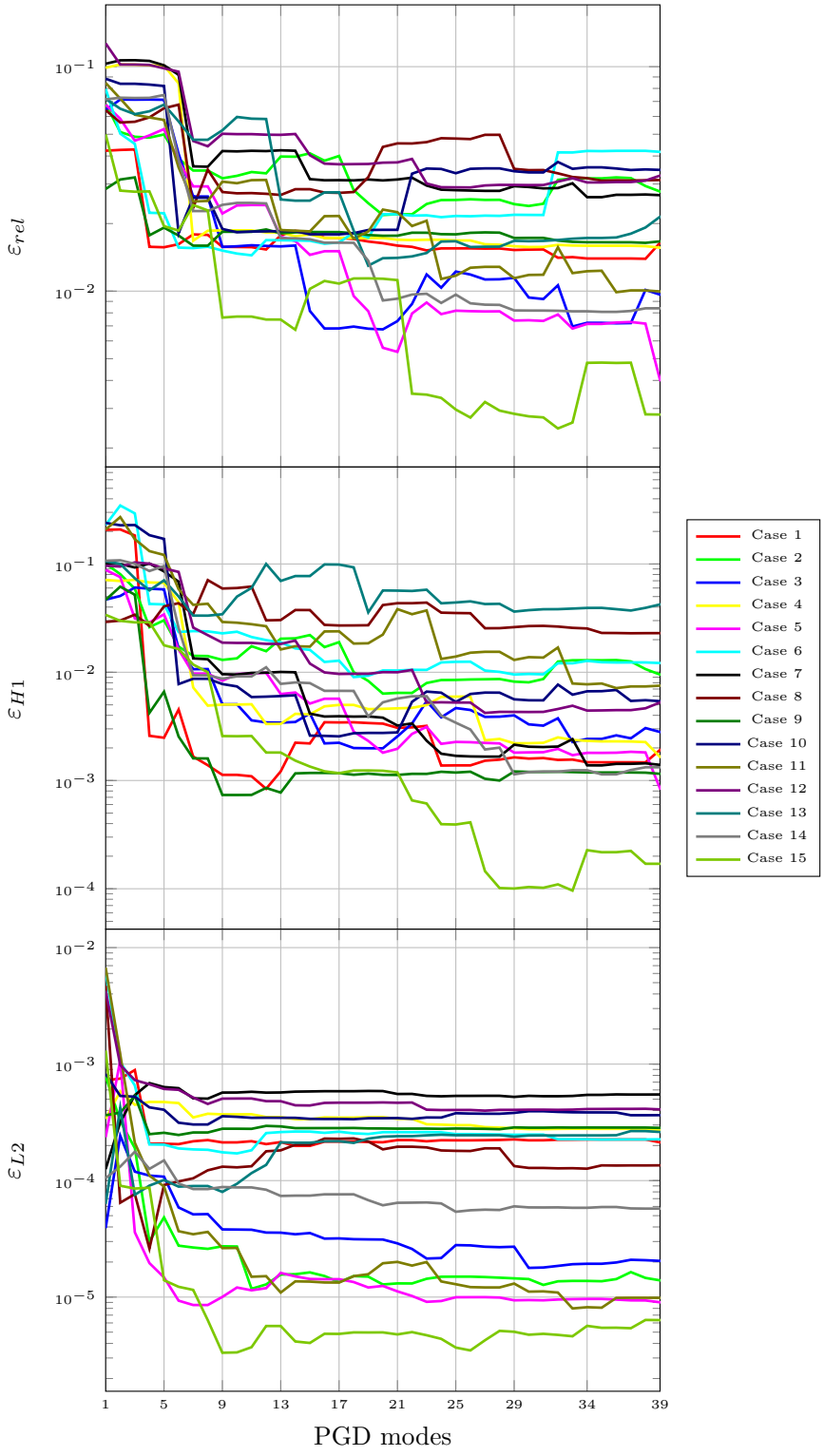


Figure 5.25: Error for all study cases in XY. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

As can be seen from the figures, the errors are dispersed, not having a common value of convergence, so the only information that can be extracted is related to the behavior of the method in a general way. The errors are plotted in order of accuracy, leaving the relative euclidean norm at the top, which gives a bigger error, since it only considers variation at the nodal positions, followed by the H_1 error and finally the L_2 one in the lower place, which is the one giving the lowest values. At first glance, it can be noticed that the order of the error depends of the case studied, were some of the cases display smaller errors, meaning that the approximation is better. Particular are the cases YY and YZ, where the error is of the order of 10^{-4} and lower for the L_2 error, as opposed to the case XX, which has the worst approximation of all. The explanation for the errors comes from the solver itself and the way it handles the solution. Since the method relies on solving residuals in a global parametric way and not the exact case for every value of the parameters, as it is done in FEM, the solutions differ from each other. It can be seen also that the general trend is a critical reduction of the error during the first 10 computed modes, leading to an stabilization of the error from then on. This is also related to the fact that there are computing errors not only between PGD and FEM, but also the error associated to the approximation already introduced in order to have a separable stiffness matrix. For the analysis of the dispersion of the values, an analysis on the variation of the parameters is made, and the results are shown in figure 5.26. For this analysis, the parameters are left constant for the same standard case used in other analysis and figures ($a = 0.3$, $b = 1.25$, $\alpha = 60°$, $\Phi = 1.0$ and $t = 0.055$), with the particularity that one of the parameters is kept as a variable, in order to see how the error performs in the interval. This values are then normalized in an interval from 0 to 1 in order to display them together.

In figure 5.26, it can be seen how for each one of the cases, different parameters are dominant in the differences between the PGD approximation and the FEM solution, which also correlates with previous analysis. For example, it has been seen previously that case YZ is not dependent on the values of b and t, which is also displayed here, since the error is constant for those parameters. Also, in figure 5.21 it is seen that case 7 display an error that is virtually zero, which is valid since it is appreciated that for the middle of the interval (i.e., $\alpha = 90°$) the approximation is exact to machine precision. In a general appreciation, it seems that the errors tend to be bigger in the boundaries of the intervals, with some of the parameters showing the opposite behavior.

In a general note, the errors are in an acceptable margin. Considering the L_2 error, for every one of the random cases studied the error is never bigger than 1%, which for the scale of the projected uses of the cell (in the order of centimeters down to micrometers) is negligible.

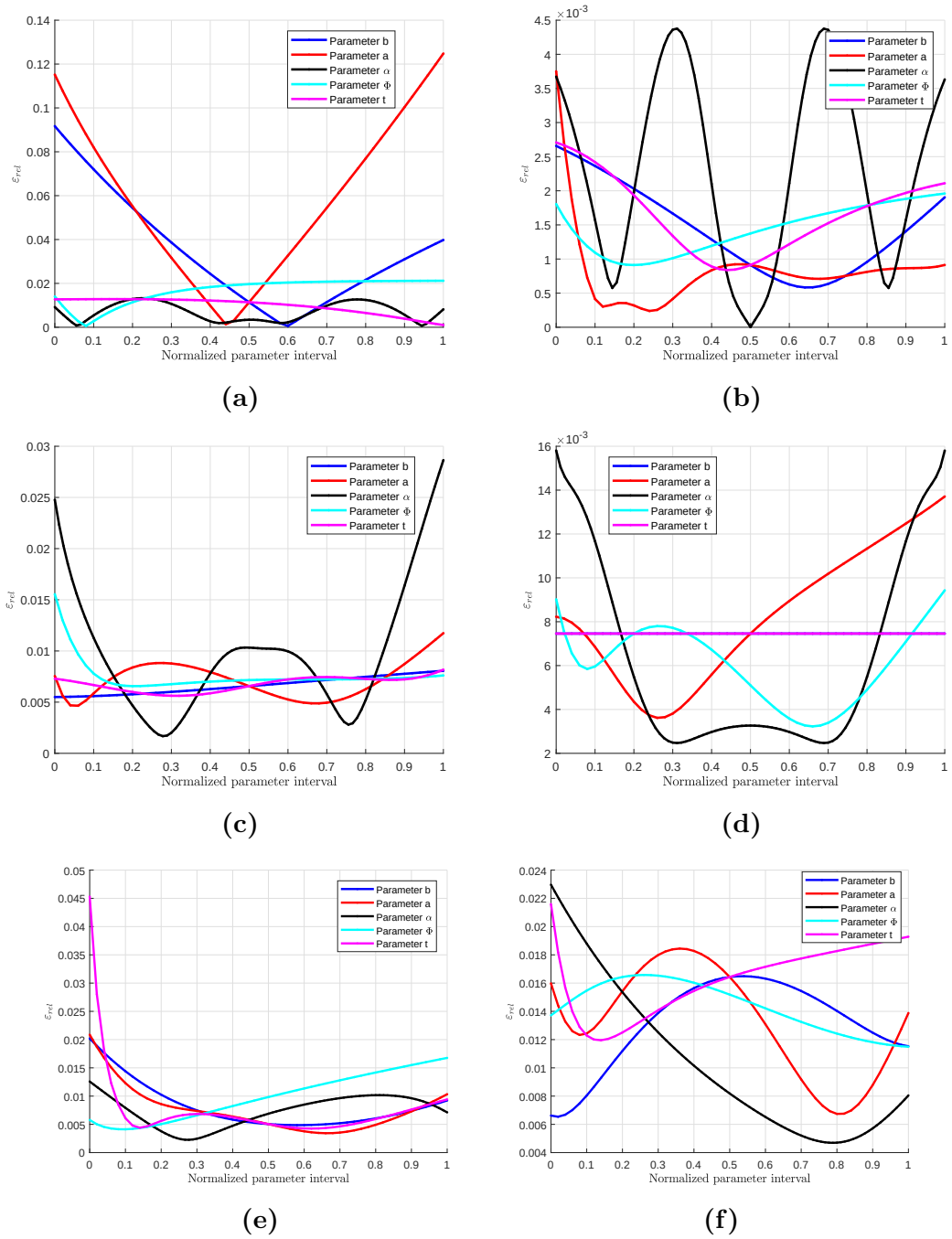


Figure 5.26: Error plots by parameter for different homogenization cases. **a)** Case XX. **b)** Case YY. **c)** Case ZZ. **d)** Case YZ. **e)** Case XZ. **f)** Case XY.

5.2 Full pattern structure

In this section, the computation of the $3 \times 3 \times 3$ pattern is performed, as presented in section 2.4.2. In this case, a prescribed strain of 10% of the length of the structure was applied on the desired direction in each of the three load cases proposed, as seen in figures 2.11a to 2.11c. Using the PGD solver, the results obtained are shown below. To display the results, the same set of parameters used in the case of homogenization was applied in here, in order to have a consistent set of images between both cases.

5.2.1 Modal amplitudes

For the full pattern problem, the solutions were only computed considering the PGD compression scheme. The modal amplitudes are plotted and shown in the figures 5.27 to 5.29. For the full pattern case, only the solutions with PGD compression are studied.

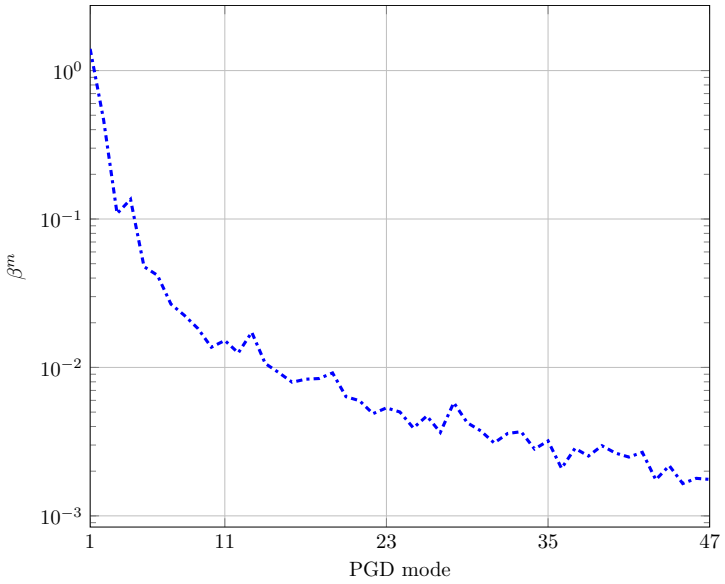


Figure 5.27: Modal amplitudes for XX case.

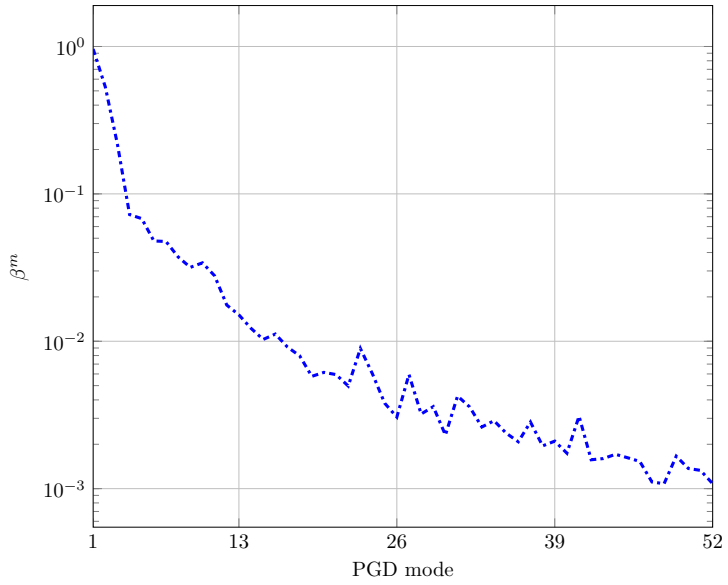


Figure 5.28: Modal amplitudes for YY case.

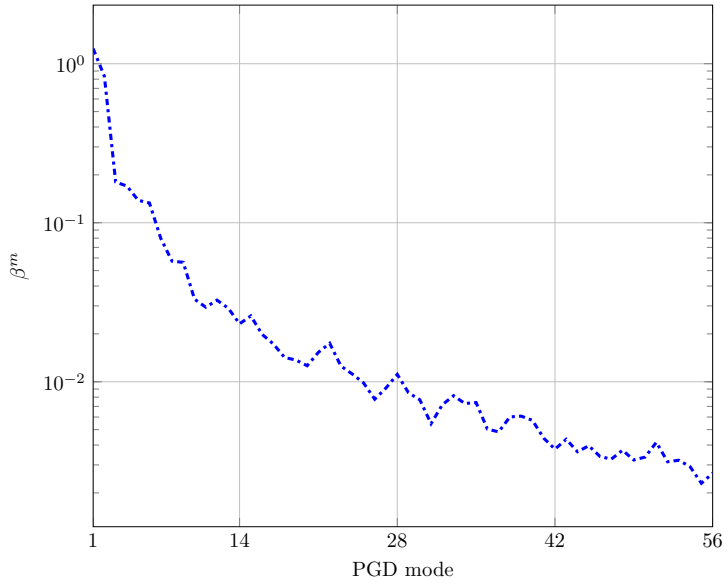
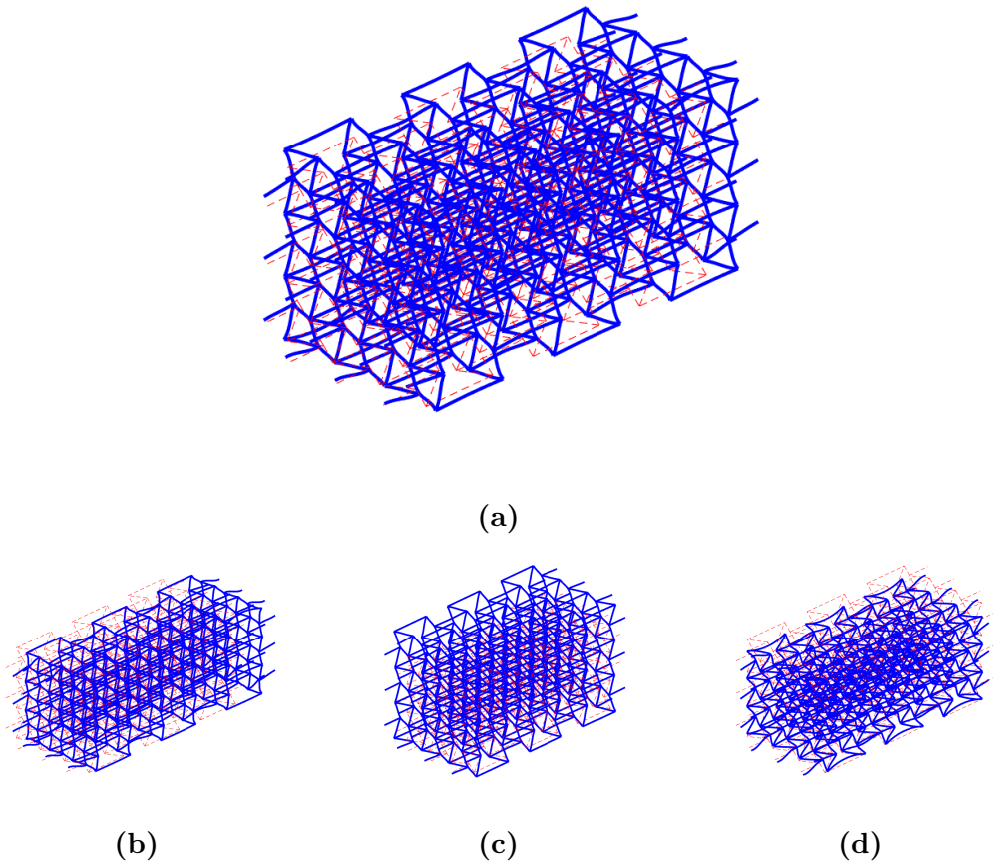


Figure 5.29: Modal amplitudes for ZZ case.

The results for the full pattern structure follow a similar trend compared to the values of the homogenized cell. The values of the modal amplitude decrease with the amount of modes computed, until a set tolerance is reached. Oscillations on the values of the amplitudes is also observed as it was seen before, due to finding of modes that are not necessarily less relevant than the previous ones.

5.2.2 Deformations and modal functions

The deformations and modal functions are also shown for the pattern, allowing us to see a visible deformation of a complete periodic lattice. The results for each strain case are shown in figures 5.30 to 5.34.



(a)

(b) **(c)** **(d)**

Figure 5.30: Modal deformations for case XX. **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 2x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 5 (scale 10x).

The deformations observed for the full pattern are the expected results. While applying the uniaxial tension in the X axis, there is a displacement occurring in the two orthogonal axis, with the structure expanding due to the auxetic behavior. Similarly to the homogenization case, the first mode of the computed presents the biggest contribution to the total solution. What is interesting to notice is the boundary effects. Since now the structure is being computed fully, the deformation on the outer elements, due to not be constricted by other unit cells manifests, capturing one of the effects that was not seen on the homogenization problem. The modal functions show a behavior that is new with respect to the homogenization, mainly the shape of the α function, in which its derivative has no

change of signs, behaving in a monotonous way. For all the other parameters, the values of the functions tend correlate to the case XX of the cell homogenization.

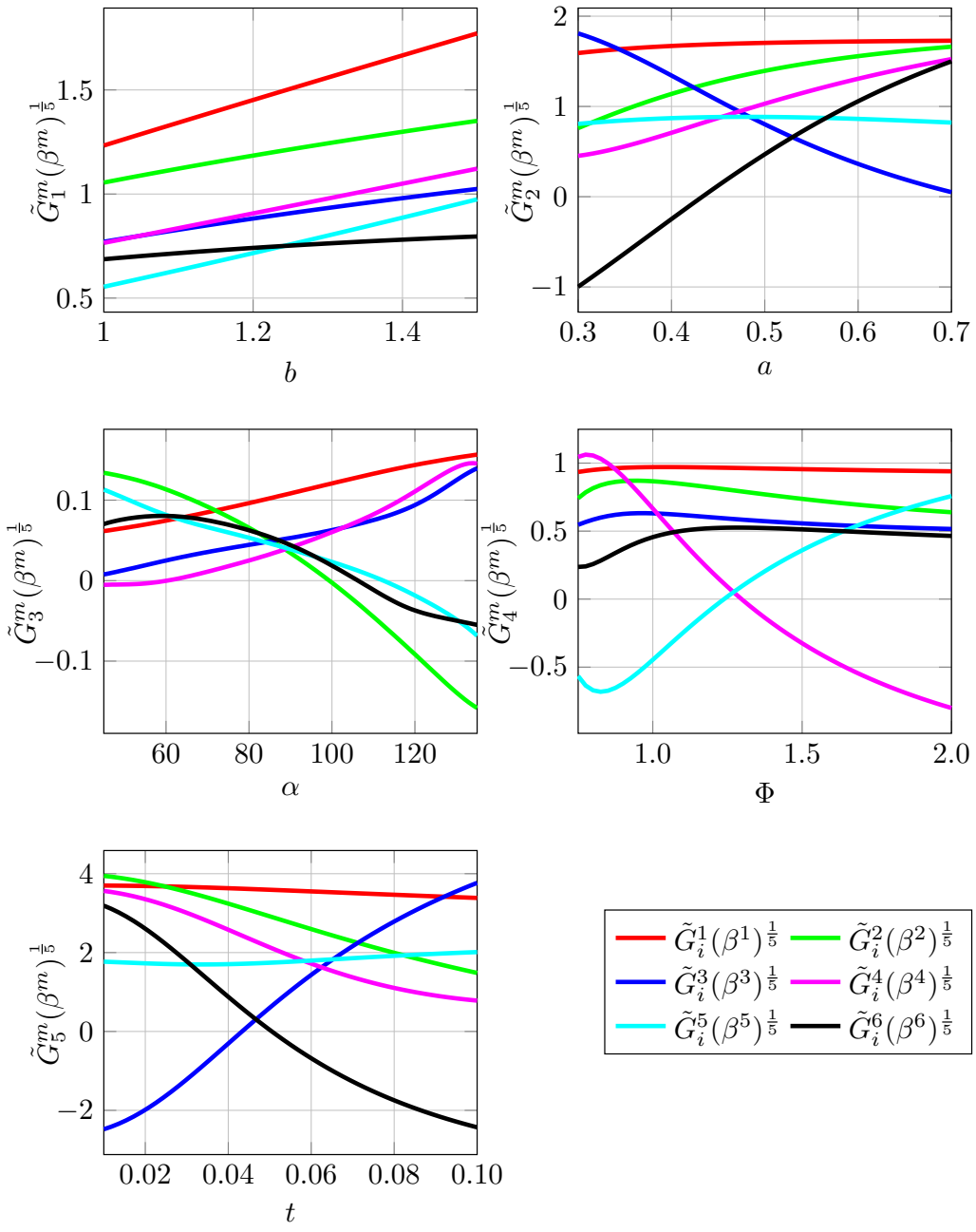
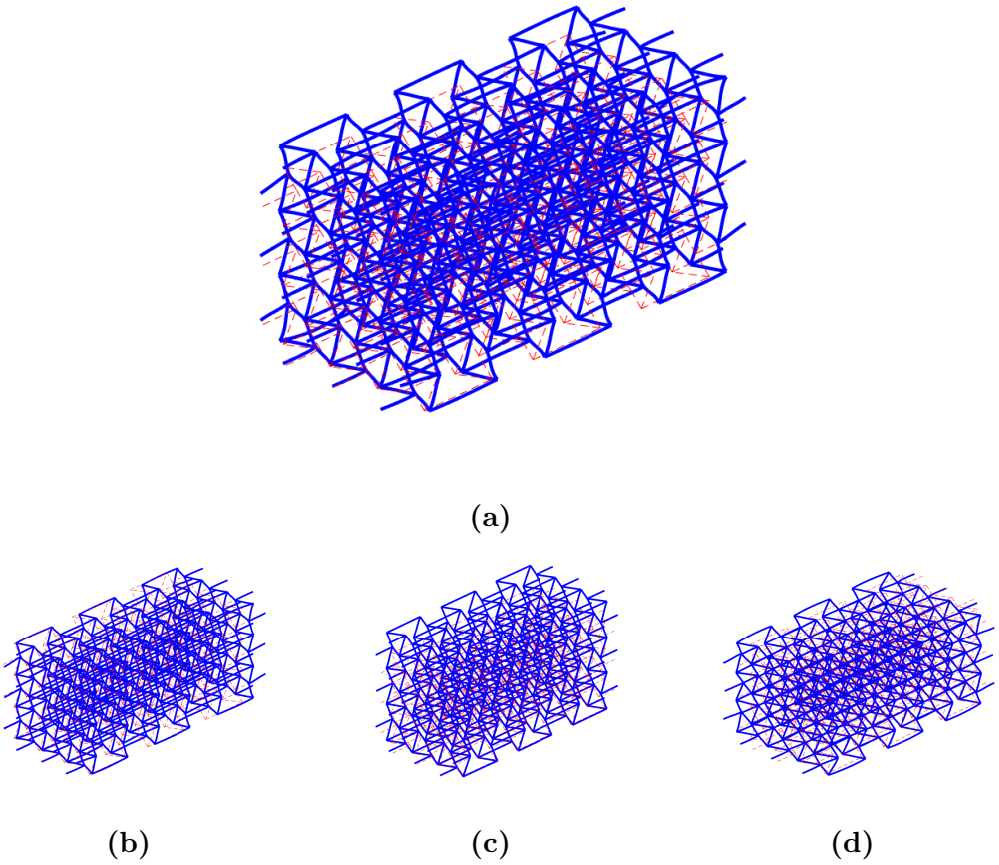


Figure 5.31: Modal functions for case XX.

Repeating the exercise for the case YY:



(a)

(b) **(c)** **(d)**

Figure 5.32: Modal deformations for case YY. **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 2x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 5 (scale 10x).

In figure 5.32, the same behavior as the case XX is observed with respect to the auxetic behavior. In the modal functions it is appreciated that the parameter b has almost no effect in the deformation if the lattice, while the dependance in α is quite strong, having changes of sign passing through 90°, just as it was seen in the cases for the homogenized cell. Parameter Φ also adds to the solution, having some variable behavior depending on the mode.

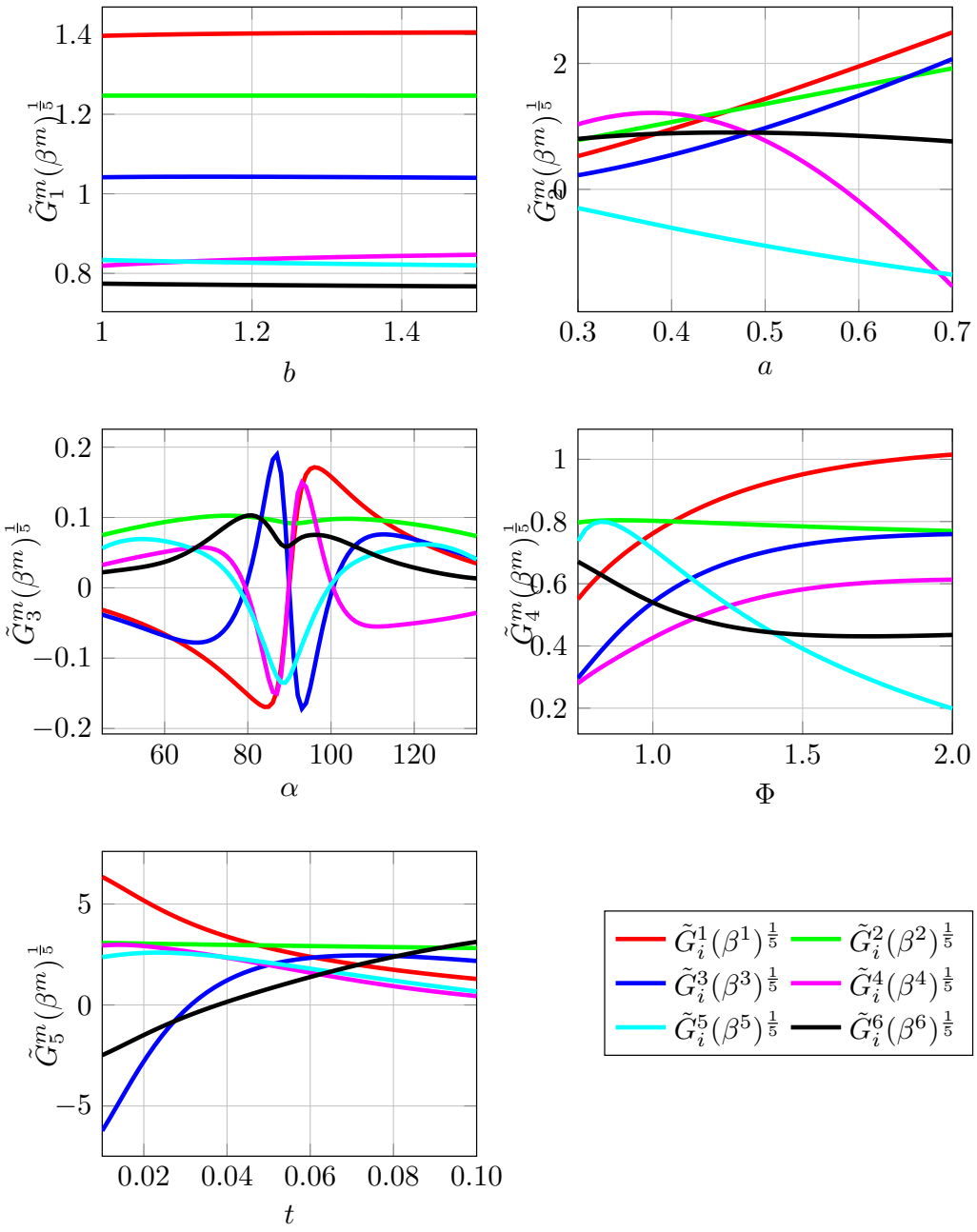
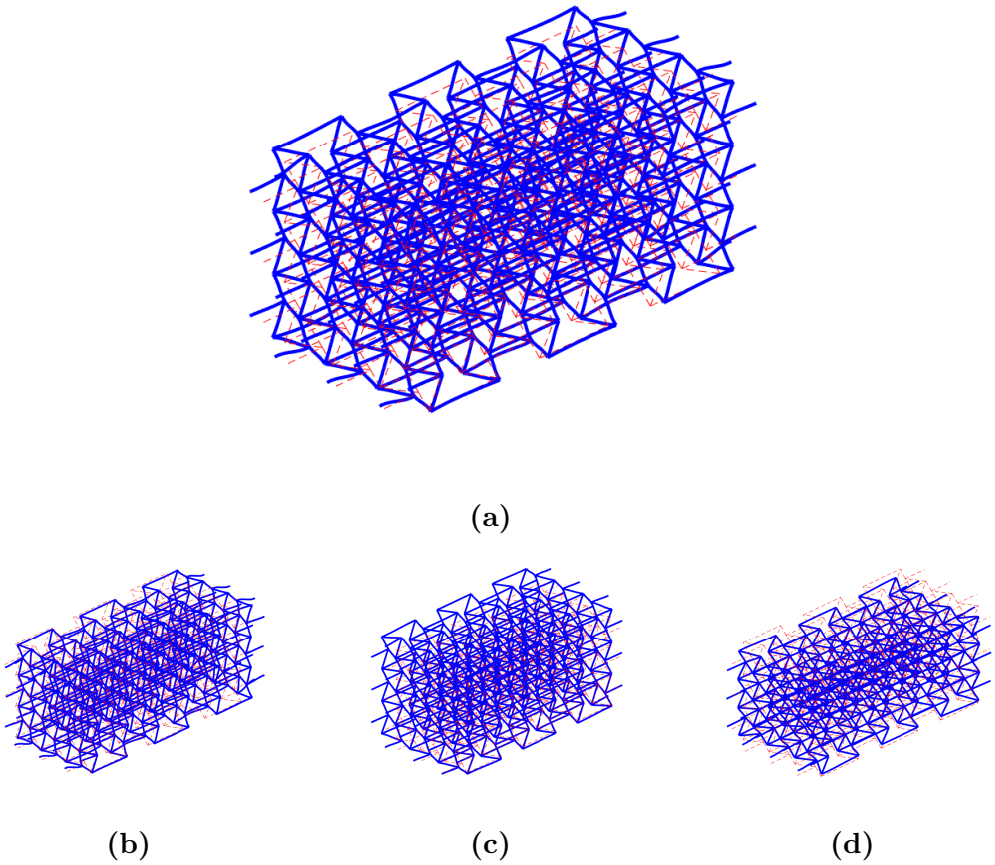


Figure 5.33: Modal functions for case YY.

For case ZZ:



(a)

(b) **(c)** **(d)**

Figure 5.34: Modal deformations for case ZZ. **a)** Deformation for all modes combined. **b)** Deformation for mode 1 (scale 2x). **c)** Deformation for mode 2 (scale 2x). **d)** Deformation for mode 5 (scale 10x).

The modal deformations for ZZ have no apparent difference from the previous cases. Analyzing the modal functions, the relation to α is apparent and even correlates with the homogenized cell case, where there is a little anomaly in one of the modal functions close to the value of $\alpha = 90°$. The dependences from the other parameters are similar and have no major impact in the deformation of the lattice.

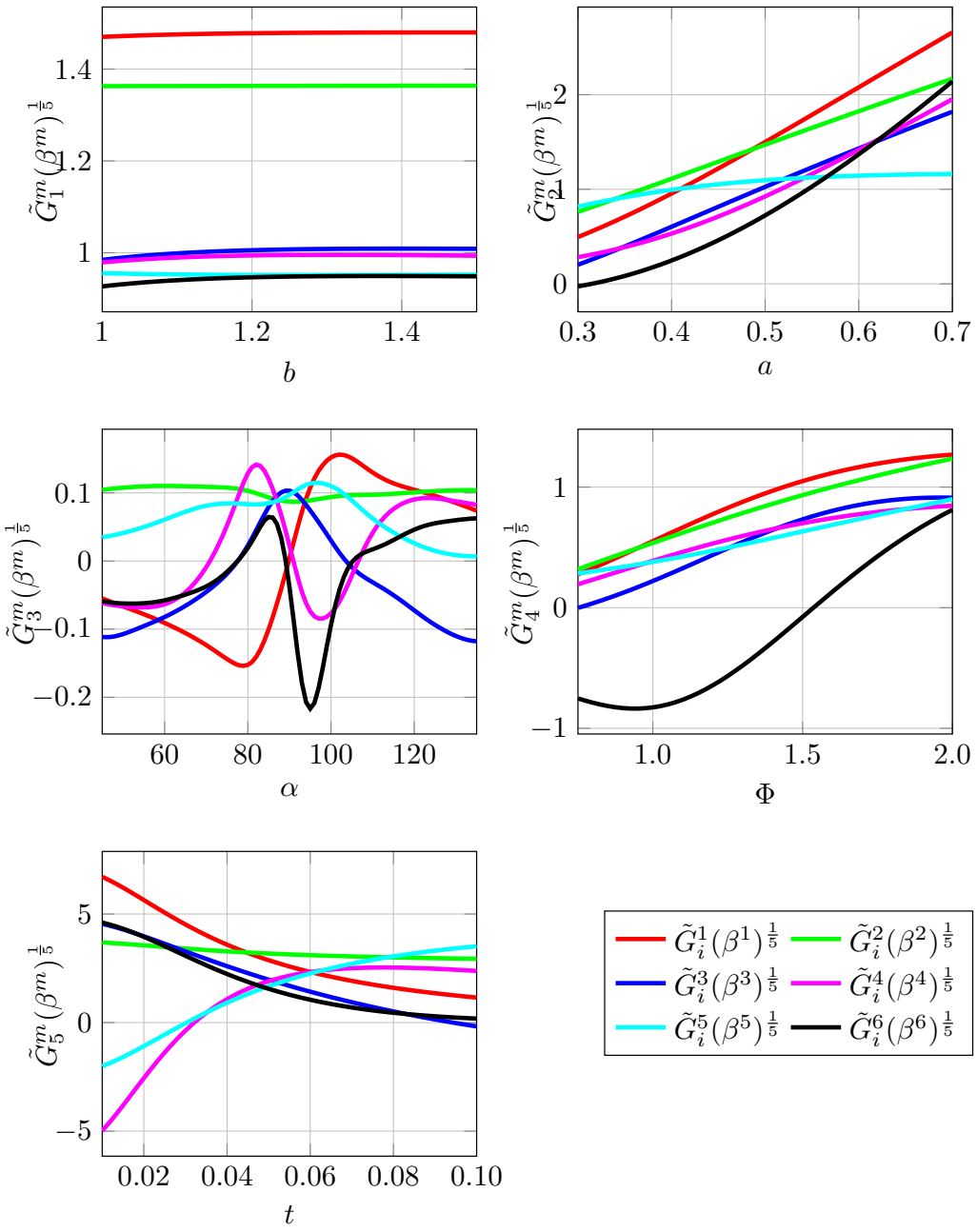


Figure 5.35: Modal functions for case ZZ.

5.2.3 Effective Poisson's values

In figure 5.36, the values of ν computed through an approximation using the strains of the lattice are shown. The main thing to notice is the correlation of the surfaces between both cases. It can be readily seen that the values of the surfaces in both the homogenized case and the full pattern case are not similar. One possible explanation for this can be given from the fact that the full pattern is small, so the boundary effects mentioned previously have a bigger influence in the results. The homogenized case represents the unit cell in a setting of infinite repetition, and this "infinity" can not be reproduced in a $3 \times 3 \times 3$ lattice, since the complete cell appears only once in the center. However, doing a qualitative analysis just by looking at the shape of the surface, a correlation can be made. For all the cases, the shapes of the surfaces follow the same pattern, having the same particularities as the homogenized case. For the cases shown in 5.36a to 5.36d, there is a critical point for $\alpha = 90°$, where the surface changes signs, associated again to the auxetic behavior. For cases 21 and 31 the same jump in the values of ν is appreciated. For the two left cases we have surfaces that yield negative values for all combinations of parameters, which was expected from the homogenized case. As a general trend, differences are accentuated for the parameters on the extreme values of the surface, which could be attributed to the already mentioned boundary effects.

5.2.4 Computed errors

For the case of the structured pattern, the same error cases shown in table 5.1 are studied in order to observe the evolutions. Errors are shown in figures 5.38 to 5.40. As can be seen, errors in the case of the full pattern are much bigger than the ones displayed in the homogenized cell, while the tendencies are kept equal. The error tends to stabilize between 5 and 10 modes, later oscillating around a stable value. For the case of the L_2 error they tend to stay below a 10%, except for the case ZZ, which displays values closer to 20%. The main cause of these differences is the amount of nodes considered in the structure. In the case of the $3 \times 3 \times 3$ pattern, 390 nodes are considered, which yield matrices of 2340×2340. Since boundary effects are relevant now, the way in which boundary conditions are set is essential, due the amount of different ways in which the structure can be restricted. Trying different settings with different nodes restricted give different values of error, so finding a particular combination that works better can help to improve the accuracy. As a way to reduce the error of the computation, reduction of parameters is one of the options, since the amounts of nodes of the system cannot be reduced. The solver tends to give more accurate solutions when it handles less parameters, but also restricts the amount of combinations that can be made, which is one of the advantages of the method.

The analysis for parameters is also done and shown in 5.37. Different to the homogenization case, the dependency on the parameters differs from previous cases, although the tendency is still higher error on the boundaries of the intervals.

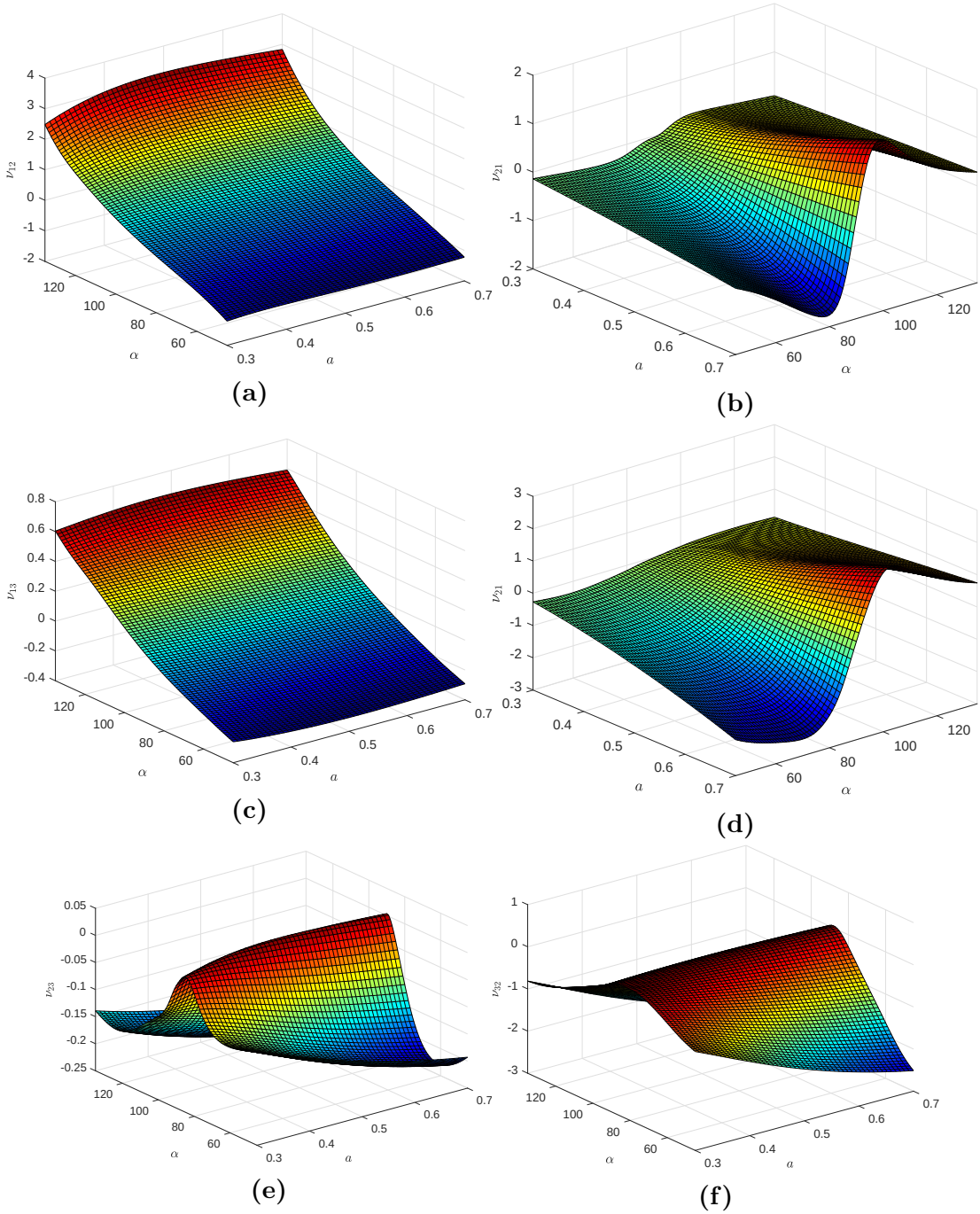


Figure 5.36: Values for different approximated ν. **a)** Values of ν_{12}. **b)** Values of ν_{21}. **c)** Values of ν_{13}. **d)** Values of ν_{31}. **e)** Values of ν_{23}. **f)** Values of ν_{32}.

Also, for cases YY and ZZ there is a strong dependency on α, which presents an oscillating pattern, correlating with the homogenized case. Same as before, case XX error seems to not be exclusively dependent on the α value.

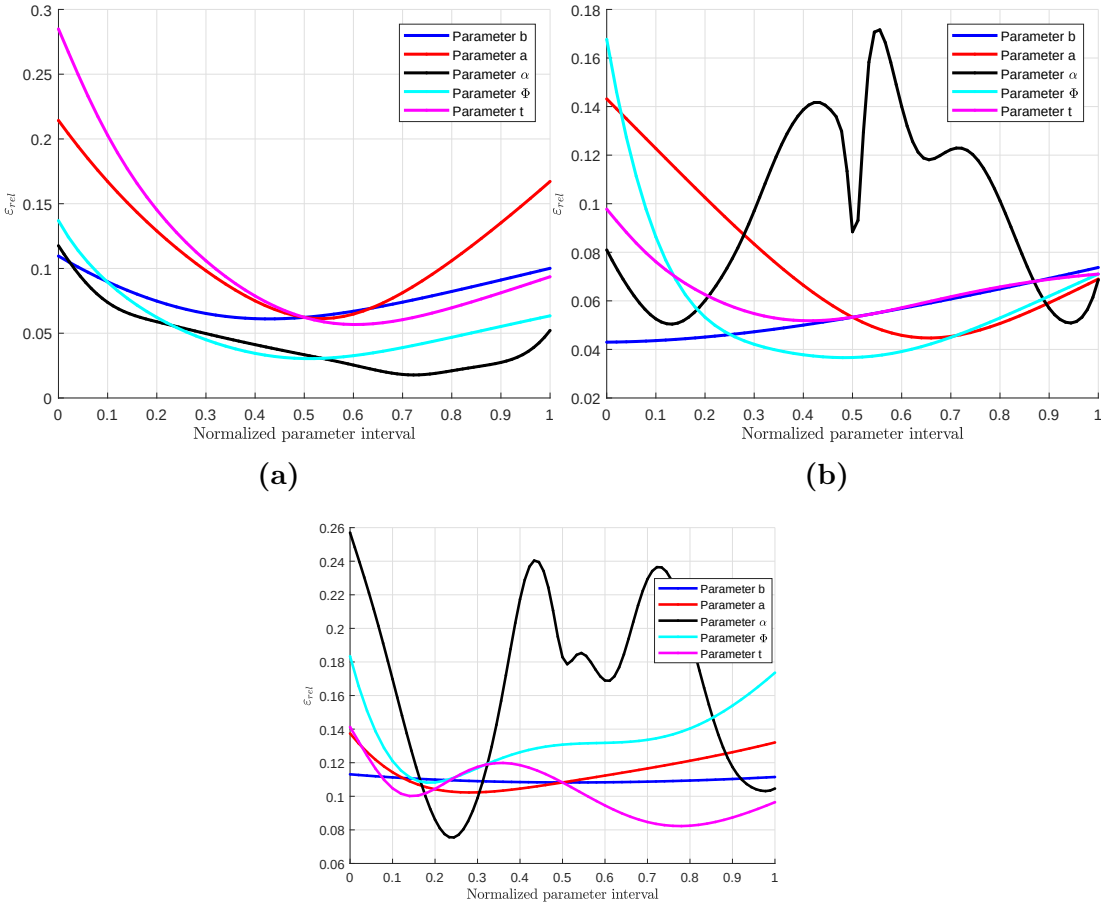


Figure 5.37: Error plots by parameter for different full pattern cases. **a)** Case XX. **b)** Case YY. **c)** Case ZZ.

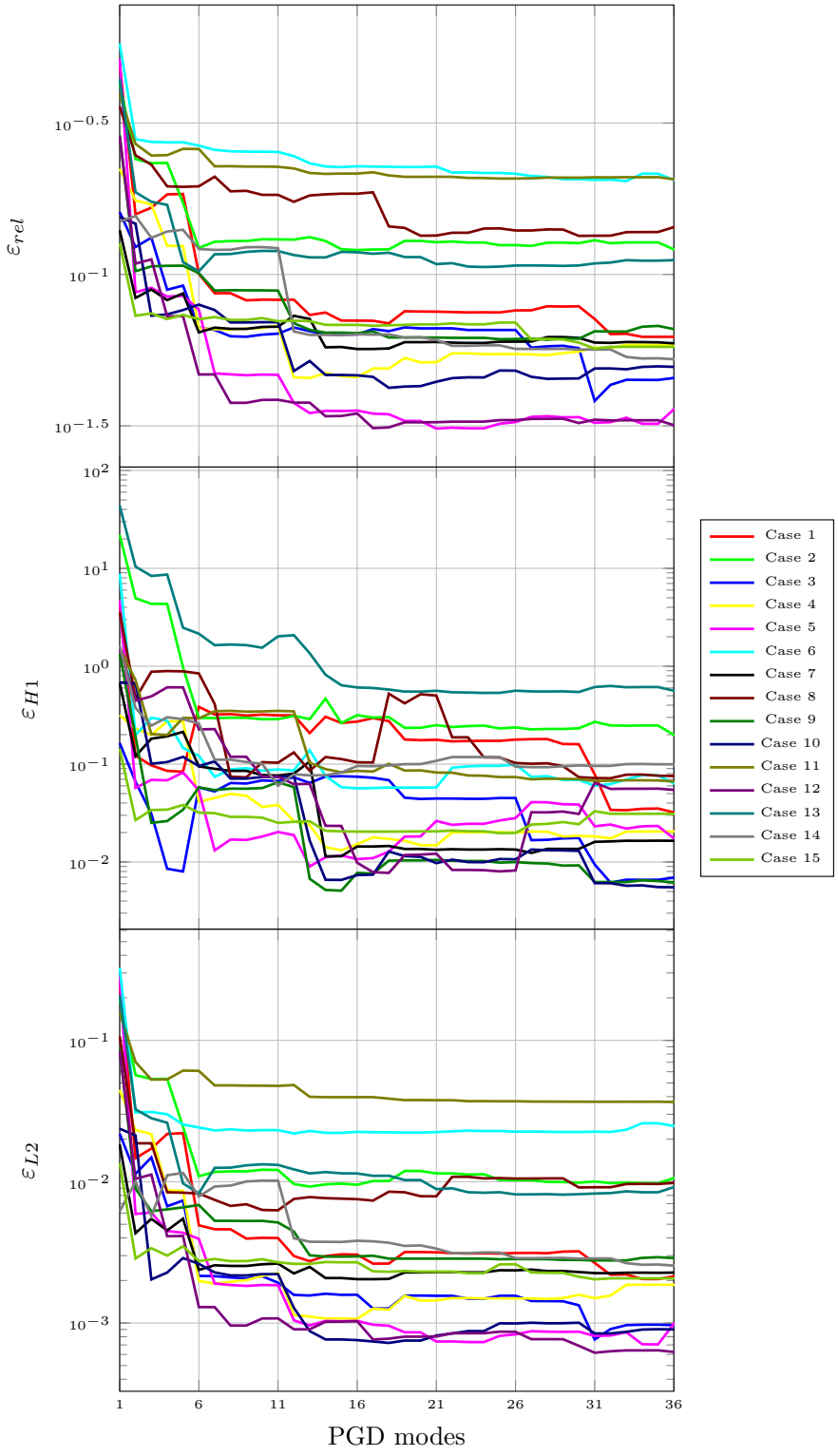


Figure 5.38: Error for all study cases in XX. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

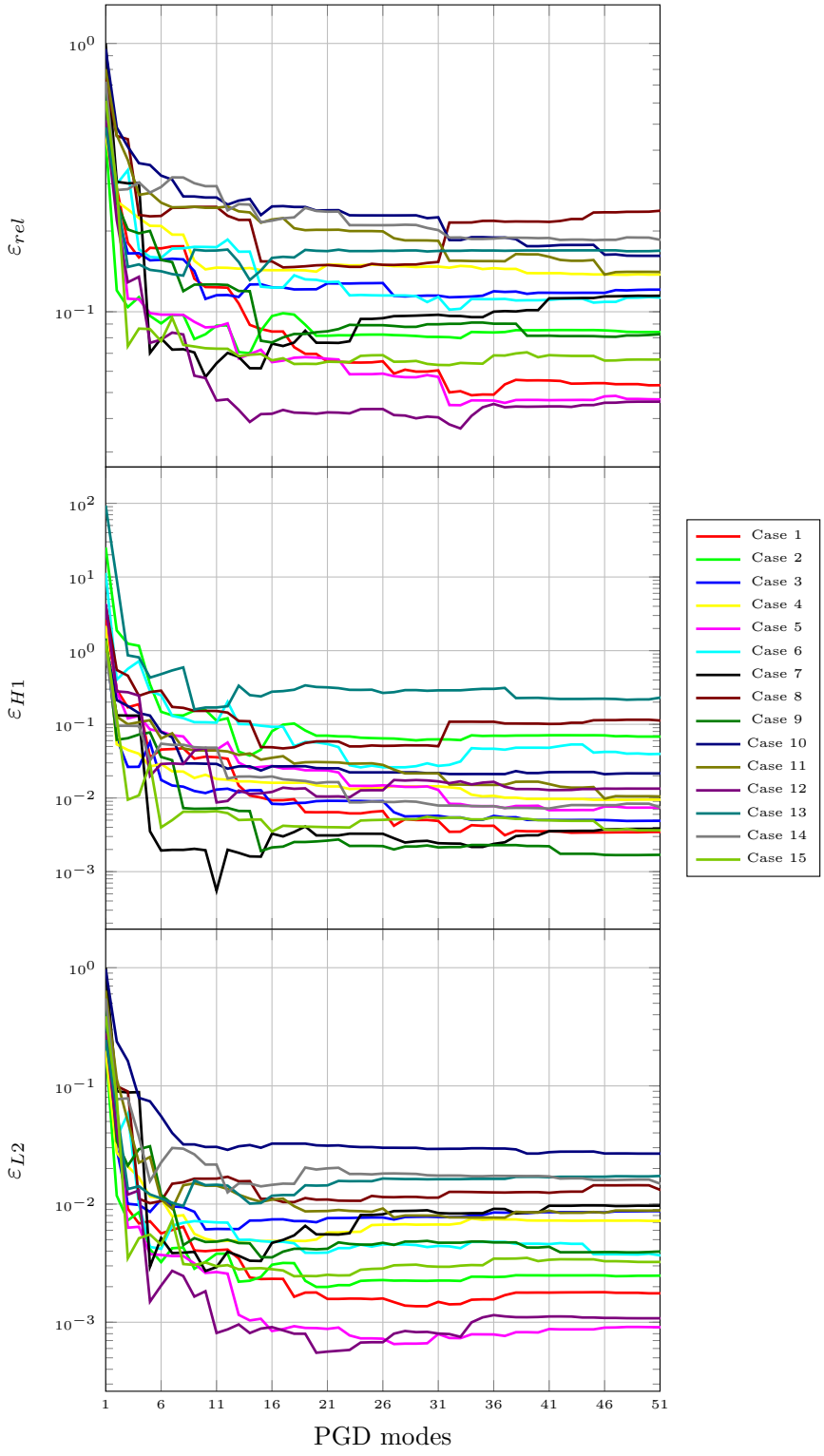


Figure 5.39: Error for all study cases in YY. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

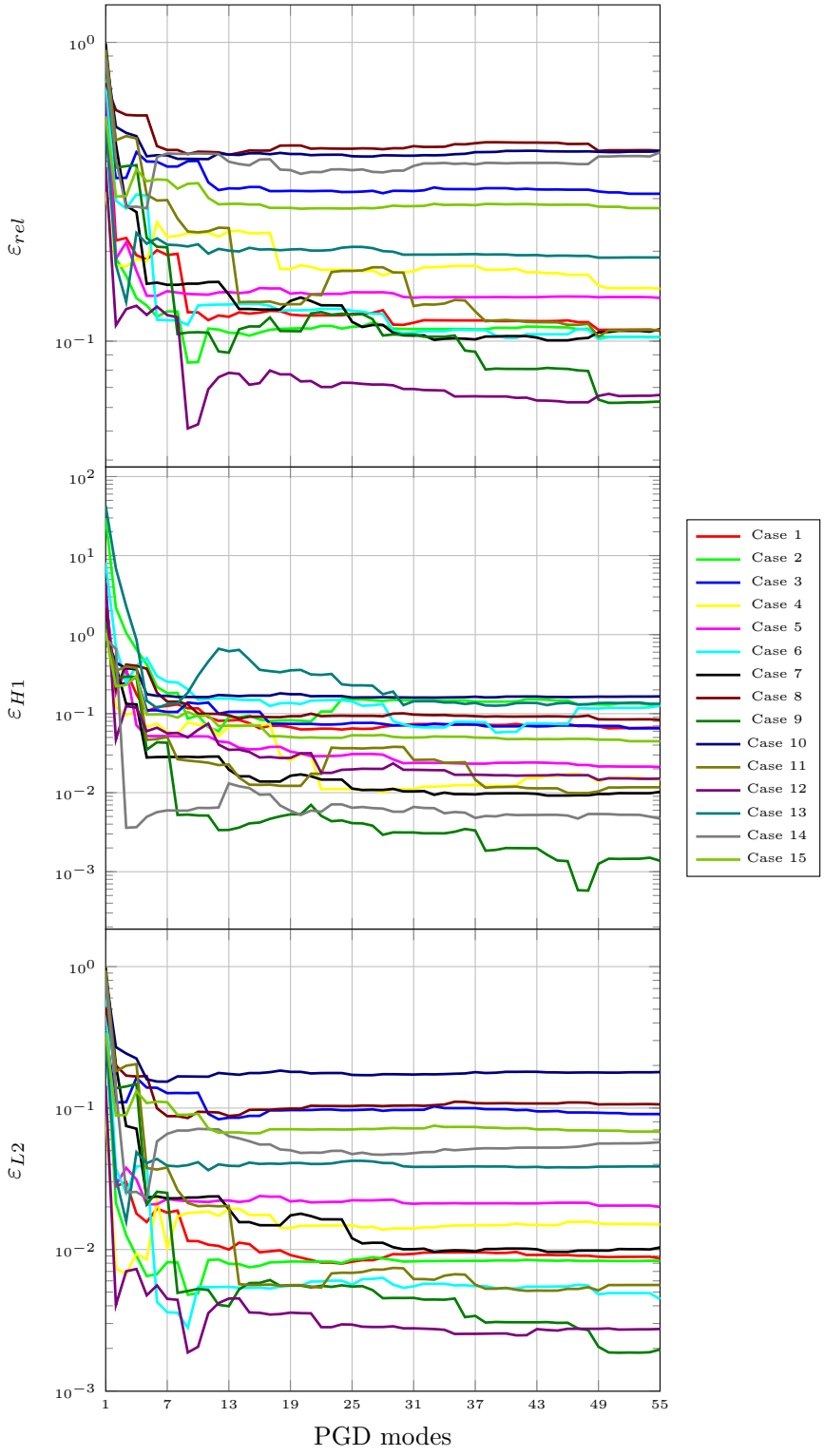


Figure 5.40: Error for all study cases in ZZ. Upper plot: normal relative error. Middle plot: H1 error. Lower plot: L2 error.

5.3 Post-processing tool

In the previous results, the cases could be analyzed from a numerical point of view. However, the graphical part is limited to just one view, since the spatial modes are plotted in a geometry of an arbitrary value of the parameters. In order to show the utility of the post-processing tool, some of its capabilities are displayed in figures 5.41 to 5.45.

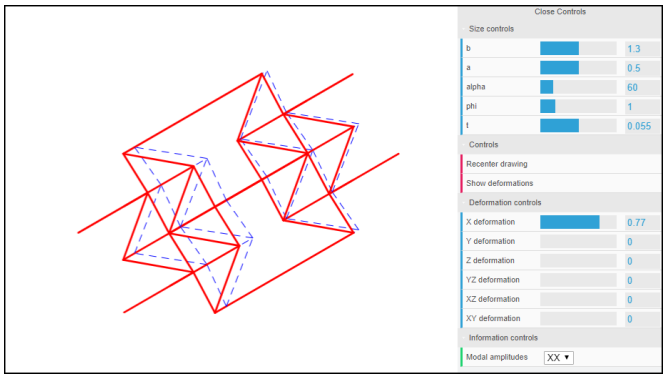


Figure 5.41: Deformation of the unit cell.

Load case XX is shown in the picture, similar to the one presented in figure 5.7a. For a given set of parameters defined in the control bar, the deformation can be adjusted from 0 to 1 to show the strain imposed in the unit cell.

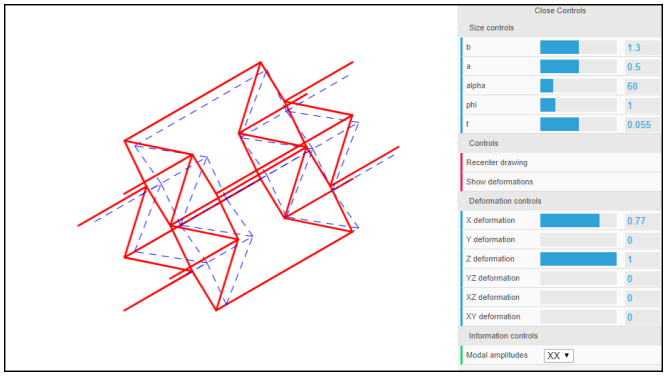


Figure 5.42: Combination of load cases.

The results obtained for the different load cases correspond to isolated cases in the unit cell, thus they are presented as such. However, due to the linear theory considered in the modelling, these deformations can be combined to represent a different case in the unit cell. In figure 5.42, the strains in X and Z are added, showing a biaxial deformation. This combination could be realized in MATLAB, but the advantage of the tool is the simplification of this task, allowing the user to test these combinations as desired.

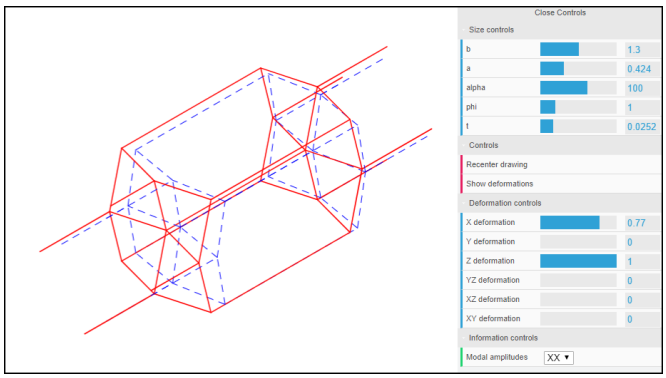


Figure 5.43: Variation of parameters.

Other advantage of the post-processing tool in combination with the PGD method is the analysis for different parameters. In 5.43 the unit cell in the same load state is now changed to a different set of values, showing the new deformation associated to the case.

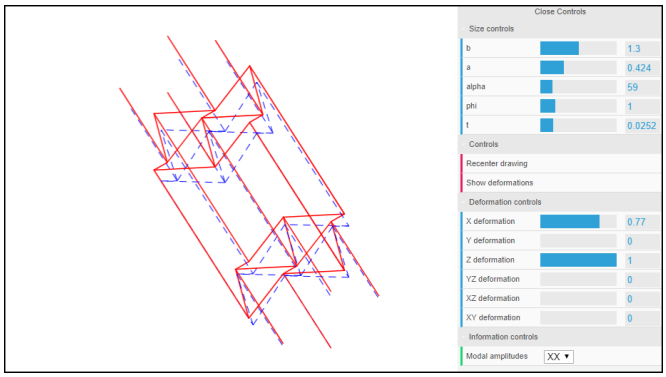


Figure 5.44: Modification of viewpoint.

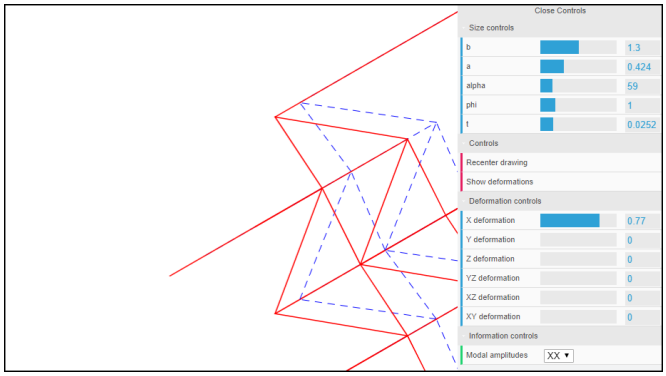


Figure 5.45: Zooming detail of the unit cell.

Finally, in the last two images it is shown the geometry handling abilities of the tool, allowing the user to modify the camera position to display the unit cell from a different angle, as well as zooming, which enables the possibility of watch

closely details in the solution. Some examples are also added for the full pattern. In here, the full graphic interface is shown, where it is also visible the inclusion of the Poisson's ratio surfaces and the modal amplitudes for the three cases.

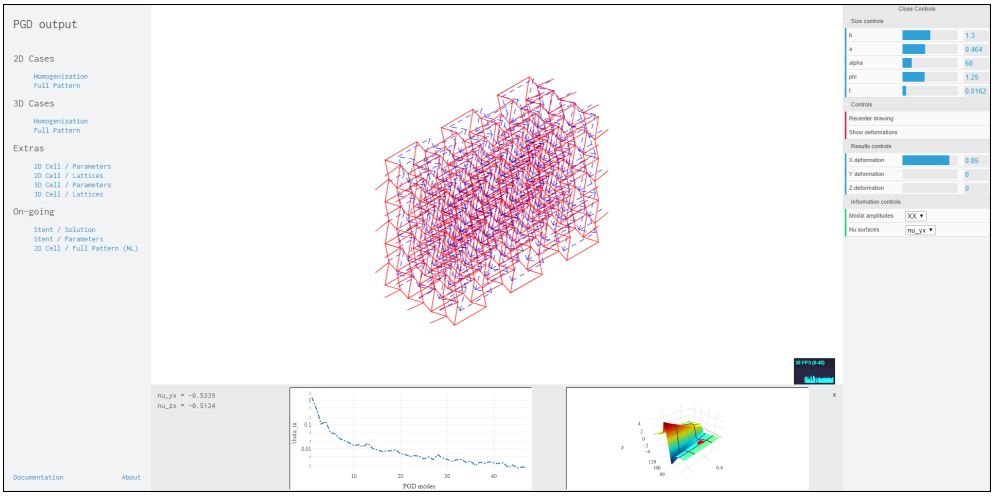


Figure 5.46: Zooming detail of the unit cell.

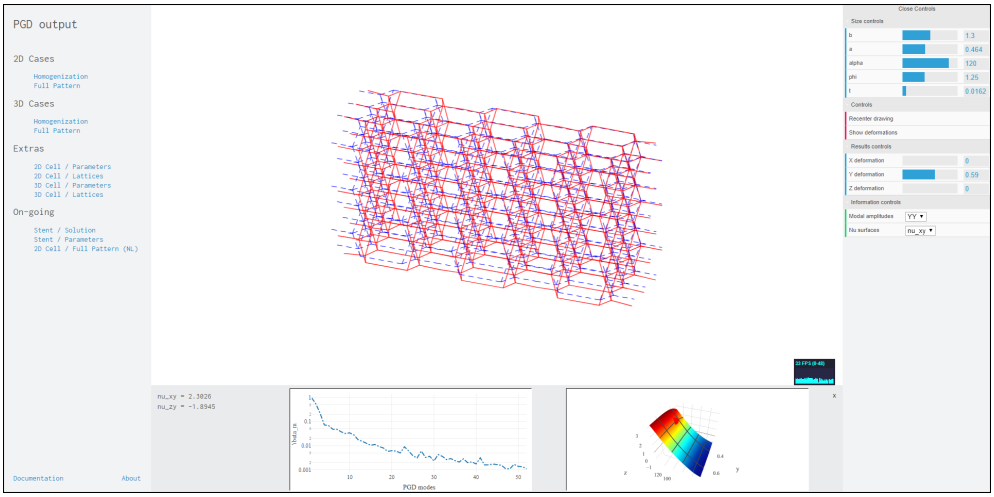


Figure 5.47: Zooming detail of the unit cell.

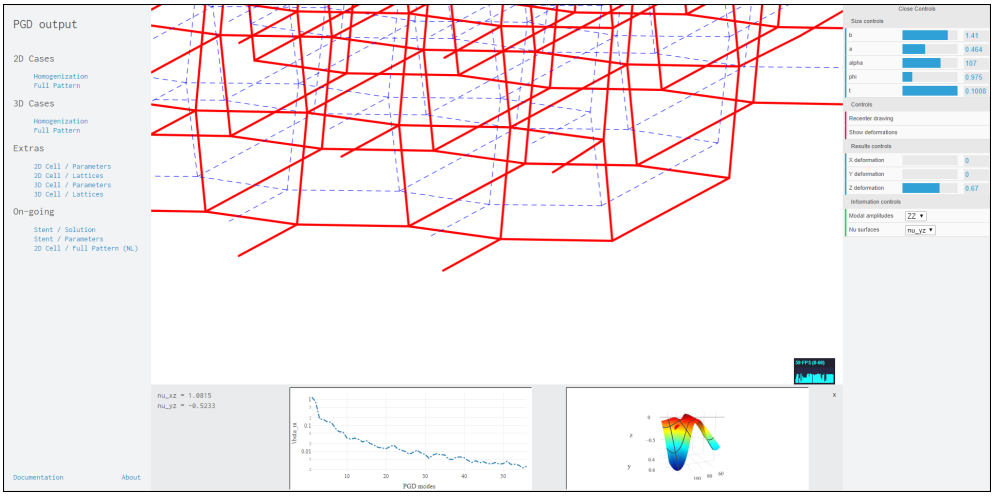


Figure 5.48: Zooming detail of the unit cell.

Chapter 6

Conclusion

In the present work, 2D and 3D parametric PGD modelling of architectural materials, and a web-based graphic application for post-processing and visualization were developed, particularly focunsing on auxetic materials, which were the main focus. First, a lattice structure known as "inverted honeycomb" was presented. A mechanical model was formulated for the solution of the unit cell defining the pattern, based on the linear Euler-Bernoulli beam theory. This was presented in a parametric form in order to be solved by the PGD method. Two cases of study were introduced, one for the unit cell for studying the homogenized problem and one for a full pattern, in order to compare the behavior. A solver by the PGD method to compute the solutions to the proposed cases was presented. In chapter 4, a post-processing tool developed as a web application was introduced. The design and functioning was explained in order to visualize the obtained results. Finally, in chapter 5 the results were shown for both the homogenized and full pattern models. For the homogenized model, the obtained deformations were used to compute an effective constitutive tensor, which allowed us to calculate the effective Poisson's coefficients. In the case of the full pattern, the same study was done using an approach based on the strains, where an approximate value of the Poisson's ratio was obtained and compared to the homogenized case.

6.1 Concluding remarks

After analyzing and solving the problem, some conclusions can be extracted from this work. The deformation of the unit cell and the full pattern were in agreement with the 2D model, and the same auxetic properties were observed. Homogenization allowed to obtain an approximation of the behavior of a cell inside an infinite lattice. Effective values of the Poisson's ratio were extracted for the relation between the fundamental parameters a and α. It can be seen from these images that the auxetic properties of the unit cell are not always present and highly depend on the α parameter, which was expected since for angles higher than $90°$ the cell has no longer the expanding mechanism expected for smaller angles. The solutions are in agreement wih those obtained in the previous work [16], where the homogenization of a 2D cell and a pattern of similar characteristics was studied.

It was shown that the accuracy of the PGD method is directly dependent on the amount of modes computed for it. Due to the particular form of the PGD approximation, every modal solution in a particular case does not represent a physical solution to the problem. However, the combination of the modes are a good approximated solution to the parametrized problem, which is why the accuracy improves. From what it was seen, the first modes are the ones responsible for the majority of the solution, while the lower modes correspond to a refinement. In the case of the unit cell, the errors, computed against the solution by the FEM method are within an acceptable range, lower than 4%. For the case of the full pattern, the obtained results showed a bigger error, but still in an acceptable order of magnitude (of about 10%).

The development of the post-processing tool has been a support for the understanding of the solutions presented in this work. PGD has proven that is an efficient method for solving parametric problems, but it is overshadowed by its post-processing, due to the complications of combining matrices and vectors. In that sense, the tool has allowed us to import the *vademecum* of results into a user-friendly graphic interface, which enabled complete visualization of the problem for all the range of the chosen parameters. Solutions were made observable for every case in different views, as well as the ability to combine them and visualize the numerical values of the effective constitutive tensor and Poisson's ratios, all of this in real-time.

The web application fashion in which the post-processing tool is programmed is the novelty of the work, along with the 3D PGD modelling. By making the application available online, every person has access to it and can use it to verify the conditions of our unit cell and pattern. The interface is developed to be as intuitive as possible, with the goal of having a tool that could be used by anyone, even without previous experience in the field. Due to the capability of real-time processing, it can be used to aid in the design of an auxetic material, helping to obtain desired properties in an easy way. It has also the educative value for inexperienced users, who can now understand about the nature of auxetics without the need of lengthy computation. Nevertheless, the tool has some drawbacks. The WebGL tool is not optimized to run in mobile devices, causing delays in the rendering of images. Also, due to the handling of memory and processing power, for bigger problems the image might lag, even on powerful computers.

A big limitation to this work is also the fact that the modelling is done using linear elements, which limits the analysis greatly due to the properties of the materials. For big deformations, a non-linear model is required to study the deformations, specially if materials with viscoelastic properties were considered.

6.2 Future work

Besides the performed work, more study cases can be developed and different geometries can be tried for further studies. Patterns like auxetic stents can be formulated from 2D unit cells arranged in a cylindrical pattern. These patterns have already been studied and some results have been provided, but there is still potential to develop from there. Furthermore, different geometries besides the "inverted honeycomb" can be analyzed. Lattice work can also be improved. Due to the limitations of computing, only small patterns were analyzed, and the effects of the homogenization should be better appreciated for bigger periodicities.

Another extension to the work would be the developing of the non-linear model. The linear formulation works as a good approximation for small displacements, but fails to represent the behavior for bigger deformations. Implementing the non-linear model would be more time consuming than the cases solved in this work, but they can provide more accurate information of the lattice.

An interesting line to be developed in future proceedings is the link between model and reality. For the 3D model it would be beneficial to develop the real structure in avia 3D printing, in order to test the solutions against a real model.

In the post-processing tools there are many opportunities for further development. One of the considered improvements is the automatization of the loading of results, which would benefit users who have their own solutions to a PGD problem. Implementations can be made in order to generate better imaging, using textures to ease the description of materials, as well as making differences between properties.

Bibliography

[1] J. Schwerdtfeger, F. Wein, G. Leugering, R.F. Singer, C. Körner, M. Stingl, and F. Schury. Design of auxetic structures via mathematical optimization. *Advanced Materials*, 23:2650–2654, 2011.

[2] T.A. Schaedler and W.B. Carter. Architected cellular materials. *Annual Reviews of Materials Research*, 46:187–210, 2016.

[3] H. Mitschke, J. Schwerdtfeger, F. Schury, M. Stingl, C. Körner, R.F. Singer, V. Robins, K. Mecke, and G.E. Schröder-Turk. Finding auxetic frameworks in periodic tessellations. *Advanced Materials*, 23:2669–2674, 2011.

[4] J. Christensen, M. Kadic, M. Wegener, and O. Kraft. Vibrant times for mechanical metamaterials. *MRS Communications*, 5:453–462, 2015.

[5] Y. Liu and H. Hu. A review on auxetic structures and polymeric materials. *Scientific Research and Essays*, 5:1052–1063, 2010.

[6] A. Anthoine. Derivation of the in-plane elastic characteristics of masonry through homogenization theory. *International Journal of Solid Structures*, 32:137–163, 1995.

[7] Q. Liu. Literature review: Materials with negative poisson's ratios and potential applications to aerospace and defence. Defence Science and Technology Organisation, Departament of Defence, Australian Goverment, 2006.

[8] T. Allen, N. Martinello, D. Zampieri, T. Hewage, T. Senior, L. Foster, and A. Alderson. Auxetic foams for sport safety applications. *Procedia Engineering*, 112:104–109, 2015.

[9] A. Díaz, A. Muslija, and J.P. García-Ruíz. Auxetic tissue engineering scaffolds with nanometric features and resonances in the megahertz range. *Smart Materials and Structures*, 24, 2015.

[10] J.W. Lee, P. Soman, J.H. Park, S. Chen, and D. Cho. A tubular biomaterial construct exhibiting a negative poisson's ratio. *PLOS One*, 11, 2016.

[11] F. Chinesta, A. Ammar, A. Leygue, and R. Keunings. An overview of the proper generalized decomposition with applications in computational rheology. *Journal of Non-Newtonian Fluid Mechanics*, 166:578–592, 2011.

[12] L.Feng. Review of model order reduction methods for numerical simulation of nonlinear circuits. *Applied Mathematics and Computation*, 167:576–591, 2005.

[13] F. Terragni. Reduced order modeling applications: Introduction and preliminaries. ECMI, Universidad Carlos III de Madrid, July 2013.

[14] A. Ammar. The proper generalized decomposition: a powerful tool for model reduction. *International Journal of Material Forming*, 3:89–102, 2010.

[15] D. Lucia, P. Beran, and W. Silva. Reduced-order modeling: new approaches for computational physics. *Progress in Aerospace Sciences*, 40:51–117, 2004.

[16] A. Sibileau, A. García-González, F. Auricchio, S. Morganti, and P. Díez. Explicit parametric solutions of lattice structures with proper generalized decomposition (pgd). *Computational Mechanics*, 61:376–380, 2018.

[17] S.M. Han, H. Benaroya, and T. Wei. Dynamics of transversely vibrating beams using four engineering theories. *Journal of Sound and Vibration*, 225:935–988, 1999.

[18] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley and Sons, Inc., 4 edition, 2002.

[19] Module 3: Constitutive equations. `http://web.mit.edu/16.20/homepage/3_Constitutive/Constitutive_files/module_3_with_solutions.pdf`. Accessed: 2018-06-12.

[20] S. Li and A. Wongsto. Unit cells for micromechanical analyses of particle-reinforced composites. *Mechanics of Materials*, 36:543–572, 2004.

[21] E. Cueto, D. González, and I. Alfaro. *Proper Generalized Decompositions: An Introduction to Computer Implementation with Matlab*. Springer, 1 edition, 2016.

[22] Pgdplots github repository. `https://github.com/grvaldes/pgdplots`. Accessed: 2018-09-11.

[23] Three.js documentation. `https://threejs.org/docs/index.html`. Accessed: 2018-05-30.

[24] L. Tamellini and A. Nouy. Proper generalized decomposition for linear and non-linear stochastic models. Radon Special Semester, WS on Multiscale Simulation and Analysis in Energy and the Environment, Linz, December 2011.

[25] G.N. Greaves, A.L. Greer, R.S. Lakes, and T. Rouxel. Poisson's ratio and modern materials. *Nature Materials*, 10:823–837, 2011.

[26] J. Greer. Materials by design: Using architecture and nanomaterial size effects to attain unexplored properties. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2015 Symposium*, 2015.

[27] J.E. Cadman, S. Zhou, Y. Chen, and Q. Li. On design of multi-functional microstructural materials. *Journal of Materials Science*, 48:51–66, 2013.

[28] K. Theerakittayakorn and P. Nanakorn. Periodic boundary conditions for unit cells of periodic cellular solids in the determination of effective properties using beam elements. In *Advances in Structural Engineering and Mechanics*, Jeju, Korea, 2013.

[29] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluids*, 139:153–176, 2006.

[30] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, part ii: Transient simulation using space-time separated representations. *Journal of Non-Newtonian Fluids*, 144:98–121, 2007.

[31] C.A. Felippa. Introduction to finite element methods. Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado, 2004.

[32] E. Onate, P. Díez, F. Zárate, and A. Larese. Introduction to the finite element method. Master of Sciente in Computational Mechanics, Universitat Politècnica de Catalunya, 2008.

[33] The benefits of web-based applications. `https://www.magicwebsolutions.co.uk/blog/the-benefits-of-web-based-applications.htm`. Accessed: 2018-05-21.

[34] Chapter 4: Singular value decomposition. `https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf`. Accessed: 2018-06-12.

Appendix A

Singular Value Decomposition

For the solution of the 3D unit cell problem, a restructure of the parameters was made in order to reduce redundant variabes. In this regard, angle θ was presented as a function of angle α and the amplification factor Φ. Considering the previous rearrangement of parameters, the function $\cos\theta$ results to be separable in terms of the variables α and Φ. However, for $\sin\theta$ there is no separable expression. In order to overcome this issue, and given that the function depends only on two parameters, the natural option in this cases is to use the Singular Value Decomposition (SVD) technique to obtain a separated representation of this function in terms of the parameters α and Φ [34]. Therefore, $\sin\theta$ is expressed as a sum of products of separated functions of Φ and α, affected by a modal amplitude. In this appendix, the method is briefly explained.

A.1 Parameter description

In section 2.2.2, it was introduced the parameter Φ, that allowed to replace the value of c (see figure 2.5). This was described by equation 2.2:

$$a\cos\alpha = c\cos\theta \tag{A.1}$$

$$a\cos\alpha = \Phi a\cos\theta \tag{A.2}$$

$$\cos\theta = \frac{\cos\alpha}{\Phi}. \tag{A.3}$$

Using the trigonometrical identity:

$$\cos^2\theta + \sin^2\theta = 1 \tag{A.4}$$

$$\sin^2\theta = 1 - \cos^2\theta \tag{A.5}$$

$$\sin\theta = \pm\sqrt{1-\cos^2\theta} \tag{A.6}$$

$$\sin\theta = \pm\sqrt{1-\frac{\cos^2\alpha}{\Phi^2}}, \tag{A.7}$$

which is a function that is dependent on both α and Φ, but with no separation available. Since the intervals for α and Φ are defined, it is possible to assure that $\sin\theta$ has only positive values within that range. Therefore, we take the positive sign from of equation A.7 and evaluate $f(\alpha, \Phi) = \sin\theta$, which results numerically in a matrix. The rows of the matrix are taken as the values of the angle α, where the columns represent the values of Φ, between the ranges defined for them ($\alpha \in [45, \ldots, 135]$ in 91 steps and $\Phi \in [0.75, 2.00]$ in 51 steps). The dimension of the matrix is then $m = 91 \times n = 51$, written as

$$f(\alpha, \Phi) = \begin{bmatrix} f(45, 0.75) & f(45, 0.775) & \cdots & f(45, 2.00) \\ f(46, 0.75) & f(46, 0.775) & \cdots & f(46, 2.00) \\ \vdots & \vdots & \vdots & \vdots \\ f(135, 0.75) & f(135, 0.775) & \cdots & f(135, 2.00) \end{bmatrix}$$

The matrix is decomposed using the SVD method, as presented in the following section.

A.2 Methodology

Singular Value Decomposition [34] corresponds to a factorization performed on a matrix. Using the factorization, any matrix of size $m \times n$ can be expressed as

$$\boldsymbol{A} = \boldsymbol{U} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{V}^*, \tag{A.8}$$

where $\boldsymbol{U}$ is an $m \times m$ matrix, $\boldsymbol{\Sigma}$ is a diagonal matrix of size $m \times n$ and $\boldsymbol{V}^*$ one of size $n \times n$. In figure A.1 a graphical representation of the method is shown.

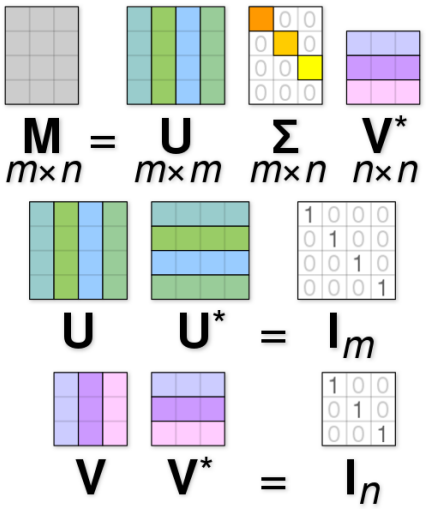


Figure A.1: Schematics of the SVD (Source: Wiki Commons).

The particularity of the decomposition lies in the fact that the elements of matrix $\boldsymbol{\Sigma}$ correspond to the singular values of the matrix, ordered from biggest

to smallest, which can be treated analogously to the PGD, with the particularity that for two parameters, the SVD provides the optimal separated representation. The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are each one a set of orthonormal eigenvectors, which means that they represent a basis on its own. Since each parameter in the approximation is related to one of the dimensions of the matrix, each outer product of $\boldsymbol{U}$ columns against $\boldsymbol{V}$ columns, multiplied by the corresponding singular value in $\boldsymbol{\Sigma}$ represents a weighted mode of the original matrix. The decomposition will have as many modes as linear independent rows has the original matrix, which can be ordered from the highest to the lowest weights. A truncation can be made selecting the amount of modes to consider, with the corresponding error associated to this operation.

A.3 Application

The method was applied in order to approximate the term $\sin\theta$ in a separated way (also referred to as an *affine decomposition*). A relative error between the evaluated function and the SVD approximation is computed, and the maximum is taken expressed as

$$\varepsilon = \max \left| \frac{||f(\alpha, \Phi) - f_{\texttt{SVD}}(\alpha, \Phi)||}{||f(\alpha, \Phi)||} \right|.$$

The values of the matrix $\boldsymbol{\Sigma}$ for the first seven terms are presented in the next table:

Table A.1: Singular Values for the decomposition of $\sin\theta$.

S_i
63.6573
3.1066
0.1100
0.0078
0.0006
0.0001
0.0000

Here, it can be seen in table A.1 how the first mode takes the biggest part of the approximation. In this work, 7 modes were chosen, since they represent the function with an acceptable error lower that 10^{-7}. The function can be seen in figure A.2.

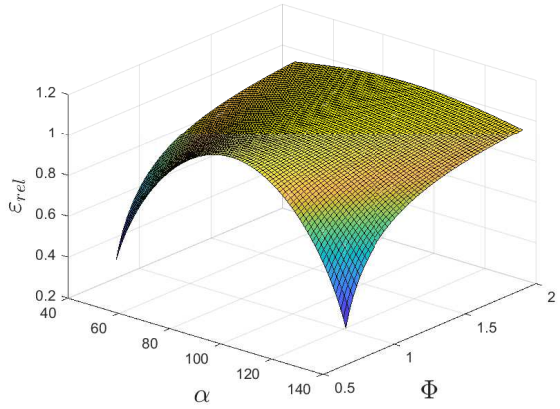
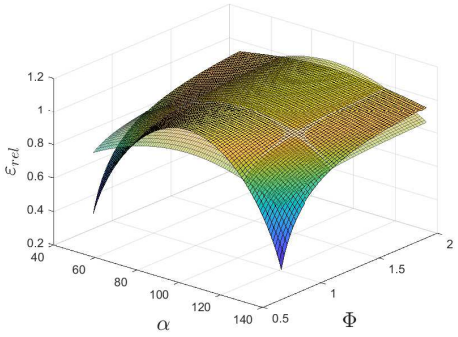
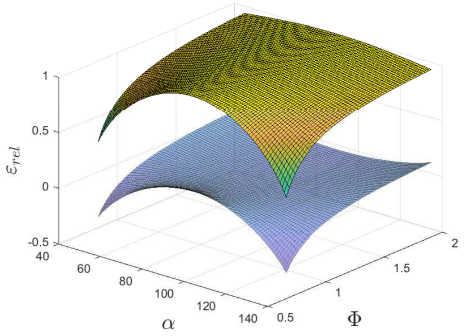


Figure A.2: Complete approximation with seven modes.

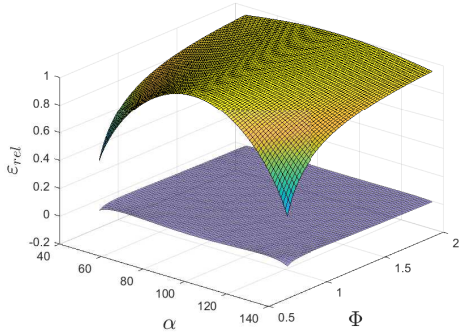
Both the function and each of the first three modes of the approximation are plotted together in order to show, in a graphic manner, the proximity between the values and their approximation. The first three modes are shown in A.3.



(a)



(b)



(c)

Figure A.3: First three modes used in the approximation.

Each mode is not necessarily close to the analytical function, but the sum of terms approximates to an optimal degree. Computing the maximum relative

error with the previous expression (i.e., maximum value of surface described in A.4) yields a value of $9.5558 \cdot 10^{-9}$. The surface for the error in the Φ-α is shown in figure A.4:

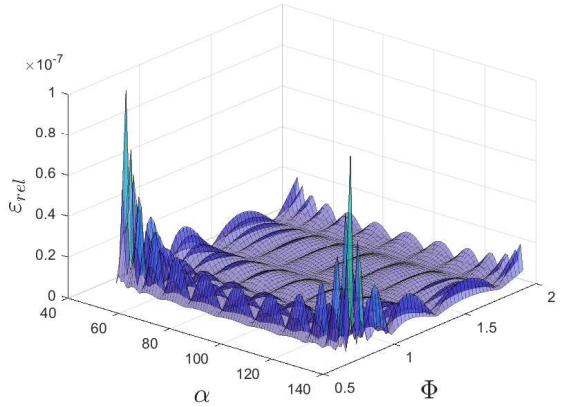


Figure A.4: Surface error for the SVD approximation.

In the figure it can be appreciated that the error is concentrated mostly on the lower end of the Φ interval, coinciding with the extrema of the α interval, where the error reaches values of the order of 10^{-7}. This introduces a source of error which is added to the error provided by the PGD solver itself. This means that before starting the PGD solution, the problem is already approximated in order to have the affine decomposition of the stiffness matrix described in equation 3.5, which can be seen in figure A.5. In there it is observed that for some of the cases, particularly the case ZZ of the homogenized solution, there is an influence on the approximation in the shape of the error surface in the $\Phi - \alpha$ space. However, it can be seen from the other cases that there is error present in other parts of the space, sometimes bigger than the one in the conflicting parts, which can lead to the conclusion that, even if SVD affects the computation, it may not be always the domination error source in the PGD approximation, at least when 7 modes are selected.

It is important to state that the total numerical scheme error is a sum of different effects. As was mentioned, it will be based firstly in the error provided by the SVD separable approximation to the stiffness matrix of the problem, and secondly by the PGD solver of the parametric problem itself. One important detail to observe in all the cases of the homogenized solution shown in figure A.5 is the fact that the error is bigger in the boundaries of the parameter space $\Phi - \alpha$, which implies that a PGD solution evaluated at values in this boundaries can have an error bigger than average.

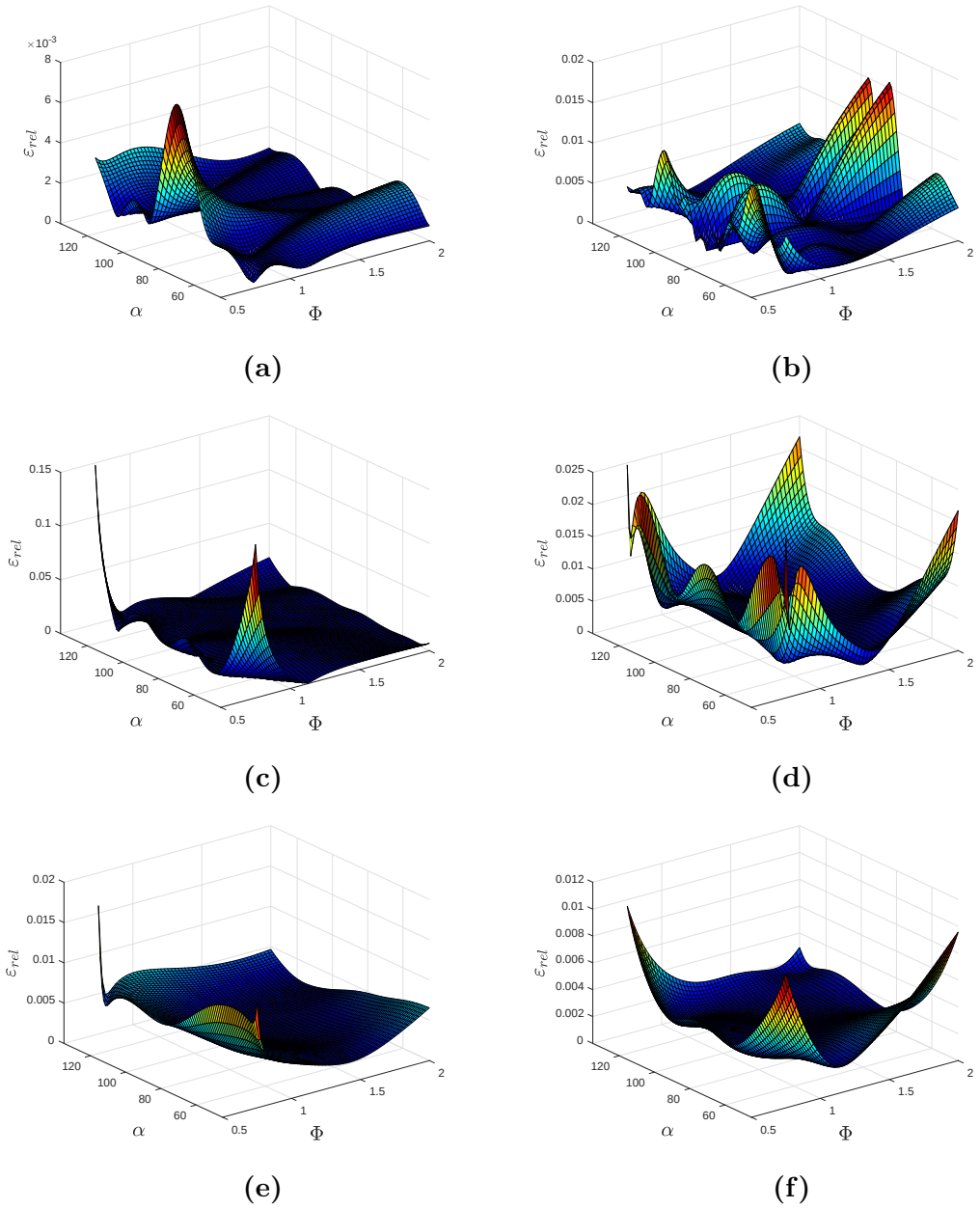


Figure A.5: Error surfaces for the six homogenization cases considering 7 SVD modes. **a)** Case XX. **a)** Case YY. **a)** Case ZZ. **a)** Case YZ. **a)** Case XZ. **a)** Case XY.

Infuence of number of SVD modes

A small error analysis is performed using random values for all the parameters, increasing the amount of modes for the approximation, in order to show the effect in the solution. This can be appreciated in figure A.6:

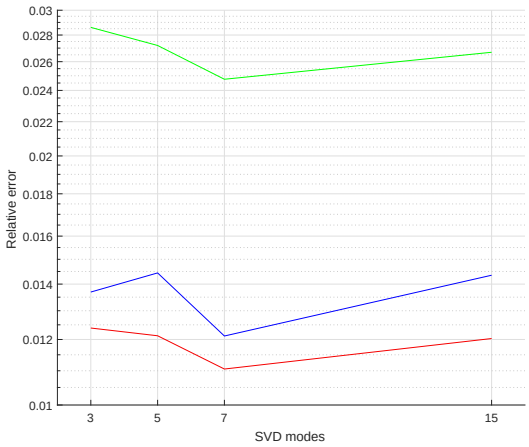


Figure A.6: Error progression for higher amount of modes.

As can be seen from A.6, for the three cases included the optimal amount of modes to use is seven, which is the reason why it was used in this work. This could have to do with the fact that the amount of modes to be computed is limited to a maximum of 50 modes. Adding more SVD modes to the approximation increases the amount of separable terms of the stiffness matrix, which has an impact in the computation time of the solver. For the maximum amount of 50 modes it is seen that there is an improvement of the error up to seven modes, which is the optimal value for this setting. Increasing this to 15 modes gives a bigger error, which can be attributed to the fact that 50 modes are not enough to display the results with the same level of accuracy. Due to the scope of this work, the error of using seven modes is deemed as acceptable and is used as the optimal value to work in the homogenized case.

Appendix B

PGD algorithm implemented

Algorithms implemented in MATLAB (as appeared in [16]).

Data: $\boldsymbol{K}^k$, B_i^k, $\boldsymbol{f}^\ell$, S_i^ℓ and $\boldsymbol{U}_{\texttt{PGD}}^{n-1}(\boldsymbol{\mu})$: $\boldsymbol{u}^m$, G_i^m, for $m = 1, 2, \ldots, n-1$, $k = 1, 2, \ldots,$, $\ell = 1, 2, \ldots, n_{\texttt{f}}$
Result: $\boldsymbol{u}$ and G_i, for $i = 1, 2, \ldots, n_{\texttt{p}}$
Initialize: assign values to $\boldsymbol{u}$ and G_i; select a tolerance $\eta_{\texttt{tol}}$

while $\varepsilon > \zeta$ **do**
  Compute $\widetilde{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$, $\widetilde{G}_i = \frac{G_i}{\|G_i\|}$ and $\beta = \|\boldsymbol{u}\| \prod_{i=1}^{n_{\texttt{p}}} \|G_i\|$.
  for $k = 1 \ldots n_{\boldsymbol{k}}$ **do**
    Compute $c^k := \prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} B_i^k \, (G_i)^2 \, \mathrm{d}\mu_i$.
    for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**
      Compute $d_i^k(\cdot) := \left(\prod_{\substack{j=1; \\ j \neq i}}^{n_{\texttt{p}}} \int_{I_j} B_j^k \, (G_j)^2 \, \mathrm{d}\mu_j \right) B_i^k(\cdot)$.
    end
    for $m = 1 \ldots (n-1)$ **do**
      Compute $c^{k,m} := \prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} B_i^k \, G_i^m \, G_i \, \mathrm{d}\mu_i$.
      for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**
        Compute
        $d_i^{k,m}(\cdot) := \left(\prod_{\substack{j=1; \\ j \neq i}}^{n_{\texttt{p}}} \int_{I_j} B_j^k \, G_j^m \, G_j \, \mathrm{d}\mu_j \right) B_i^k(\cdot) \, G_i^m(\cdot)$.
      end
    end
  end
  for $\ell = 1 \ldots n_{\boldsymbol{f}}$ **do**
    Compute $\hat{c}^\ell := \prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} S_i^\ell \, G_i \, \mathrm{d}\mu_i$.
    for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**
      Compute $\hat{d}_i^\ell(\cdot) := \left(\prod_{\substack{j=1; \\ j \neq i}}^{n_{\texttt{p}}} \int_{I_j} S_j^\ell \, G_j \, \mathrm{d}\mu_j \right) S_i^\ell(\cdot)$.
    end
  end
  Solve $\boldsymbol{u}^{\texttt{new}}$;
  $\left[\sum_{k=1}^{n_{\texttt{k}}} \boldsymbol{K}^k \, c^k \right] \boldsymbol{u}^{\texttt{new}} = \sum_{\ell=1}^{n_{\texttt{f}}} \boldsymbol{f}^\ell \, \hat{c}^\ell - \sum_{m=1}^{n-1} \left[\sum_{k=1}^{n_{\texttt{k}}} \boldsymbol{K}^k \, c^{k,m} \right] \boldsymbol{u}^m$.
  for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**
    Update $G_i^{\texttt{new}}(\cdot) = \frac{\sum_{\ell=1}^{n_{\texttt{f}}} \left(\boldsymbol{u}^{\mathsf{T}} \boldsymbol{f}^\ell \right) \hat{d}_i^\ell(\cdot) - \sum_{m=1}^{n-1} \sum_{k=1}^{n_{\texttt{k}}} \left(\boldsymbol{u}^{\mathsf{T}} \boldsymbol{K}^k \boldsymbol{u}^m \right) d_i^{k,m}(\cdot)}{\sum_{k=1}^{n_{\texttt{k}}} \left(\boldsymbol{u}^{\mathsf{T}} \boldsymbol{K}^k \boldsymbol{u} \right) d_i^k(\cdot)}$.
  end
  Compute $\widetilde{\boldsymbol{u}}^{\texttt{new}} = \frac{\boldsymbol{u}^{\texttt{new}}}{\|\boldsymbol{u}^{\texttt{new}}\|}$, $\widetilde{G}_i^{\texttt{new}} = \frac{G_i^{\texttt{new}}}{\|G_i^{\texttt{new}}\|}$ and $\beta^{\texttt{new}} = \|\boldsymbol{u}^{\texttt{new}}\| \prod_{i=1}^{n_{\texttt{p}}} \|G_i^{\texttt{new}}\|$.
  Compute $\varepsilon = (\beta^{\texttt{new}})^2 + \beta^2 - 2\beta^{\texttt{new}} \beta \left(\widetilde{\boldsymbol{u}}^{\mathsf{T}} \boldsymbol{M}_u \widetilde{\boldsymbol{u}}^{\texttt{new}} \right) \prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} \widetilde{G}_i^{\texttt{new}} \widetilde{G}_i \, \mathrm{d}\mu_i$.
  Update values $\boldsymbol{u} \leftarrow \boldsymbol{u}^{\texttt{new}}$; $G_i \leftarrow G_i^{\texttt{new}}$; $\beta \leftarrow \beta^{\texttt{new}}$ and $\zeta = \eta_{\texttt{tol}} \beta^{\texttt{new}}$
end

Algorithm 1: PGD alternated directions nonlinear solver

Data: $\boldsymbol{U}^n_{\texttt{PGD}}(\boldsymbol{\mu})$: $\boldsymbol{u}^m$, G^m_i, for $m = 1, 2, \ldots, n$
$\boldsymbol{u}^{\widehat{n}-1}_{\texttt{com}}$: $\widehat{\boldsymbol{u}}^{\hat{m}}$, $\widehat{G}^{\hat{m}}_i$, for $\hat{m} = 1, 2, \ldots, (\widehat{n}-1)$
Result: $\widehat{\boldsymbol{u}}$ $\widehat{G}_i$, for $i = 1, 2, \ldots, n_{\texttt{p}}$
Initialize: assign values to $\widehat{\boldsymbol{u}}$ and $\widehat{G}_i$; select a tolerance $\eta_{\texttt{tol}}$

while $\varepsilon > \zeta$ **do**

Compute $\widetilde{\widehat{\boldsymbol{u}}} = \frac{\widehat{\boldsymbol{u}}}{\|\widehat{\boldsymbol{u}}\|}$, $\widetilde{\widehat{G}}_i = \frac{\widehat{G}_i}{\|\widehat{G}_i\|}$ and $\psi = \|\widehat{\boldsymbol{u}}\| \prod_{i=1}^{n_{\texttt{p}}} \left\|\widehat{G}_i\right\|$.

Compute $\lambda = \prod_{i=1}^{n_{\texttt{p}}} \left(\widehat{G}_i, \widehat{G}_i\right)$

for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**

Compute $\eta_i = \left(\widehat{\boldsymbol{u}}^{\mathsf{T}} \boldsymbol{M}_u \widehat{\boldsymbol{u}}\right) \prod_{j \neq i} \left(\widehat{G}_j, \widehat{G}_j\right)$

end

for $m = 1 \ldots (n-1)$ **do**

Compute $\gamma^m = \prod_{i=1}^{n_{\texttt{p}}} \left(\widehat{G}_i, G^m_i\right)$

for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**

Compute $\xi^m_i = \left(\widehat{\boldsymbol{u}}^{\mathsf{T}} \boldsymbol{M}_u \boldsymbol{u}^m\right) \prod_{j \neq i} \left(\widehat{G}_j, G^m_j\right)$

end

end

for $\hat{m} = 1 \ldots (\widehat{n}-1)$ **do**

Compute $\phi^{\hat{m}} = \prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} \widehat{G}_i \, \widehat{G}^{\hat{m}}_i \, \mathrm{d}\mu_j$.

for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**

Compute $\kappa^{\hat{m}}_i = \left(\widehat{\boldsymbol{u}}^{\mathsf{T}} \boldsymbol{M_u} \, \widehat{\boldsymbol{u}}^{\hat{m}}\right) \prod_{\substack{j=1; \\ \forall j \neq i}}^{n_{\texttt{p}}} \int_{I_j} \widehat{G}_j \, \widehat{G}^{\hat{m}}_j \, \mathrm{d}\mu_j$.

end

end

Update $\widehat{\boldsymbol{u}}^{\texttt{new}} = \sum_{m=1}^{n} \widehat{\boldsymbol{u}}^m \left(\frac{\gamma^m}{\lambda}\right) - \sum_{\hat{m}=1}^{\widehat{n}-1} \widehat{\boldsymbol{u}}^{\hat{m}} \left(\frac{\phi^{\hat{m}}}{\lambda}\right)$.

for $i = 1 \ldots n_{\boldsymbol{p}}$ **do**

Update $\widehat{G}^{\texttt{new}}_i(\cdot) = \sum_{m=1}^{n} \left(\frac{\xi^m_i}{\eta_i}\right) G^m_i(\cdot) - \sum_{\hat{m}=1}^{\widehat{n}-1} \left(\frac{\kappa^{\hat{m}}_i}{\eta_i}\right) \widehat{G}^{\hat{m}}_i(\cdot)$.

end

Compute $\widehat{\boldsymbol{u}}^{\texttt{new}} = \frac{\widehat{\boldsymbol{u}}^{\texttt{new}}}{\|\widehat{\boldsymbol{u}}^{\texttt{new}}\|}$, $\widetilde{\widehat{G}}^{\texttt{new}}_i = \frac{\widehat{G}^{\texttt{new}}_i}{\|\widehat{G}^{\texttt{new}}_i\|}$ and $\psi^{\texttt{new}} = \|\widehat{\boldsymbol{u}}^{\texttt{new}}\| \prod_{i=1}^{n_{\texttt{p}}} \left\|\widehat{G}^{\texttt{new}}_i\right\|$.

Compute

$\varepsilon = (\psi^{\texttt{new}})^2 + \psi^2 - 2\,\psi^{\texttt{new}} \psi \left((\widetilde{\widehat{\boldsymbol{u}}}^{\texttt{new}})^{\mathsf{T}} \boldsymbol{M_u} \, \widetilde{\widehat{\boldsymbol{u}}}\right) \left(\prod_{i=1}^{n_{\texttt{p}}} \int_{I_i} \widetilde{\widehat{G}}^{\texttt{new}}_i \, \widetilde{\widehat{G}}_i \, \mathrm{d}\mu_j\right)$.

Update values $\widehat{\boldsymbol{u}} \leftarrow \widehat{\boldsymbol{u}}^{\texttt{new}}$; $\widehat{G}_i \leftarrow \widehat{G}^{\texttt{new}}_i$; $\psi \leftarrow \psi^{\texttt{new}}$ and $\zeta = \eta_{\texttt{tol}} \psi^{\texttt{new}}$

end

Algorithm 2: Least-Squares PGD alternated directions nonlinear solver