



## Project report

# Numerical homogenisation of mechanical properties

*Study programme:*

*Study branch:*

*Author:*

*Supervisor:*

B0588A110003 – Applied Sciences in Engineering

B0588A110003AVI – Applied Sciences in Engineering

**Ronald Christopher Siddall**

doc. Mgr. Jan Stebel, Ph.D.

Liberec 2024

## ZADÁNÍ SEMESTRÁLNÍHO PROJEKTU

Jméno a příjmení: **Ronald Christopher Siddall**  
Název práce: **Numerická homogenizace mechanických vlastností**  
Zadávající katedra: **Ústav nových technologií a aplikované informatiky**  
Vedoucí práce: **doc. Mgr. Jan Stebel Ph.D.**  
Rozsah práce: **15–20 stran**

### Zásady pro vypracování:

1. Rešerše:
  - Nastudování základů mechaniky pružných těles a jejich modelování metodou konečných prvků.
  - Nastudování základů výpočetní mikromechaniky a její využití pro modelování heterogenních materiálů.
2. Numerická homogenizace
  - Seznámení se s výpočetním SW Flow123d.
  - Návrh výpočetního postupu pro určení efektivního elastického tenzoru ve 2D.
  - Implementace zvoleného postupu do programu, který provede výpočetní analýzu detailního (mikrostrukturálního) modelu a určí efektivní elastický tenzor.
3. Aplikace
  - Demonstrace výpočetního postupu na modelových úlohách s heterogenními materiály.

### Seznam odborné literatury:

- [1] JOHNSON, C. Numerical solution of partial differential equations by the finite element method. Courier Corporation, 2012.
- [2] SCHÄFER, M. Computational engineering: introduction to numerical methods. Berlin: Springer, 2006.
- [3] ZOHDİ, T. I.; WRIGGERS, P. An introduction to computational micromechanics. Springer Science & Business Media, 2004.

V Liberci dne .....

.....  
doc. Mgr. Jan Stebel Ph.D.

## Declaration

I hereby certify, I, myself, have written my project report as an original and primary work using the literature listed below and consulting it with my thesis supervisor and my thesis counsellor.

I acknowledge that my project report is fully governed by Act No. 121/2000 Coll., the Copyright Act, in particular Article 60 – School Work.

I acknowledge that the Technical University of Liberec does not infringe my copyrights by using my project report for internal purposes of the Technical University of Liberec.

I am aware of my obligation to inform the Technical University of Liberec on having used or granted license to use the results of my project report; in such a case the Technical University of Liberec may require reimbursement of the costs incurred for creating the result up to their actual amount.

I acknowledge that the Technical University of Liberec will make my project report public in accordance with paragraph 47b of Act No. 111/1998 Coll., on Higher Education Institutions and on Amendment to Other Acts (the Higher Education Act), as amended.

I am aware of the consequences which may under the Higher Education Act result from a breach of this declaration.

24. 5. 2024

Ronald Christopher Siddall

## Numerical homogenisation of mechanical properties

### Abstrakt

Tato práce se zaměřuje na numerickou homogenizaci mechanických vlastností dvoudimenzionálních fyzikálních problémů. Numerická homogenizace umožňuje modelovat mechanické vlastnosti materiálů s komplikovanou mikroheterogenní strukturou. K řešení byl implementován program, který automaticky prováděl homogenizaci s využitím simulátoru Flow123d.

**Klíčová slova:** efektivní elastický tenzor, homogenizace, reprezentativní objemový vzorek, mikromechanika, mechanika pružných těles, modelování heterogenních materiálů

## Numerical homogenisation of mechanical properties

### Abstract

This work focuses on the numerical homogenization of mechanical properties of two-dimensional physics problems. Numerical homogenization allows to model the mechanical properties of materials with complicated micro heterogeneous structure. A program was developed to automatically perform the homogenization using the Flow123d simulator.

**Keywords:** effective elastic tensor, homogenization, representative elementary volume, micromechanics, mechanics of elastic bodies, modelling of heterogeneous materials

## Acknowledgements

I would like to especially thank my thesis advisor, Assoc. Mgr. Jan Stebel, Ph.D.

His expertise, constructive criticism, patience, fair approach and constant willingness to provide valuable advice and comments during the development of this thesis were indispensable for me.

In addition, I would like to thank my parents and family who also provided me with the necessary support, energy and space to work.

And of course to my colleagues at AVI.

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
<b>2</b>	<b>Theory</b>	<b>9</b>
2.1	Mathematical description of mechanics of elastic bodies . . . . .	9
2.1.1	Deformation and displacement . . . . .	9
2.1.2	Elasticity, strength, stiffness and Young's modulus . . . . .	10
2.1.3	Poisson's constant . . . . .	11
2.1.4	Generalized Hooke's Law and balance of forces . . . . .	11
2.1.5	Voigtova notace . . . . .	12
2.2	Numerical homogenization . . . . .	13
2.2.1	Definition of homogenization and effective elastic tensor . . . . .	13
2.2.2	Problem boundary conditions and load matrices . . . . .	13
2.2.3	Effective elastic tensor for general load matrices . . . . .	14
<b>3</b>	<b>Description of microstructures and calculation of the effective tensor</b>	<b>16</b>
3.1	Chessboard and sandwich . . . . .	16
3.2	Description of the numerical calculation of the results . . . . .	18
3.2.1	Effective elastic tensor for specific load matrices . . . . .	18
3.2.2	Representative volume element and tolerances . . . . .	19
3.2.3	Calculation of the relative residue . . . . .	20
<b>4</b>	<b>Program description, functions and documentation</b>	<b>22</b>
4.1	Prerequisites before using the program . . . . .	22
4.1.1	Booting in Windows OS . . . . .	22
4.2	General program structure . . . . .	23
4.3	Description of individual classes and files . . . . .	25
4.3.1	config_file.yaml . . . . .	25
4.3.2	GenerateMesh.py . . . . .	26
4.3.3	ConfigManager.py . . . . .	27
4.3.4	GenerateVtuFiles.py . . . . .	27
4.3.5	EffectiveElasticTensor.py . . . . .	27
4.3.6	main.py . . . . .	29
4.3.7	automatic_simulation.py . . . . .	30

<b>5</b>	<b>Calculations of sample problems</b>	<b>31</b>
5.1	Example of a one-off simulation . . . . .	31
5.2	Example of automatic simulation . . . . .	36
<b>6</b>	<b>Results of the calculations performed</b>	<b>39</b>
6.1	Summary of all calculations performed . . . . .	39
6.2	Final results of calculations performed . . . . .	40
6.2.1	Chessboard, $Y1 = 100$ , $Y2 = 50$ . . . . .	40
6.2.2	Chessboard, $Y1 = 200$ , $Y2 = 50$ . . . . .	41
6.2.3	Chessboard, $Y1 = 500$ , $Y2 = 50$ . . . . .	41
6.2.4	Chessboard, $Y1 = 2500$ , $Y2 = 50$ . . . . .	41
6.2.5	Sandwich, $Y1 = 100$ , $Y2 = 50$ . . . . .	42
6.2.6	Sandwich, $Y1 = 200$ , $Y2 = 50$ . . . . .	42
6.2.7	Sandwich, $Y1 = 500$ , $Y2 = 50$ . . . . .	42
6.2.8	Sandwich, $Y1 = 2500$ , $Y2 = 50$ . . . . .	43
6.2.9	Tables with overview of REV dimension values . . . . .	43
6.3	Graphs of different dependencies . . . . .	43
6.4	Validation of results . . . . .	46
6.4.1	Results for even and odd multiples of $n$ . . . . .	46
6.4.2	Analysis of computed results using the mixture rule . . . . .	46
6.5	Summary of results and observations . . . . .	47
<b>7</b>	<b>Conclusion</b>	<b>48</b>

# 1 Introduction

We have all encountered a material that has a heterogeneous, i.e. spatially variable, microstructure. There are many materials with this microstructure, but some of the most important ones are rocks, biomaterials, wood and composite materials.

Granite is an example of a rock composed of various minerals, mainly quartz, feldspar and mica. Similarly, biomaterials such as bone and muscle exhibit microheterogeneity. Muscles contain different types of muscle fibres, while bones are made up of many minerals and organic substances. Last but not least, composites, such as Kevlar, are a good example of a heterogeneous material, as composites are usually composed of several distinct components with different properties.

What do all these materials have in common? Due to their very complex microstructure, it is extremely difficult to predict or model their mechanical properties in general. Mechanical properties such as stiffness, strength or elasticity are characteristics of a material that describe its behaviour and response to different mechanical loads (e.g. tensile, compressive or shear).

These properties are central to a deeper understanding of the behaviour of a given material. If we do not know the mechanical properties of a material, it is difficult to predict its behaviour in different situations. The moment we are unable to predict its behaviour, it makes it a huge complication in researching, developing or improving these materials in industry, technology, science, medicine, etc.

In other words, the general modelling of microheterogeneous materials with complex microstructure is a very challenging to almost impossible task, but one that must be addressed.

Numerical homogenization of mechanical properties offers one way to address this problem, and this is what this semester project addresses.

Thus, in this project I show a way to obtain effective mechanical properties of a material with a microheterogeneous structure using numerical homogenization.

First, I analyze the homogenization problem mathematically. Then I describe how I implement a program that automatically calculates the effective mechanical properties using numerical homogenization for two different types of microstructures. I then discuss how to optimally obtain mechanical properties for different types of materials. Finally, I give a comprehensive summary of the results and dependencies I observed.



## 2 Theory

The description of general numerical homogenization for three-dimensional space and arbitrarily large deformations is very complicated. Therefore, this project will deal with a simplified physical model that considers only **two-dimensional** (planar) problems and only small, i.e. **linear elastic deformations**. The theoretical part is divided into two subsections. First, the chapter 2.1 summarizes the basic findings of the mathematical theory of elasticity. Then, in chapter 2.2, the basic idea of numerical homogenization and the general procedure for obtaining the effective elastic tensor are described.

### 2.1 Mathematical description of mechanics of elastic bodies

We start by defining the scalar products of vectors and tensors. We define the scalar product of the vectors  $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$  as follows:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i. \quad (2.1)$$

Similarly, the scalar product of the second order tensors  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$  is defined as follows:

$$\mathbf{A} : \mathbf{B} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} B_{ij}. \quad (2.2)$$

Next, we need to define the so-called **Frobenius norm** of the tensor. If we have a general tensor  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with elements  $a_{ij}$ , then the Frobenius norm is:

$$\|\mathbf{A}\| = \sqrt{\sum_{i=1}^n \sum_{j=1}^m a_{ij}^2} \quad (2.3)$$

#### 2.1.1 Deformation and displacement

Our aim is to describe the response of a microheterogeneous body or material under load or **during deformation**.

The term deformation of a solid is understood as a change in the shape or volume of the solid caused by the application of some force. These forces, also known as

deformation forces, induce stresses in the body and there are several types of them. The most common deformation forces are tensile, compressive, shear, bending or torsion. As such, deformations are divided into two basic types, namely **elastic** (elastic) and **plastic** (inelastic). Elastic deformation is a reversible (reversible) process, i.e. if a body is elastically deformed, then after the action is over the body returns to its original state. If it is plastic deformation, we speak of irreversible (irreversible) agencies. In other words, in plastic deformations, the shape of the material will change permanently (ch. 5.2 (Brdička et al., 2005)).

It is quite natural to conclude that elastic deformations will only occur for relatively small forces (depending on the type of material, of course), while plastic deformations will occur for larger forces.

For the mathematical description of elastic deformation, we first introduce the so-called **displacement vector**.

Consider an arbitrary point whose Cartesian coordinates before deformation are denoted by  $x_i$ . The external forces will cause the point to shift by the vector  $\mathbf{u}$  to another point with coordinates  $y_i$ . The displacement vector (or displacement for short)  $\mathbf{u}$  is then defined for this point as follows (Chap. 4.1. (Brdička et al., 2005)):

$$\mathbf{u} = \mathbf{y} - \mathbf{x}, \quad (2.4)$$

where vector  $\mathbf{x}$  with components  $x_i, i = 1, 2$  is the position of the point before deformation and vector  $\mathbf{y}$  with components  $y_i, i = 1, 2$  is the position of the point after deformation.

It should be emphasized that the displacement vector is a **generic** description of the change in the position of the point. In other words, while the displacement vector is related to the deformation, they are not equivalent concepts. For example, translation (physical displacement) or rotation (rotation) will lead to a non-trivial (non-zero) displacement, i.e.  $\mathbf{u} \neq 0$ . However, there is no deformation during these motions because there is no change in the shape and volume of the body.

The change of shape is described by the so-called **tensor of small deformations** (Schafer, 2006)  $\boldsymbol{\varepsilon}$ , whose elements are given by the relation:

$$\varepsilon_{kl} = \left( \frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} \right), \quad k, l = 1, 2. \quad (2.5)$$

### 2.1.2 Elasticity, strength, stiffness and Young's modulus

Strength, elasticity, stiffness and Young's modulus are concepts that are related to the relationship between deformation and applied forces.

Elasticity is the ability of a material to deform under stress and then return to its original state.

Strength is defined by the so-called ultimate strength. The ultimate strength is the maximum stress that a material can withstand before permanent deformation or failure.

Stiffness expresses how quickly a material deforms in response to an external force or resistance to deformation under load. Young's modulus  $Y$ , in other words,

the tensile modulus, can be interpreted as a measure of the tensile stiffness of a material, but is more accurately defined as the ratio of the tensile stress  $\sigma$  and the relative elongation<sup>1</sup>  $\varepsilon$ . Relative elongation is defined as the ratio of the elongation of the material  $\Delta l$  to its original length  $l_0$  (Ch. 5.2. (Brdička et al., 2005)):

$$Y = \frac{\sigma}{\varepsilon}, \quad (2.6)$$

$$\varepsilon = \frac{\Delta l}{l_0} = \frac{l - l_0}{l_0}. \quad (2.7)$$

The equation (2.6) is sometimes referred to as Hooke's law of elasticity, often written in the form  $\sigma = Y\varepsilon$  (equation 5.2.1 ch. 5.2 (Brdička et al., 2005)). This linear equation is only valid for small values of stress (which is consistent with the assumptions of this project, since we are only concerned with linear elastic deformations). Due to the linear nature of this equation,  $Y$  can be viewed as a directive, i.e., a constant of proportionality in the linear dependence of  $\sigma$  on  $\varepsilon$ .

### 2.1.3 Poisson's constant

The Poisson constant  $\nu$  can be defined as follows:

$$\nu = -\frac{\varepsilon_x}{\varepsilon_y}, \quad (2.8)$$

where  $\varepsilon_x$  is the longitudinal stretch and  $\varepsilon_y$  is the transverse stretch.

For most materials,  $\varepsilon_x \geq 0, \varepsilon_y < 0$ , which means that the material narrows transversely when stretched. For these most common materials, the value of  $\nu$  lies in the range of 0 to 0.5. However, there are also modern so-called auxetic materials which also expand in the transverse direction when stretched ( $\varepsilon_y > 0$ ) and for which  $\nu < 0$  applies. (Carneiro et al., 2013).

### 2.1.4 Generalized Hooke's Law and balance of forces

Young's modulus and Poisson's constant characterize the elastic response of so-called *isotropic materials*, whose mechanical properties are **same** for all directions. The behaviour of general anisotropic materials is described by the so-called **elastic tensor** via **generalized Hooke's law**.

Consider a solid microheterogeneous body which is bounded by a region  $\Omega \subset \mathbb{R}^2$ . Then the linear elastic deformation of the body on the  $\Omega$  region can be described by the generalized Hooke's law:

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon}, \quad \text{tj.} \quad \sigma_{ij} = \sum_{k,l=1}^2 C_{ijkl} \varepsilon_{kl}, \quad i, j = 1, 2, \quad (2.9)$$

where  $\boldsymbol{\sigma}, \boldsymbol{\varepsilon} \in \mathbb{R}^{2 \times 2}$ ,  $\mathbf{C} \in \mathbb{R}^{2 \times 2 \times 2 \times 2}$ ,  $\boldsymbol{\sigma}$  is the **stress tensor** and  $\mathbf{C}$  is the elastic tensor<sup>2</sup>.

<sup>1</sup>There are several names for this, e.g., relative strain, longitudinal strain, etc.

<sup>2</sup>This is also called the Hooke tensor

The generalized Hooke's law alone is not sufficient to fully describe the vector fields of displacement  $\mathbf{u}$  and stress  $\boldsymbol{\sigma}$  in the whole  $\Omega$  region. For a complete description it is necessary to consider the so-called force balance and boundary conditions:

$$\begin{aligned} \text{v } \Omega : \quad \nabla \cdot \boldsymbol{\sigma} &= \mathbf{f} \quad (\text{balance of forces}), \\ \text{na } \Gamma_1 : \quad \mathbf{u} &= \mathbf{u}_D \quad (\text{Dirichlet condition}), \\ \text{na } \Gamma_2 : \quad \boldsymbol{\sigma} \cdot \mathbf{n} &= \mathbf{t}_N \quad (\text{Neumann Condition}), \end{aligned} \quad (2.10)$$

where  $\nabla$  is the nabla operator,  $\mathbf{f}$  are the external applied forces,  $\Gamma_1$  and  $\Gamma_2$  are the two disjoint parts of the boundary  $\partial\Omega$ ,  $\mathbf{n}$  is the unit normal vector,  $\mathbf{u}_D$  is the specified displacement vector and  $\mathbf{t}_N$  is the specified surface force.

### 2.1.5 Voigtova notace

In mathematics, Voigt notation is a way to represent a **symmetric** tensor by a lower order tensor (Povey, 2023).

Consider the second order two-dimensional symmetric tensor  $\mathbf{A} \in \mathbb{R}^{2 \times 2}$  and the fourth order two-dimensional symmetric tensor  $\mathbf{B} \in \mathbb{R}^{2 \times 2 \times 2 \times 2}$ .

In Voigt notation, the tensor  $\mathbf{A}$  reduces to a first order tensor (vector) and the tensor  $\mathbf{B}$  reduces to a second order tensor (matrix):

$$\begin{aligned} \mathbf{A} \in \mathbb{R}^{2 \times 2} &\longrightarrow \tilde{\mathbf{A}} \in \mathbb{R}^3, \\ \mathbf{B} \in \mathbb{R}^{2 \times 2 \times 2 \times 2} &\longrightarrow \tilde{\mathbf{B}} \in \mathbb{R}^{3 \times 3}, \end{aligned} \quad (2.11)$$

where  $\tilde{\mathbf{A}}$  and  $\tilde{\mathbf{B}}$  denote the reduced tensors. These reduced tensors contain only *various* elements of the original tensors:

$$\tilde{\mathbf{A}} = \begin{bmatrix} A_{11} \\ A_{22} \\ A_{12} \end{bmatrix}, \quad \tilde{\mathbf{B}} = \begin{bmatrix} B_{1111} & B_{1122} & B_{1112} \\ B_{2211} & B_{2222} & B_{2212} \\ B_{1211} & B_{1222} & B_{1212} \end{bmatrix}.$$

Since  $\boldsymbol{\sigma}$ ,  $\boldsymbol{\varepsilon}$  and  $\mathbf{C}$  are symmetric tensors, it is possible to apply (2.11) to them. However, a small correction is introduced for the small deformation tensor  $\boldsymbol{\varepsilon}$ :

$$\tilde{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{bmatrix} \quad (2.12)$$

kde  $\gamma_{12} = 2\varepsilon_{12}$ . This preserves the scalar product  $\tilde{\boldsymbol{\sigma}} \cdot \tilde{\boldsymbol{\varepsilon}} = \boldsymbol{\sigma} : \boldsymbol{\varepsilon}$ , which is a quantity important for expressing mechanical energy. In Voigt notation, the equation (2.9) takes the form:

$$\tilde{\boldsymbol{\sigma}} = \tilde{\mathbf{C}} \tilde{\boldsymbol{\varepsilon}}, \quad (2.13)$$

where

$$\tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix}, \quad \tilde{\mathbf{C}} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1112} \\ C_{2211} & C_{2222} & C_{2212} \\ C_{1211} & C_{1222} & C_{1212} \end{bmatrix}, \quad \tilde{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{bmatrix}, \quad (2.14)$$

which is **generalized Hooke's law in Voigt notation**.

## 2.2 Numerical homogenization

Microheterogeneous materials have different properties at different locations, which are generally described by the elastic tensor. In other words, these materials are spatially variable, which can be seen mathematically as the dependence of all the partial tensors appearing in the equation (2.13) on the position of space. Thus, a dependence on  $\mathbf{x}$  and  $\mathbf{u}(\mathbf{x})$  can be added to the equation (2.13):

$$\tilde{\boldsymbol{\sigma}}(\mathbf{x}) = \tilde{\mathbf{C}}(\mathbf{x})\tilde{\boldsymbol{\varepsilon}}(\mathbf{u}(\mathbf{x})). \quad (2.15)$$

The main goal of this work is to replace the space-dependent tensor  $\tilde{\mathbf{C}}(\mathbf{x})$  with a constant effective elastic tensor  $\tilde{\mathbf{C}}^*$  with the most similar properties.

### 2.2.1 Definition of homogenization and effective elastic tensor

Homogenization is performed using the averaging operation  $\langle \cdot \rangle$ , defined as:

$$\langle \cdot \rangle = \frac{1}{|\Omega|} \int_{\Omega} \cdot d\Omega. \quad (2.16)$$

If we apply this operator to the equation (2.15), we get:

$$\frac{1}{|\Omega|} \int_{\Omega} \tilde{\boldsymbol{\sigma}}(\mathbf{x}) d\Omega = \frac{1}{|\Omega|} \int_{\Omega} \tilde{\mathbf{C}}(\mathbf{x})\tilde{\boldsymbol{\varepsilon}}(\mathbf{u}(\mathbf{x})) d\Omega. \quad (2.17)$$

Our goal is to replace the tensor  $\tilde{\mathbf{C}}(\mathbf{x})$  by a tensor  $\tilde{\mathbf{C}}^*$  such that it is independent of  $\mathbf{x}$  and satisfies the following requirement:

$$\frac{1}{|\Omega|} \int_{\Omega} \tilde{\boldsymbol{\sigma}}(\mathbf{x}) d\Omega = \tilde{\mathbf{C}}^* \cdot \frac{1}{|\Omega|} \int_{\Omega} \tilde{\boldsymbol{\varepsilon}}(\mathbf{u}(\mathbf{x})) d\Omega$$

respectively

$$\langle \tilde{\boldsymbol{\sigma}} \rangle = \langle \tilde{\mathbf{C}}\tilde{\boldsymbol{\varepsilon}} \rangle = \tilde{\mathbf{C}}^* \langle \tilde{\boldsymbol{\varepsilon}} \rangle. \quad (2.18)$$

The homogenization of the equation (2.13) can be written symbolically as follows:

$$\tilde{\boldsymbol{\sigma}} = \tilde{\mathbf{C}}\tilde{\boldsymbol{\varepsilon}} \xrightarrow{\text{homogenizace}} \langle \tilde{\boldsymbol{\sigma}} \rangle = \tilde{\mathbf{C}}^* \langle \tilde{\boldsymbol{\varepsilon}} \rangle \quad (2.19)$$

### 2.2.2 Problem boundary conditions and load matrices

Given that  $\tilde{\mathbf{C}}^* \in \mathbb{R}^{3 \times 3}$  and  $\langle \tilde{\boldsymbol{\sigma}} \rangle, \langle \tilde{\boldsymbol{\varepsilon}} \rangle \in \mathbb{R}^3$ , the relation (2.19) is a system of 3 equations with 9 unknowns, which is not enough equations to find  $\tilde{\mathbf{C}}^*$ .

To completely determine all elements of the tensor  $\tilde{\mathbf{C}}^*$ , it is necessary to perform a number of **linearly independent** loads via **boundary conditions** defined as follows:

$$\mathbf{u}_D = \mathbf{E}\mathbf{x}, \quad (2.20)$$

where  $\mathbf{u}_D$  is the displacement,  $\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} \\ E_{12} & E_{22} \end{bmatrix}$  denotes **symmetric load matrix** and  $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ .

Substituting these boundary conditions into the problem (2.10) we obtain the displacement field  $\mathbf{u}$  and the stress field  $\boldsymbol{\sigma}$  in the whole  $\Omega$  region.

The result of averaging is then 3 *different averaged stress components*. Thus we get 3 equations for the elements of  $\tilde{\mathbf{C}}^*$ . In order to determine the entire tensor  $\tilde{\mathbf{C}}^*$ , we need 3 different cases to have 9 equations for 9 unknowns. In order to apply the general load matrix to the equation (2.18), it is first necessary to define the relationship between  $\tilde{\mathbf{E}}$  (the general load matrix in Voigt notation) and  $\langle \tilde{\boldsymbol{\varepsilon}} \rangle$ . This equation is a result of "The Average Strain Theorem", which is described in detail in chapter 4.1.1 (Zohdi and Wriggers, 2005):

$$\tilde{\mathbf{E}} = \langle \tilde{\boldsymbol{\varepsilon}} \rangle \iff \begin{bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{bmatrix} = \begin{bmatrix} \langle \varepsilon_{11} \rangle \\ \langle \varepsilon_{22} \rangle \\ \langle \gamma_{12} \rangle \end{bmatrix}. \quad (2.21)$$

We now show the procedure for determining the effective elastic tensor.  $\tilde{\mathbf{C}}^*$ .

### 2.2.3 Effective elastic tensor for general load matrices

By applying three different load matrices, we obtain from the equation (2.18) these conditions:

$$\begin{aligned} \langle \tilde{\boldsymbol{\sigma}} \rangle^1 &= \tilde{\mathbf{C}}^* \langle \tilde{\boldsymbol{\varepsilon}} \rangle^1, \\ \langle \tilde{\boldsymbol{\sigma}} \rangle^2 &= \tilde{\mathbf{C}}^* \langle \tilde{\boldsymbol{\varepsilon}} \rangle^2, \\ \langle \tilde{\boldsymbol{\sigma}} \rangle^3 &= \tilde{\mathbf{C}}^* \langle \tilde{\boldsymbol{\varepsilon}} \rangle^3, \end{aligned} \quad (2.22)$$

where  $\langle \tilde{\boldsymbol{\sigma}} \rangle^i$  and  $\langle \tilde{\boldsymbol{\varepsilon}} \rangle^i, i = 1, 2, 3$ , represent the obtained averaged tensors for each case. These systems of equations (2.22) can be rewritten to give the following more compact notation:

$$\begin{aligned} \mathbf{K}\mathbf{c}_1 &= \mathbf{s}_1, \\ \mathbf{K}\mathbf{c}_2 &= \mathbf{s}_2, \\ \mathbf{K}\mathbf{c}_3 &= \mathbf{s}_3, \end{aligned} \quad (2.23)$$

or equivalently:

$$\begin{bmatrix} \mathbf{K} & & \\ & \mathbf{K} & \\ & & \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_3 \end{bmatrix}, \quad (2.24)$$

where  $\mathbf{K}$ ,  $\mathbf{c}_1$ ,  $\mathbf{c}_2$ ,  $\mathbf{c}_3$ ,  $\mathbf{s}_1$ ,  $\mathbf{s}_2$  and  $\mathbf{s}_3$  are defined as follows:

$$\begin{aligned}
\mathbf{K} &= \begin{bmatrix} E_{11}^1 & E_{22}^1 & 2E_{12}^1 \\ E_{11}^2 & E_{22}^2 & 2E_{12}^2 \\ E_{11}^3 & E_{22}^3 & 2E_{12}^3 \end{bmatrix}, \\
\mathbf{c}_1 &= \begin{bmatrix} C_{1111}^* \\ C_{1122}^* \\ C_{1112}^* \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} C_{2211}^* \\ C_{2222}^* \\ C_{2212}^* \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} C_{1211}^* \\ C_{1222}^* \\ C_{1212}^* \end{bmatrix}, \\
\mathbf{s}_1 &= \begin{bmatrix} \langle \sigma_{11} \rangle^1 \\ \langle \sigma_{11} \rangle^2 \\ \langle \sigma_{11} \rangle^3 \end{bmatrix}, \quad \mathbf{s}_2 = \begin{bmatrix} \langle \sigma_{22} \rangle^1 \\ \langle \sigma_{22} \rangle^2 \\ \langle \sigma_{22} \rangle^3 \end{bmatrix}, \quad \mathbf{s}_3 = \begin{bmatrix} \langle \sigma_{12} \rangle^1 \\ \langle \sigma_{12} \rangle^2 \\ \langle \sigma_{12} \rangle^3 \end{bmatrix}.
\end{aligned} \tag{2.25}$$

Our goal is to express  $\mathbf{c}_1$ ,  $\mathbf{c}_2$  and  $\mathbf{c}_3$  from each of the respective equations, because this gives us the solution for the elements of the effective elastic tensor  $\tilde{\mathbf{C}}^*$  that are contained within  $\mathbf{c}_1$ ,  $\mathbf{c}_2$  and  $\mathbf{c}_3$ . This system of equations can be solved analytically or numerically.

## 3 Description of microstructures and calculation of the effective tensor

### 3.1 Chessboard and sandwich

In general, microheterogeneous materials can have very diverse microstructures, which usually depend on the material. In this project, two types of microstructures were addressed, they were named:

- 1) **chessboard** - a prototype microstructure representing e.g. a cross section through a fibre composite material;
- 2) **sandwich** - prototype of a longitudinal cut with laminated or fibre composite material.

The whole area of the considered sample of material  $\Omega$  can be divided into a clearly defined periodically repeating number of sub-areas or subsets according to the given type of microstructure. We denote these subsets by A and B (see figures 3.1, 3.2 and 3.3). Both of these microstructures are defined by three optional parameters:

- $n$
- $Y_1$
- $Y_2$

The number  $n \in \mathbb{N}$ ,  $n > 1$  defines the dimension of the given structure, i.e. the number of subareas in one direction.

The numbers  $Y_1, Y_2 \in \mathbb{R}$ ,  $Y_1, Y_2 > 0$  are Young's moduli of elasticity. The subset A corresponds to  $Y_1$  and the subset B corresponds to  $Y_2$ . For the purposes of this project, we will assume that  $Y_1 > Y_2$ .

Due to the nature of the problems this project solves, we will assume  $\nu = 0$  for all simulations because otherwise we would get non-zero elements in the effective elastic tensor in the  $z$ -axis, which can be neglected for two-dimensional problems.



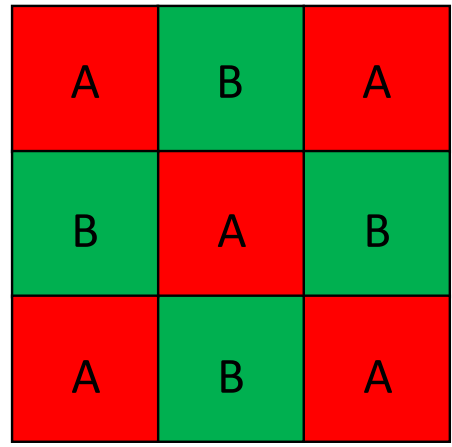
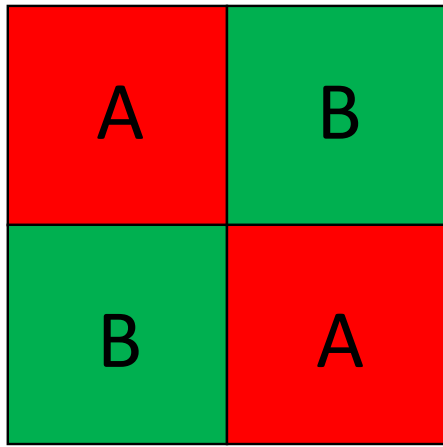


Figure 3.1: Example of chessboard structure for  $n = 2$  (left) and  $n = 3$  (right)

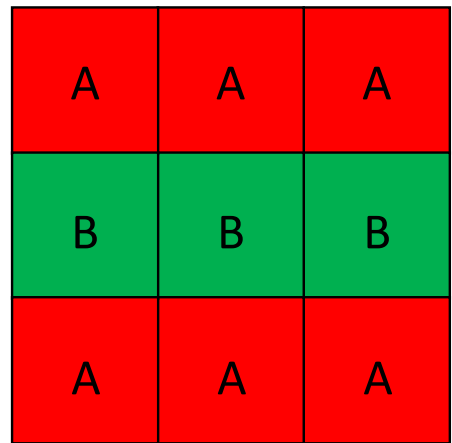
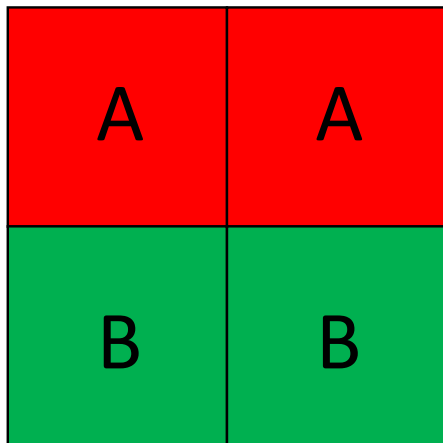


Figure 3.2: Sample sandwich structure for  $n = 2$  (left) and  $n = 3$  (right)

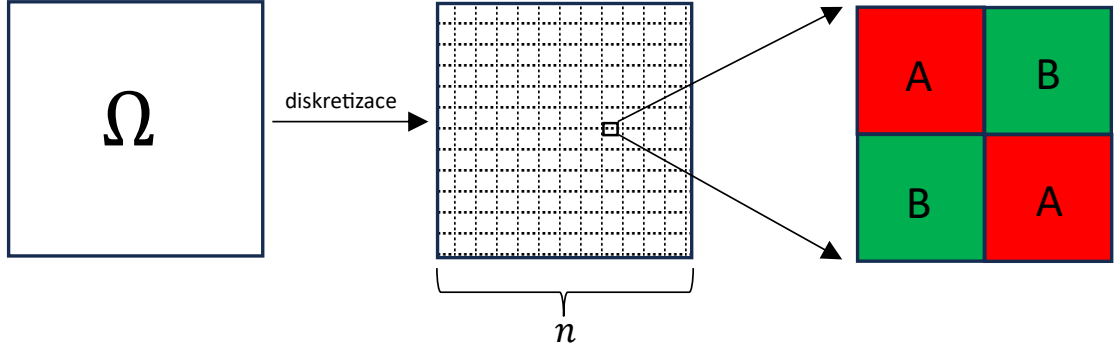


Figure 3.3: Visualization of the discretization of the  $\Omega$  region into  $n^2$  subregions

## 3.2 Description of the numerical calculation of the results

In this section, we first present the calculation of the effective tensor  $\tilde{\mathbf{C}}^*$  for the particular load matrices chosen. Next, we introduce the concept of „representative volume sample” and describe the procedure to find its size at the chosen tolerance.

### 3.2.1 Effective elastic tensor for specific load matrices

In the chapter 2.2.3, the procedure for calculating  $\tilde{\mathbf{C}}^*$  for three general load matrices was derived. In the following, we consider only the following three linearly independent load matrices:

$$\mathbf{E}_1 = \begin{bmatrix} \beta & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{E}_2 = \begin{bmatrix} 0 & 0 \\ 0 & \beta \end{bmatrix}, \quad \mathbf{E}_3 = \begin{bmatrix} 0 & \beta \\ \beta & 0 \end{bmatrix}, \quad (3.1)$$

where  $\beta > 0$  is the stress parameter<sup>1</sup>.

These load matrices can be equivalently expressed using Voigt notation (2.11):

$$\tilde{\mathbf{E}}_1 = \begin{bmatrix} \beta \\ 0 \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{E}}_2 = \begin{bmatrix} 0 \\ \beta \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{E}}_3 = \begin{bmatrix} 0 \\ 0 \\ 2\beta \end{bmatrix}. \quad (3.2)$$

<sup>1</sup>For our case of linear elasticity, it is sufficient to consider e.g. the value  $\beta = 1$ .

To find the solution  $\tilde{\mathbf{C}}^*$  for a particular matrix defined in (3.2), využijeme rovnosti (2.21), kterou dosadíme do (2.22). As a result, we get the following:

$$\begin{aligned} C_{1111}^* &= \frac{\langle \sigma_{11} \rangle^1}{\beta}, & C_{2211}^* &= \frac{\langle \sigma_{22} \rangle^1}{\beta}, & C_{1211}^* &= \frac{\langle \sigma_{12} \rangle^1}{\beta}, \\ C_{1122}^* &= \frac{\langle \sigma_{11} \rangle^2}{\beta}, & C_{2222}^* &= \frac{\langle \sigma_{22} \rangle^2}{\beta}, & C_{1222}^* &= \frac{\langle \sigma_{12} \rangle^2}{\beta}, \\ C_{1112}^* &= \frac{\langle \sigma_{11} \rangle^3}{2\beta}, & C_{2212}^* &= \frac{\langle \sigma_{22} \rangle^3}{2\beta}, & C_{1212}^* &= \frac{\langle \sigma_{12} \rangle^3}{2\beta}. \end{aligned} \quad (3.3)$$

If we fix the stress parameter  $\beta = 1$ , each partial homogenized element of the stress tensor is simply equal to the corresponding element of the effective elastic tensor (or its half).

### 3.2.2 Representative volume element and tolerances

The values in the effective tensor  $\tilde{\mathbf{C}}^*$  are influenced by the size of the microstructure considered, i.e. the parameter  $n$ . In order to achieve values close to the infinite periodic structure,  $n$  must be chosen large enough so that at the same time the problem remains computationally tractable. For this purpose, the concept of **representative volume element** (REV for short) is used. REV is defined as the **minimum volume**<sup>2</sup> (or range) beyond which the material **characteristics remain** essentially **constant** (Min and Jing, 2003). In other words, it is the smallest volume over a given area that can, on average, describe the mechanical properties under investigation over the entire area.

---

<sup>2</sup>In the context of the project, volume is seen as an area.

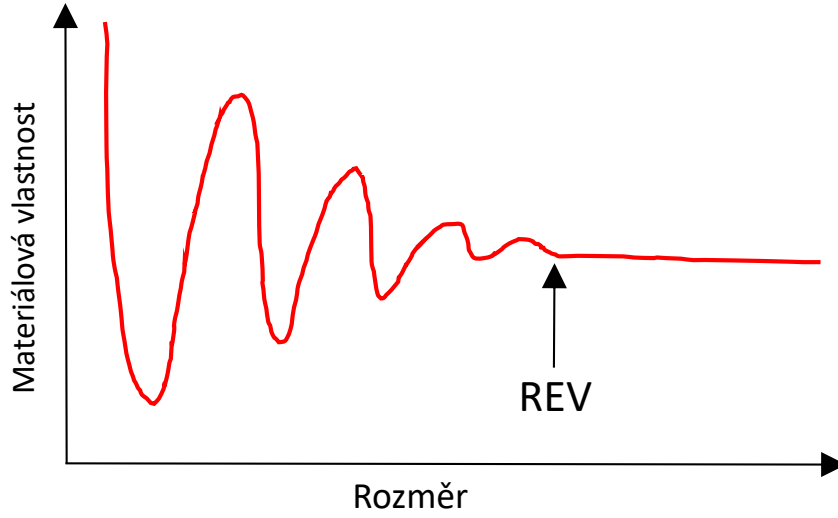


Figure 3.4: Dependence of properties on dimension and REV (inspiration (Min and Jing, 2003))

It is obvious that REV does not have a clearly defined dimension, because we can still approach the exact solution or enlarge REV, but at the cost of higher computational effort and longer computation time. To find the approximate size of REV, a termination criterion based on the so-called relative residual was chosen, which compares  $\tilde{\mathbf{C}}^*$  for two different values of  $n$ . If this residual falls below the chosen tolerance, the calculation is terminated.

A tolerance of 0.1% was considered for all calculations throughout the project..

### 3.2.3 Calculation of the relative residue

Residuum is generally the name for the remainder or balance, which in the context of a project means the remainder of the difference between two consecutive iterations. It is first necessary to define in advance what dimensions we will compare in these iterations. In this project design, it was chosen to compare the effective elastic tensor obtained for dimension  $n$  with the effective elastic tensor calculated for dimension  $m = n + 2$ . For clarity, let us list these tensors in matrix form:

$$\begin{aligned} \text{for } n : \quad \tilde{\mathbf{C}}_n^* &= \begin{bmatrix} C_{1111,n}^* & C_{2211,n}^* & C_{1211,n}^* \\ C_{1122,n}^* & C_{2222,n}^* & C_{1222,n}^* \\ C_{1112,n}^* & C_{2212,n}^* & C_{1212,n}^* \end{bmatrix}, \\ \text{for } m : \quad \tilde{\mathbf{C}}_m^* &= \begin{bmatrix} C_{1111,m}^* & C_{2211,m}^* & C_{1211,m}^* \\ C_{1122,m}^* & C_{2222,m}^* & C_{1222,m}^* \\ C_{1112,m}^* & C_{2212,m}^* & C_{1212,m}^* \end{bmatrix}. \end{aligned} \tag{3.4}$$

Now we define the tensor  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ , which represents the difference between  $\tilde{\mathbf{C}}_n^*$  and  $\tilde{\mathbf{C}}_m^*$  i.e:

$$\mathbf{R} = \begin{bmatrix} r_{1111} & r_{2211} & r_{1211} \\ r_{1122} & r_{2222} & r_{1222} \\ r_{1112} & r_{2212} & r_{1212} \end{bmatrix}, \quad \mathbf{R} = \tilde{\mathbf{C}}_m^* - \tilde{\mathbf{C}}_n^*, \quad (3.5)$$

or

$$r_{ijkl} = C_{ijkl,m}^* - C_{ijkl,n}^*, \quad i, j, k, l = 1, 2. \quad (3.6)$$

At this point we are all set to define the relative residue  $\hat{r}$ , which looks like this:

$$\hat{r} = \frac{\|\mathbf{R}\|}{\|\tilde{\mathbf{C}}_m^*\|}. \quad (3.7)$$

## 4 Program description, functions and documentation

In this chapter, we first describe the prerequisites and steps required to install and use the implemented program for numerical homogenization of mechanical properties, including all auxiliary software (henceforth abbreviated as SW) and programs that were used in the solution. Finally, we discuss the implementation of the program in more detail.

In particular its two possible uses, which are either to calculate for **one chosen**  $n$  or for **sequence**  $n$ , from which the size of REV is then determined.

### 4.1 Prerequisites before using the program

In order to use the implemented program for numerical homogenization, it is first necessary to install or download several software.

- Flow123d - computational simulator (Flow123d, [2024](#))
- GitHub repository - repository with complete code (Siddall, [2024](#))

It will not be described here how to work with or use these programs. Any instructions should be available in their documentation. How these programs/SW were used to solve the problem will be mentioned later.

#### 4.1.1 Booting in Windows OS

Assuming the successful installation of all mentioned software and programs, the implemented program is almost ready for use. The only complication is the need to reinstall some modules or installation packages in Python (e.g. pyvista, sympy, ruamel.yaml) every time the terminal is started. For this reason, it is also necessary to download **Dockerfile** from the aforementioned GitHub, which contains these (or any other) necessary Python packages. Once the Dockerfile is created, a new Docker image needs to be created with it. The full procedure for creating a new Docker image is described below:

- Create a new Docker image:
  - Open a command prompt in the folder where the file is located **Dockerfile**

- Run the following command to create Docker image  
("flow123d\_siddall" is an example of a new image name):

```
docker build -t flow123d_siddall
```

If all the necessary libraries are written in the Dockerfile, just run the Flow123d terminal from the `fterm_siddall.bat` file.

## 4.2 General program structure

First we will look at the general structure of the program in the broadest sense i.e. what are the relationships between the various classes<sup>1</sup>, which are used in the project and what is the overall layout of these classes within the program as a whole. We will then go into a little more detail about each class separately i.e. show in graphs what their inputs, methods and outputs are. Due to the large number of methods that these individual classes have, we will not go into depth on each sub-method. In short, the aim is to describe the structure from the broadest scale and then discuss each class separately for a comprehensive overview of the functionality of the implemented program for calculating numerical homogenization of mechanical properties.

In general, the program could be decomposed into the following parts:

1. User input:

- `config_file.yaml` (+ Related `ConfigManager.py`)
- command line input for `automatic_simulation.py`

2. Program Logic:

- `automatic_simulation.py`
- `main.py`
- `EffectiveElasticTensor.py`
- `GenerateVtu.py`
- `GenerateMesh.py`

3. Output:

- `results_effective_elastic_tensor.txt`
- `results_with_tensors.txt`
- `results_with_residues.txt`

The following page shows a class diagram showing the relationships between classes.

---

<sup>1</sup>To clarify, the classes are named with a capital letter, while the files (configuration and result), startup scripts/programs start with a lower case letter.

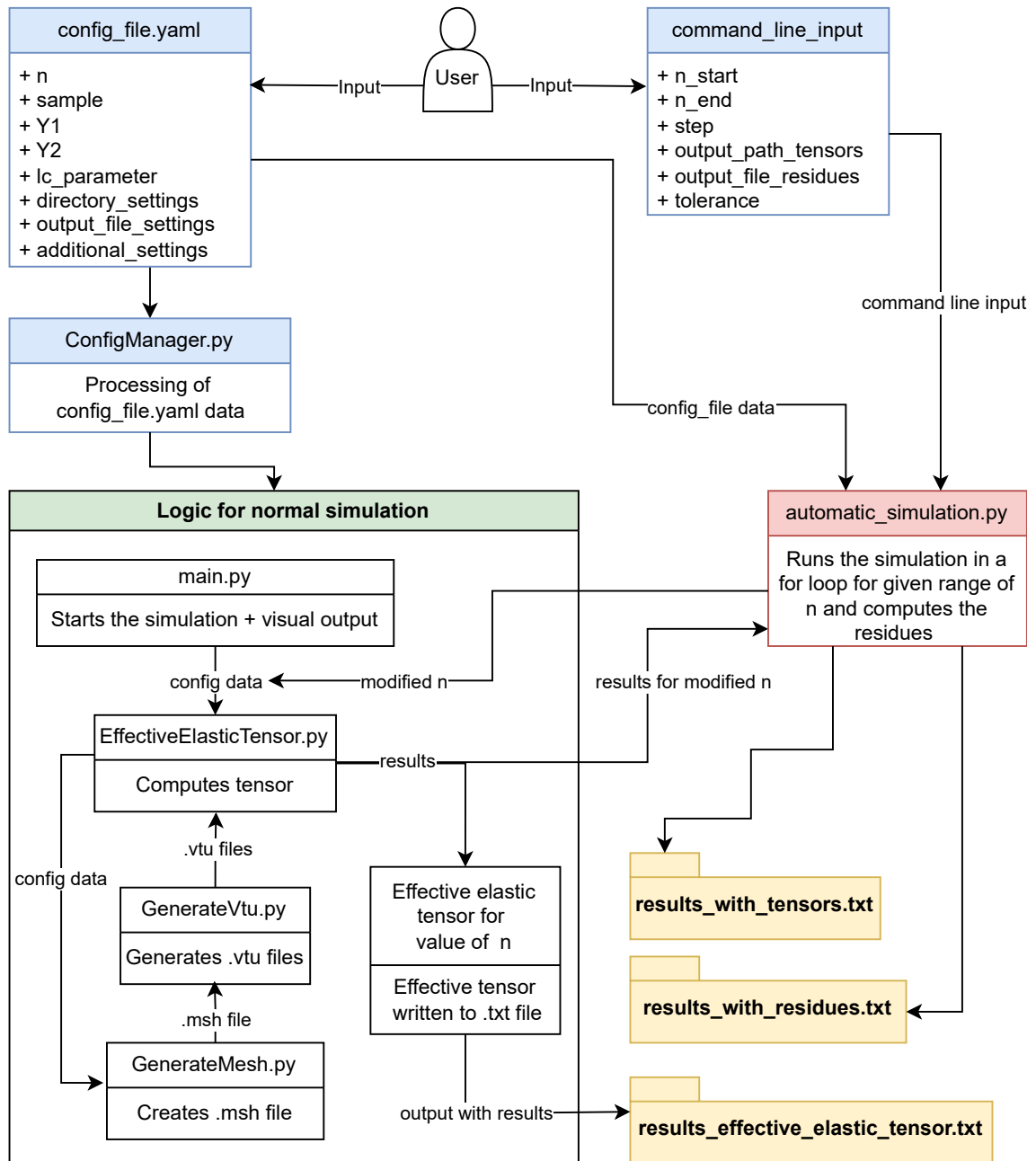


Figure 4.1: Class diagram of the whole program



## 4.3 Description of individual classes and files

### 4.3.1 config\_file.yaml

In this file are defined all user-selectable parameters or options for the simulation, which are:

1. `simulation_parameters`:
  - `n` - defines the dimension of the structure
  - `sample` - indicates whether it is a chessboard or a sandwich
  - `Y1` - Young's modulus for a subset of A
  - `Y2` - Young's modulus for a subset of B
  - `lc_parameter` - parameter affecting the fineness of generated networks
2. `directories`:
  - `absolute_path_to_dir_with_scripts` - the directory from which all other paths below are derived
  - `dir_where_yamls_are_created` - relative path to the directory where the .yaml files are created
  - `path_to_yaml_template` - relative path to the master .yaml file, which must not be deleted or changed
  - `directory_where_vtus_are_created` - relative path to .vtu files
  - `dir_where_mesh_and_geo_are_created` - relative path to the directory where the .msh (or .geo) files are created
3. `results_file_settings`
  - `name_of_file_with_tensor` - name of the simulation results file
  - `output_dir_of_file_with_tensor` - relative path to the results file
4. `additional_settings`
  - `delete_yaml_dir_after_simulation` - toggle whether to delete directory with .yaml files
  - `delete_vtu_dir_after_simulation` - toggle whether to delete directory with .vtu files
  - `change_names_of_computed_yamls` - toggle whether the user wants to choose .yaml file names
  - `new_names_of_yamls` - if the user has chosen to change the names, he defines them here
  - `change_names_of_computed_output_dirs` - toggle whether the user wants to choose the names of the directories with the resulting .vtu files

- `new_names_of_output_dirs` - if the user has chosen to change the names, he defines them here
- `change_name_of_msh_file` - switch to select the name of the `.msh` file
- `new_name_of_mesh` - possible new `.msh` file name
- `create_geo_file` - toggle whether to create a `.geo` file
- `change_name_of_geo_file` - toggle whether to choose the name of the `.geo` file
- `new_name_of_geo` - if the user has chosen to change the name, he defines it here
- `measure_time_of_computation` - switch whether the user wants to measure the time

### 4.3.2 GenerateMesh.py

Based on the input data of the configuration file, this class automatically creates the geometry, i.e. points, boundaries, areas, subareas and finally the computational mesh in `.msh` format, which is then returned.

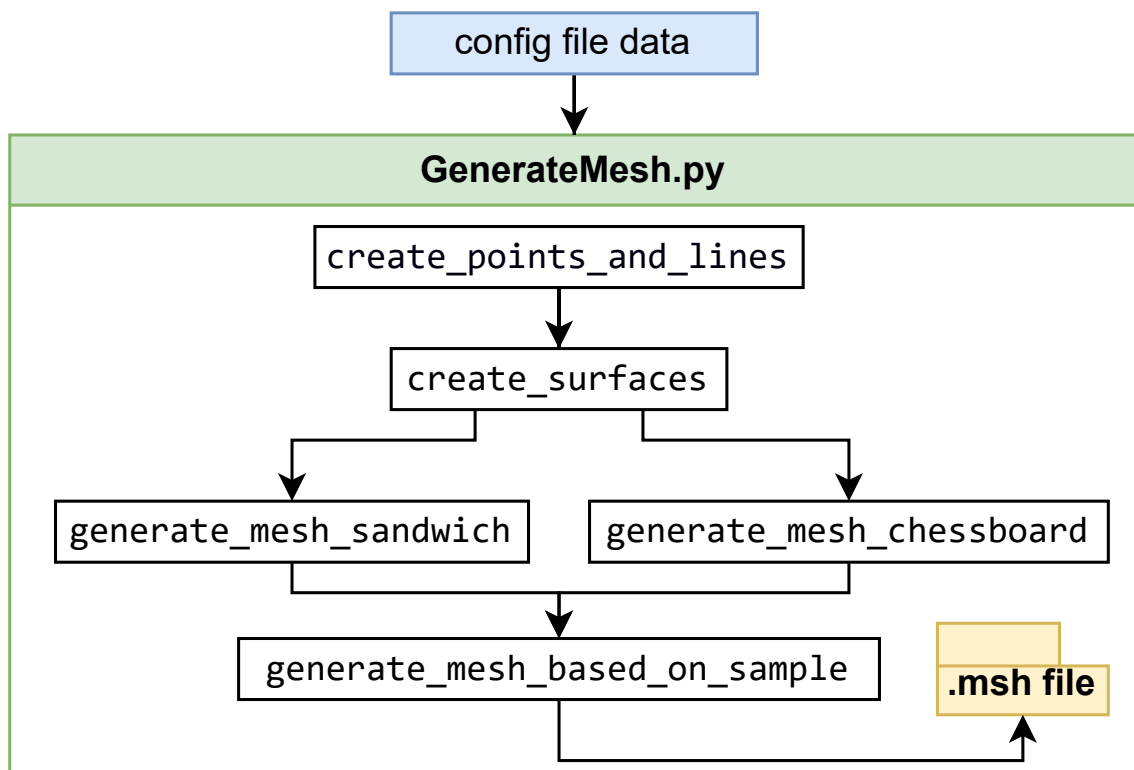


Figure 4.2: Visualisation of how methods work in the classroom `GenerateMesh.py`

### 4.3.3 ConfigManager.py

This class handles all parameters selected by the user in `config_file.yaml` and then returns them.

### 4.3.4 GenerateVtuFiles.py

The main goal of this class is to use an already generated `.msh` file and the necessary data from `config_file.yaml` to create relevant `.yaml` files from a template, which are then used to solve the solution using the Flow123d computational simulator. This results in 3 `.vtu` files for each load type:

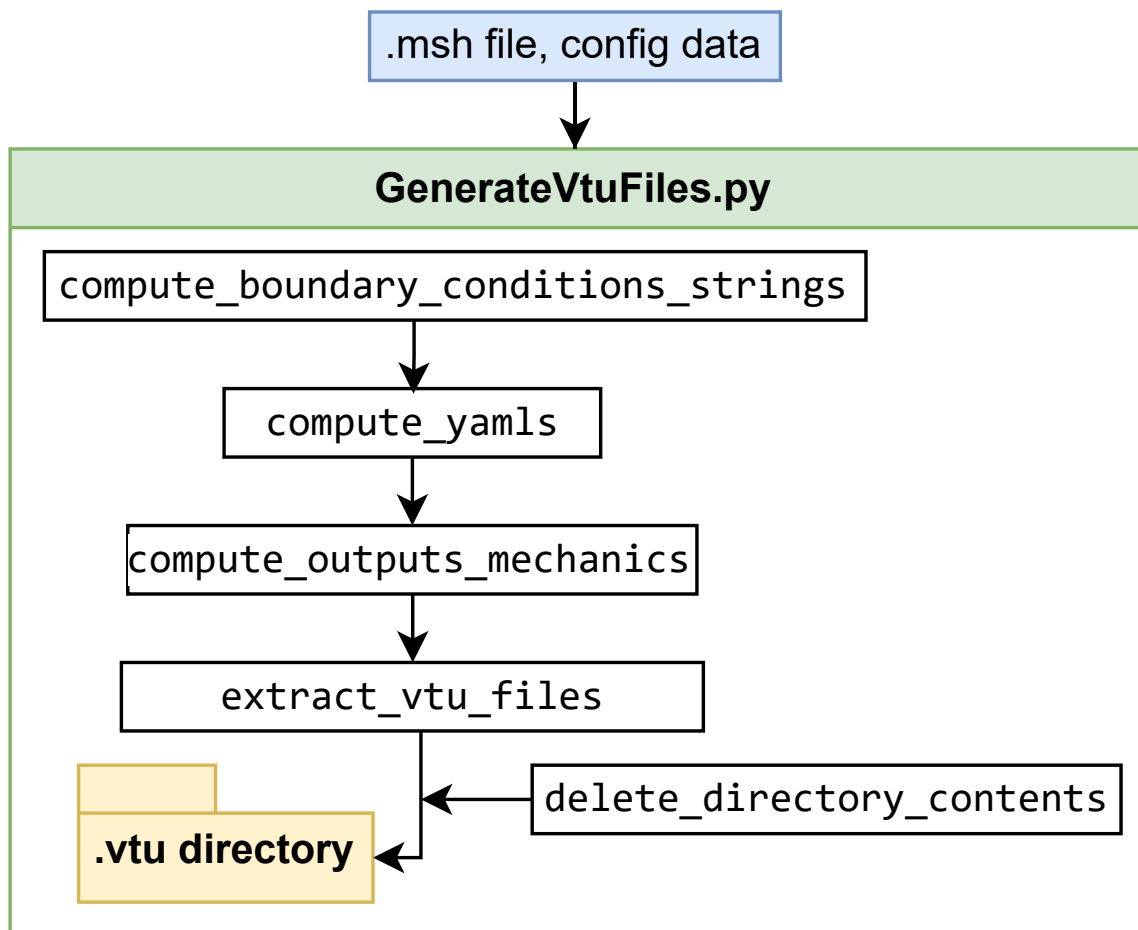


Figure 4.3: Visualisation of how methods work in the class `GenerateVtuFiles.py`

### 4.3.5 EffectiveElasticTensor.py

In this class, the input is the created `.vtu` files and the necessary data from the `config_file.yaml` file. Then the whole simulation process is actually completed, i.e. the resulting elements of the effective elastic tensor are obtained in this class. The resulting tensor is written to a text file, but as can be seen in the diagram

below, it depends on whether the user performs the simulation using `main.py` or `automatic_simulation.py`. If the former is chosen, the resulting tensor is written once to the file chosen by the user, but in the latter case, individual tensors are added incrementally for the entire range of dimensions that the user has specified in the command line.

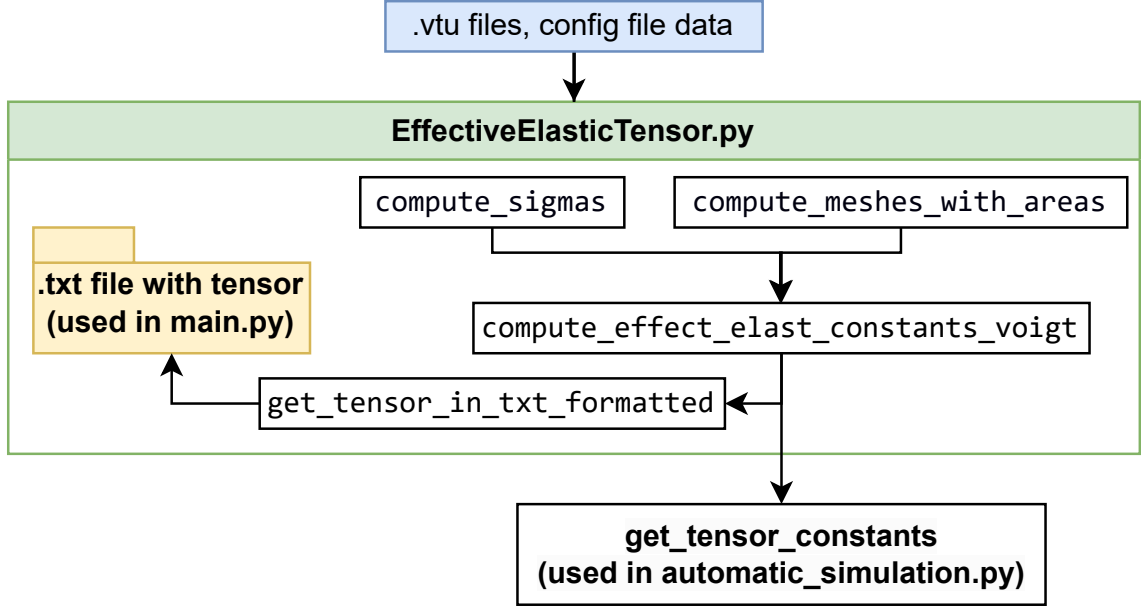


Figure 4.4: Visualisation of how methods work in the class `EffectiveElasticTensor.py`

Because of its importance, it is necessary to go into a little more detail about the `compute_effect_elast_constants_voigt` method. This method is crucial because it calculates all the effective elastic constants themselves, which are then converted into Voigt notation.

The method starts by creating an empty list `constants_voigt_list`, into which the calculated constants are successively stored.

Then the `compute_sigmas` and `compute_meshes_with_areas` methods are used to obtain the stresses (sigmas) and areas of the elements (areas).

Then, in the loop, each stress value is multiplied by the area of the element and the results are stored in the `stress_area` list. Once the loop is complete, all values in the `stress_area` list are added together and this sum is then divided by the total area of the area or the sum of all sub-elements. These resulting constant values are selected according to the indices  $[0, 4, 1]$ . The reason for these indices is as follows: There are 27 effective elastic constants in the list `all_constants`. For each load matrix we get 9 constants, 5 of which are zero. The remaining 4 constants form a symmetric matrix. Thus, we are only interested in the unique elements with indices  $[0, 4, 1]$  for a given load matrix, which are the results forming one of the columns of the resulting effective elastic tensor. The constants are then stored in the resulting list `constants_voigt_list`, which is finally returned as the output of the method.

The code for the method can be found in the GitHub repository, i.e. (Siddall,

2024)) or in the preview below:

```
def compute_effect_elast_constants_voigt(self):
    constants_voigt_list = []
    sigmas = self.compute_sigmas()
    areas = self.compute_meshes_with_areas()
    amount_of_meshes = len(self.meshes)

    for i in range(amount_of_meshes):
        sigma = sigmas[i]
        area = areas[i]
        stress_area = []
        for j in range(len(sigma)):
            stress_area.append(sigma[j, :] * area[j])

        all_constants = np.sum(stress_area, axis=0) / np.sum(area)
        needed_indexes = [0, 4, 1]
        constants_voigt = [all_constants[index] for index in needed_indexes]
        constants_voigt_list.append(constants_voigt)
    return constants_voigt_list
```

Figure 4.5: Detail of the method for calculating the effective elastic constants

The last thing that is important to discuss in terms of program functionality is the difference between the `main.py` and `automatic_simulation.py` startup scripts, which will be discussed in the next two chapters.

### 4.3.6 main.py

`Main.py` is used to initialize a **single** simulation just for a **single set** of data. This data is exclusively data from the `config_file.yaml` configuration file. In this configuration file, the user can set all the necessary parameters before running the simulation and then simply run the script in the Flow123d terminal as follows:

```
<python3> <main.py> <config_file.yaml>
```

The output of this script is only the **computed effective tensor** based on the input data, which is located in a text file that is in the appropriate directory that the user has set up before the simulation. The vast majority of the optional parameters in `config_file.yaml` are for **exclusively for modifying the output of main.py**, since no parameters from the configuration file are used when using `automatic_simulation.py`. The only exceptions that also need to be set for `automatic_simulation.py` are `sample`, `Y1`, `Y2` and `lc_parameter`, but more on that in the next section.

### 4.3.7 automatic\_simulation.py

`Automatic_simulation.py` is used to initialize **several consecutive automatically run simulations**. The command is run as follows:

```
<python3> <automatic_simulation.py> <config_file.yaml> <n_start> ...  
... <n_end> <step> <output_tensors> <output_residues> <tolerance>
```

Before running the script it is necessary to set `config_file.yaml` in `sample`, `Y1`, `Y2` and `lc_parameter`, which will be used for all simulations. Once this is set, the remaining parameters need to be entered:

- `n_start` - specifies the initial `n` at which to start
- `n_end` - specifies the final `n` where the script ends
- `step` - specifies the step with which `n` increases
- `output_tensors` - absolute path to the text file with the stored tensors
- `output_residues` - absolute path to the text file with stored residues
- `tolerance` - tolerance for numerical calculations

The way `automatic_simulation.py` works is that at each iteration, creates a new temporary configuration file with the given `n` (actually, a temporary `.yaml` configuration file is created for each `n` value in the specified range), which is then used for the new simulation. The relative residual is then computed for two consecutive iterations or two consecutive tensors. Among other things, these temporary files are continuously deleted.

In other words, `automatic_simulation.py` is used to automatically control several successive simulations with some given range of dimensions, step and tolerance. It checks whether the relative residuals have fallen below a given tolerance, if so, it prints out for which dimension or optimal `n` convergence has been achieved (the same is then also in the result file) and the process is automatically terminated. If it was not possible to fall below the tolerance, then the results have not yet reached sufficient precision and hence REV.

The resulting tensors for a given dimension `n` are written column by column to a text file. The files with the relative residuals and the resulting tensors are continuously overwritten during the run of the program, i.e. we do not lose the computed results if there is a manual or sudden interruption of the program.

A better demonstration of what some specific calculations and outputs look like (both in the terminal and in text files), for the given data, is shown in the following chapter, where we show the complete calculation procedure for two specific examples.

## 5 Calculations of sample problems

The aim of this chapter is to fully describe and visualize two example simulations. First, we will show how one sample calculation for a one-time simulation (`main.py`) looks like and in the second part of the chapter we will show a second example for an automatic simulation (`automatic_simulation.py`).

### 5.1 Example of a one-off simulation

In general, the procedure can be summarized as follows; first, a simple checkerboard geometry is created for the specified dimension. Then a mesh is created on the given geometry. The process continues by automatically creating three different `.yaml` files for each load type with corresponding parameters `Y1` and `Y2` for the  $x$ -axis thrust. This is followed by a simulation using `Flow123d`, which returns three different `.vtu` files, visualizing the  $x$ -axis tension, from which the resulting effective elastic tensor is then calculated. Finally, a sample of the output `main.py`.

The task that is solved here has this specific data in the configuration file (the directories listed in `directories` are the specific directories where the code was run):

1. `simulation_parameters`:
  - `n`: 3
  - `sample`: "chessboard"
  - `Y1`: 100
  - `Y2`: 50
  - `lc_parameter`: 0.16
2. `directories`:
  - `absolute_path_to_dir_with_scripts`:  
"/C/Plocha/Semestral\_project/Python\_scripts/"
  - `dir_where_yamls_are_created`: "data\_vtu/yamls\_dir"
  - `path_to_yaml_template`:  
"data\_vtu/TEMPLATE\_DO\_NOT\_DELETE/template.yaml"
  - `directory_where_vtus_are_created`: "data\_vtu/vtu\_files"

- `dir_where_mesh_and_geo_are_created`: "data\_vtu/"
3. `results_file_settings`
- `name_of_file_with_tensor`: "example\_tensor"<sup>1</sup>
  - `output_dir_of_file_with_tensor`: "results\_elastic\_tensor"
4. `additional_settings`
- `delete_yaml_dir_after_simulation`: "no"
  - `delete_vtu_dir_after_simulation`: "no"
  - `change_names_of_computed_yamls`: "no"
  - `new_names_of_yamls`: ["name1.yaml", "name2.yaml", "name3.yaml"]
  - `change_names_of_computed_output_dirs`: "no"
  - `new_names_of_output_dirs`: ["name1", "name2", "name3"]
  - `change_name_of_msh_file`: "yes"
  - `new_name_of_mesh`: "example.msh"
  - `create_geo_file`: "yes"
  - `change_name_of_geo_file`: "yes"
  - `new_name_of_geo`: "example.geo\_unrolled"
  - `measure_time_of_computation`: "yes"

For the sake of completeness, let us state the boundary conditions of this problem. If we apply the load matrices defined in (3.1), for each displacement vector, see (2.20), we get:

$$\begin{aligned} \mathbf{u}_1 = \mathbf{E}_1 \cdot \mathbf{x} &\iff \mathbf{u}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}, \\ \mathbf{u}_2 = \mathbf{E}_2 \cdot \mathbf{x} &\iff \mathbf{u}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ x_2 \end{bmatrix}, \\ \mathbf{u}_3 = \mathbf{E}_3 \cdot \mathbf{x} &\iff \mathbf{u}_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}. \end{aligned}$$

Verbally speaking, the first load represents stretching in the  $x$  axis, the second load simulates stretching in the  $y$  axis and the third load represents the stretching in shear in the  $x, y$  axes. These boundary conditions or displacement vectors  $\mathbf{u}$  are used to define the boundary conditions in the individual `.yaml` files.

An analogous procedure for these calculations and the overall procedure would apply to the „sandwich” structure. Below are several slides illustrating some of the sub-steps.

---

<sup>1</sup>The program defaults to creating `.txt` files, there is no need to add an extension.



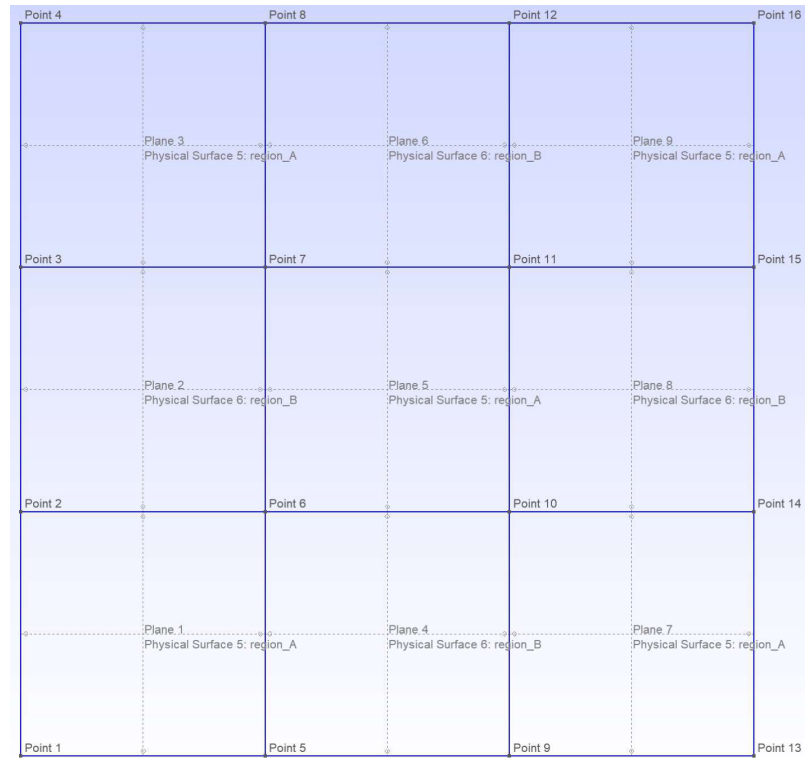


Figure 5.1: Created geometry example.geo\_unrolled for chapter example 5.1

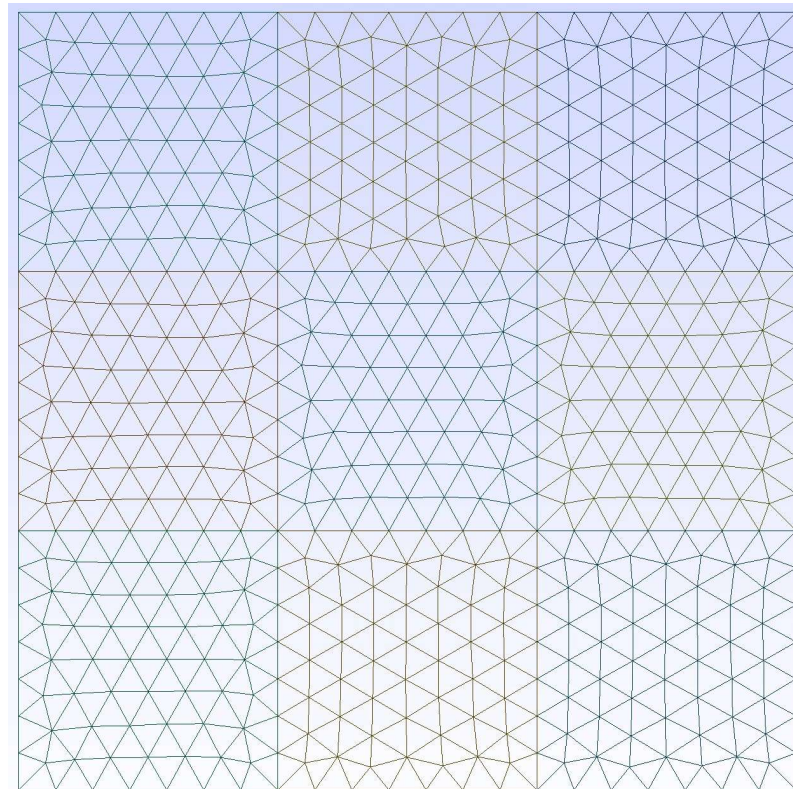


Figure 5.2: Generated mesh for the chapter example 5.1

```

# Author: Ronald Ch. Siddall

flow123d_version: 3.1.0
problem: !Coupling_Sequential
  description: 2D linear elastic Dirichlet BC problem.
  mesh:
    mesh_file: /C/Plocha/Semestral_project/Python_skripts/data_vtu/example.msh
  flow_equation: !Coupling_Iterative
    input_fields:
      time:
        end_time: 3

    flow_equation: !Flow_Darcy_LMH
      input_fields:
        output:
          fields: []

  mechanics_equation:
    output_stream:
      file: mechanics.pvd
      format: !vtk
    output:
      fields:
        - displacement
        - stress
        - displacement_divergence
        - region_id
    solver: !Petsc
      a_tol: 1e-20
      r_tol: 1e-15
    input_fields:
      - region: region_A
        young_modulus: 100
        poisson_ratio: 0

      - region: region_B
        young_modulus: 50
        poisson_ratio: 0

      - region: .BOUNDARY
        bc_type: displacement
        bc_displacement: !FieldFormula
          value: [x,0,0]

```

Figure 5.3: Created .yaml file for  $u_1$  for the chapter sample 5.1

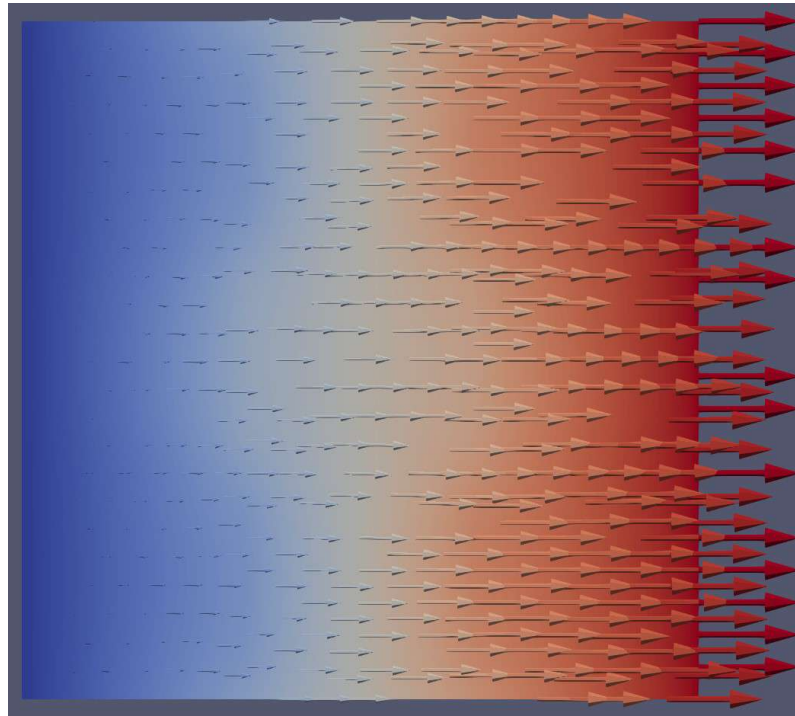


Figure 5.4: Visualization of displacement vectors for the first load in ParaView for an example from the chapter 5.1

```

example_tensor.txt
Soubor  Upravit  Zobrazit

Settings of the simulation:
- n: 3
- type of geometry: chessboard
- Y1: 100
- Y2: 50
=====
Effective elastic tensor in matrix form for 2D problems
=====

C =
    73.81537919579347    0.7197680236474644   -0.00012916870045901304
    0.7197680236473514    73.81396535529225    1.9395047606476615e-05
    -0.00012916870025809276    1.9395047286411653e-05    37.093813130071304

-----
This result was computed using these files:
/C/Plocha/Semestral_project/Python_skripts/data_vtu/vtu_files/1_mechanics-00000.vtu
/C/Plocha/Semestral_project/Python_skripts/data_vtu/vtu_files/2_mechanics-00000.vtu
/C/Plocha/Semestral_project/Python_skripts/data_vtu/vtu_files/3_mechanics-00000.vtu

```

Figure 5.5: Representation of the calculated effective elastic tensor for an example from the chapter 5.1

## 5.2 Example of automatic simulation

The sample automatic simulation was performed with the same configuration file settings as in the previous chapter and for the following input data:

- n\_start: 3
- n\_end: 7
- step: 2
- output\_tensors: "/C/Plocha/results\_tensors.txt"
- output\_residues: "/C/Plocha/results\_residues.txt"
- tolerance: 0.001

```
=====
                        AUTOMATIC SIMULATION STARTED
=====
Settings of the simulation:
- Type of geometry: chessboard
- Y1: 100
- Y2: 50
- n: 3 to 7
- step: 2
- tolerance: 0.001
- output with tensors: /C/Plocha/results_tensors.txt
- output with residues: /C/Plocha/results_residues.txt
-----
Simulation started:2024-05-07 17:44:34
Progress status of the simulation:
-----
2024-05-07 17:44:44: (1/2) - 50.00% completed
2024-05-07 17:44:58: (2/2) - 100.00% completed
-----
Optimal n not found within the given tolerance
Tolerance: 0.001
=====
```

Figure 5.6: Ukázka výstupu automatic\_simulation.py pro ukázkovou úlohu z kapitoly 5.2

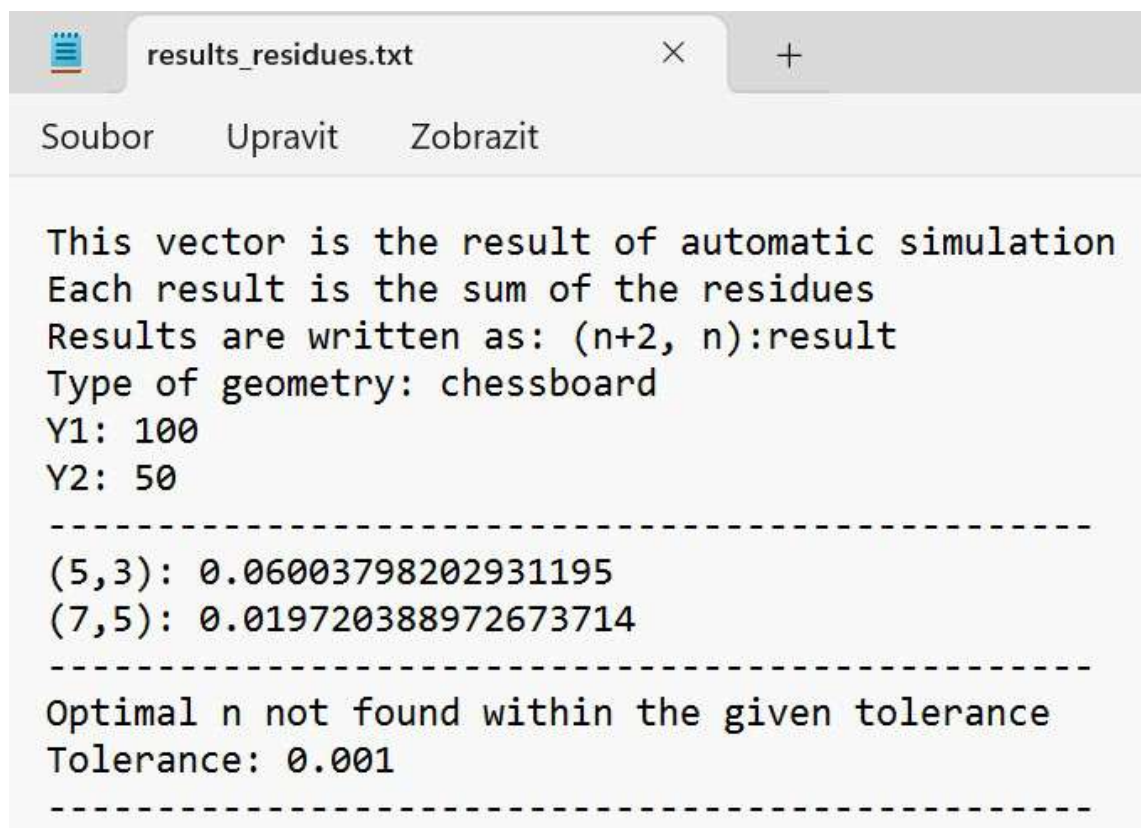
As you can see from the image 5.6, the output first lists the settings of the simulation and the paths to the result files. Then it continuously prints information about how the simulation itself is running, i.e. it prints the time when each sub-simulation/iteration ran. In this sample case, convergence was not achieved as the dimensions specified are too small.

For completeness, the following figure shows an example of the output of `automatic_simulation.py` when convergence has occurred within the specified range of dimensions.

```
-----
Optimal n: 21
Result for optimal n: 0.00093871256304287
Tolerance: 0.001
-----
```

Figure 5.7: Example of the last part of the output if convergence has been achieved

Figure 5.8 shows a detail of a file containing relative residues.



The screenshot shows a text editor window with the title bar 'results\_residues.txt'. The menu bar contains 'Soubor', 'Upravit', and 'Zobrazit'. The text content is as follows:

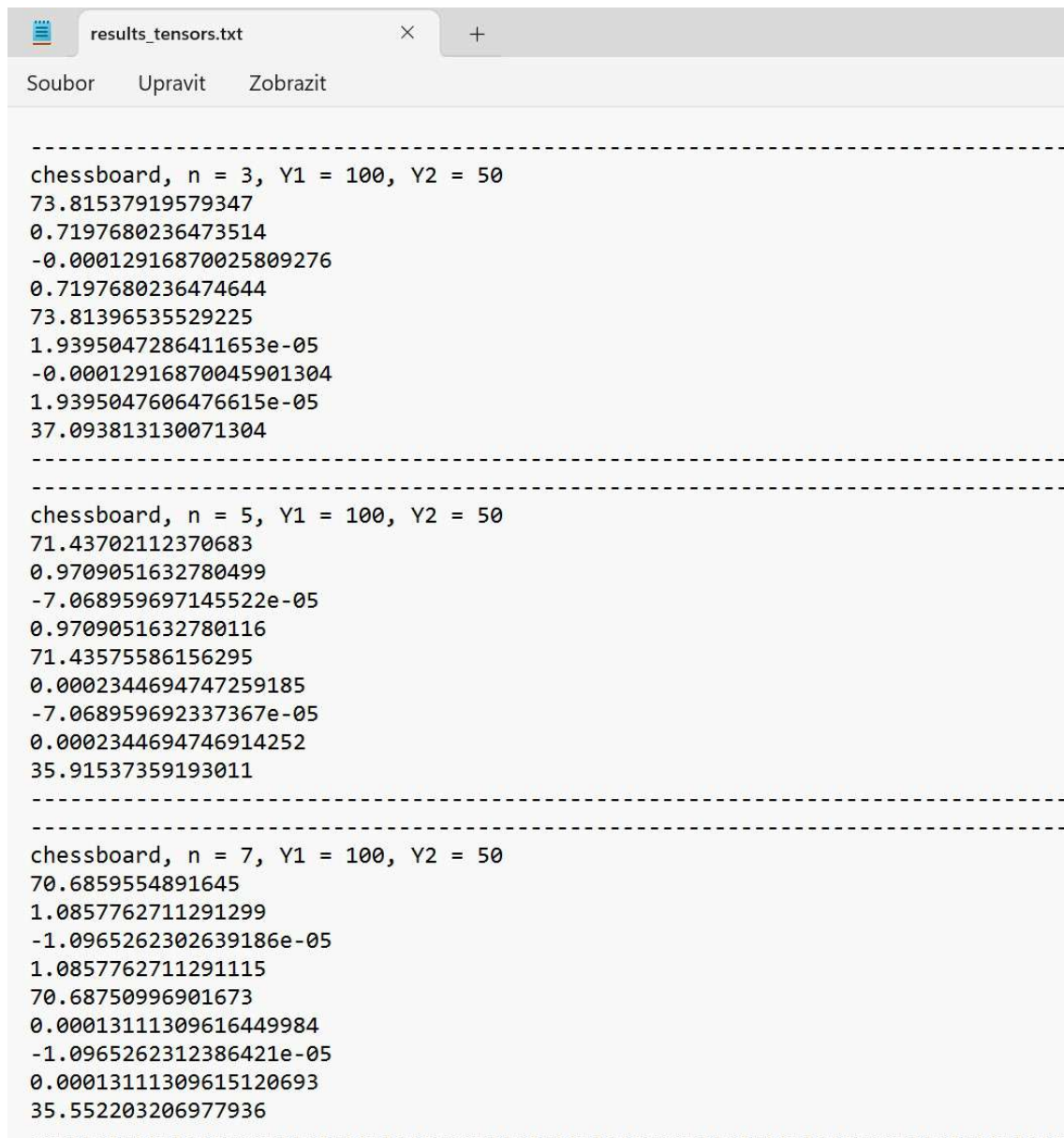
```
This vector is the result of automatic simulation
Each result is the sum of the residues
Results are written as: (n+2, n):result
Type of geometry: chessboard
Y1: 100
Y2: 50
-----
(5,3): 0.06003798202931195
(7,5): 0.019720388972673714
-----
Optimal n not found within the given tolerance
Tolerance: 0.001
-----
```

Figure 5.8: Sample file with relative residuals from the specified range for the sample problem from the chapter 5.2

The 5.9 image shows a file with the results of the automatic simulation. Note



the identical result for  $n = 3$ . The solution is the same as for the sample problem in Chap. 5.1.



```

-----
chessboard, n = 3, Y1 = 100, Y2 = 50
73.81537919579347
0.7197680236473514
-0.00012916870025809276
0.7197680236474644
73.81396535529225
1.9395047286411653e-05
-0.00012916870045901304
1.9395047606476615e-05
37.093813130071304
-----

chessboard, n = 5, Y1 = 100, Y2 = 50
71.43702112370683
0.9709051632780499
-7.068959697145522e-05
0.9709051632780116
71.43575586156295
0.0002344694747259185
-7.068959692337367e-05
0.0002344694746914252
35.91537359193011
-----

chessboard, n = 7, Y1 = 100, Y2 = 50
70.6859554891645
1.0857762711291299
-1.0965262302639186e-05
1.0857762711291115
70.68750996901673
0.00013111309616449984
-1.0965262312386421e-05
0.00013111309615120693
35.552203206977936
-----

```

Figure 5.9: Example of the resulting file with tensors from the specified range for the sample problem from the chapter 5.2

## 6 Results of the calculations performed

In this chapter we will discuss the results of all the calculations performed. We start by defining what cases were calculated and what their parameters were. Next, we list the computed results of each calculation, which are the value of the dimension where convergence occurred and the associated tensor that came out for that dimension. We then show the various dependencies in graphs that are supported by the measured data. This chapter concludes with a summary of all the findings and observations on how different types and sizes of geometries behave in the result for different values of Young's modulus.

### 6.1 Summary of all calculations performed

While performing the individual measurements, it was found that there is quite a significant difference in convergence between even and odd multiples of  $n$ . That is, in general, the relative residuals for odd multiples of  $n$  always came out larger compared to even multiples of  $n$ , which is due to the different distribution of subregions A, B for odd and even multiples of  $n$ . This distribution ensured a monotonic decrease in the relative residuals separately for even and odd multiples of  $n$ .

A total of **16 different cases** were analysed, which are:

a) **chessboard**:

- 1) for even multiples of  $n$ ,  $Y1 = 100$ ,  $Y2 = 50$
- 2) for odd multiples of  $n$ ,  $Y1 = 100$ ,  $Y2 = 50$
- 3) for even multiples of  $n$ ,  $Y1 = 200$ ,  $Y2 = 50$
- 4) for odd multiples of  $n$ ,  $Y1 = 200$ ,  $Y2 = 50$
- 5) for even multiples of  $n$ ,  $Y1 = 500$ ,  $Y2 = 50$
- 6) for odd multiples of  $n$ ,  $Y1 = 500$ ,  $Y2 = 50$
- 7) for even multiples of  $n$ ,  $Y1 = 2500$ ,  $Y2 = 50$
- 8) for odd multiples of  $n$ ,  $Y1 = 2500$ ,  $Y2 = 50$

b) **Sandwich:**

- 9) for even multiples of  $n$ ,  $Y1 = 100$ ,  $Y2 = 50$
- 10) for odd multiples of  $n$ ,  $Y1 = 100$ ,  $Y2 = 50$
- 11) for even multiples of  $n$ ,  $Y1 = 200$ ,  $Y2 = 50$
- 12) for odd multiples of  $n$ ,  $Y1 = 200$ ,  $Y2 = 50$
- 13) for even multiples of  $n$ ,  $Y1 = 500$ ,  $Y2 = 50$
- 14) for odd multiples of  $n$ ,  $Y1 = 500$ ,  $Y2 = 50$
- 15) for even multiples of  $n$ ,  $Y1 = 2500$ ,  $Y2 = 50$
- 16) for odd multiples of  $n$ ,  $Y1 = 2500$ ,  $Y2 = 50$

Throughout the rest of the paper, we will refer to each case either by these numerical designations (1-16) or, alternatively, abbreviated by the ratio of  $Y1$  to  $Y2$ , i.e. the ratio.

## 6.2 Final results of calculations performed

Here we simply list the results for each case, which are the computed tensors whose change in relative residual is less than the tolerance of 0.1% for the mentioned dimension. In the following sections, the individual effective elastic tensors for all 16 cases are listed, with the elements listed in the following order from above:  $C_{1111}^*$ ,  $C_{1122}^*$ ,  $C_{1112}^*$ ,  $C_{2211}^*$ ,  $C_{2222}^*$ ,  $C_{2212}^*$ ,  $C_{1211}^*$ ,  $C_{1222}^*$ ,  $C_{1212}^*$ . Finally, there are also two tables summarizing the dimensions where convergence occurred for all cases.

### 6.2.1 Chessboard, $Y1 = 100$ , $Y2 = 50$

1) REV for  $n = 18$

69.77790412852876  
1.268966397406456  
0.0015400102086822543  
1.2689663974064747  
69.77883933121191  
0.0016499129583467062  
0.0015400102086773626  
0.0016499129583499147  
35.123542320394286

2) REV for  $n = 21$

69.7974325270019  
1.2851838850191901  
-3.375389169509585e-05  
1.285183885019233  
69.79814363461671  
2.3699074542096177e-05  
-3.3753891691373876e-05  
2.3699074524563276e-05  
35.13635599256736



### 6.2.2 Chessboard, $Y1 = 200$ , $Y2 = 50$

3) REV for  $n = 32$

95.4942863782584  
6.656104637850853  
0.005186057016212931  
6.656104637850688  
95.49868643195286  
0.005215034742229218  
0.005186057016260705  
0.0052150347422039904  
49.206199367922125

4) REV for  $n = 35$

95.49456610791185  
6.679170336660807  
9.452142842176485e-05  
6.6791703366609285  
95.49745108473925  
-4.876211455186902e-05  
9.452142840686483e-05  
-4.876211463879784e-05  
49.20988234501693

### 6.2.3 Chessboard, $Y1 = 500$ , $Y2 = 50$

5) REV for  $n = 46$

149.58438163454568  
24.686233344953514  
0.014817396738947754  
24.686233344953838  
149.59112513217812  
0.014350614316744005  
0.014817396738886173  
0.014350614316633963  
81.55103665042539

6) REV for  $n = 47$

149.64439418609484  
24.695220464874005  
-0.00013076264218503112  
24.695220464873877  
149.64674457689802  
-0.0001013015961663917  
-0.00013076264236725653  
-0.00010130159631682866  
81.58496382152258

### 6.2.4 Chessboard, $Y1 = 2500$ , $Y2 = 50$

7) REV for  $n = 56$

448.7538921227705  
140.7095943919707  
0.07791784541829024  
140.70959439197912  
448.7620760445914  
0.07456287673995386  
0.07791784541861534  
0.07456287674009215  
271.843579476085

8) REV for  $n = 59$

448.7339575220418  
140.79816572557928  
0.0011317897175975495  
140.79816572558238  
448.7337631201216  
-0.0021087192578492474  
0.001131789718711193  
-0.002108719257941151  
271.8367569322998

### 6.2.5 Sandwich, $Y1 = 100$ , $Y2 = 50$

9) REV for  $n = 14$

74.99847398140356  
-1.8652219694723548e-06  
9.379221466503778e-09  
-1.8652219623201964e-06  
66.95726604003637  
6.284790911036168e-06  
9.37921262326113e-09  
6.284790825319649e-06  
33.607666763522374

10) REV for  $n = 35$

75.71366861854767  
-2.12962485752517e-06  
-9.057722796937459e-10  
-2.129624893284967e-06  
67.41017637236148  
-2.0214943893211664e-06  
-9.057388262221473e-10  
-2.0214943622162866e-06  
33.75199014276259

### 6.2.6 Sandwich, $Y1 = 200$ , $Y2 = 50$

11) REV for  $n = 24$

124.9985145125482  
-3.4044149069349482e-06  
-6.816667895814417e-09  
-3.4044148744409204e-06  
80.81139931837016  
-3.3183225646630386e-05  
-6.816549437608952e-09  
-3.318322585535557e-05  
40.72025010831128

12) REV for  $n = 49$

126.52987545600318  
-3.371304213952673e-06  
5.8145736437936237e-11  
-3.3713043170585957e-06  
81.3700253890178  
-3.1376314452288377e-06  
5.835365329858265e-11  
-3.137631512646047e-06  
40.83068228679981

### 6.2.7 Sandwich, $Y1 = 500$ , $Y2 = 50$

13) REV for  $n = 32$

274.9975478691609  
-7.607953550492294e-06  
4.196254145044891e-10  
-7.607953767531503e-06  
93.18302310854162  
-5.9475703427960495e-05  
4.198236537246162e-10  
-5.947570449568044e-05  
47.14168882592856

14) REV for  $n = 57$

278.9459720317214  
-6.764279305520123e-06  
1.3702033346499766e-09  
-6.764279319092109e-06  
93.50080575801393  
-9.295384282596763e-06  
1.3694921969209712e-09  
-9.295384394406406e-06  
47.04557919993169

## 6.2.8 Sandwich, $Y1 = 2500$ , $Y2 = 50$

15) REV for  $n = 34$

1274.9892989500217  
-4.201983699667877e-05  
-1.2066016661293384e-08  
-4.201983625644647e-05  
110.91815212605339  
0.0001425361974907446  
-1.206623923450982e-08  
0.00014253619142419577  
55.202924641297024

16) REV for  $n = 57$

1296.4847385718072  
-3.2121213682415905e-05  
8.458767170776745e-09  
-3.212121229496166e-05  
107.53471487023648  
-3.9190121007972586e-05  
8.456839269883612e-09  
-3.919012748561843e-05  
53.56243516868323

## 6.2.9 Tables with overview of REV dimension values

For the sake of clarity, let's write down the REV values for the checkerboard and the sandwich in tables.

Ratio	Even $n$	Odd $n$
2	18	21
4	32	35
10	46	47
50	56	59

Table 6.1: Summary of REV dimensions for the chessboard

Ratio	Even $n$	Odd $n$
2	14	35
4	24	49
10	32	57
50	34	57

Table 6.2: Summary of REV dimensions for the sandwich

## 6.3 Graphs of different dependencies

Figures 6.1 and 6.2 show the dependence of the relative residual on the dimension, i.e. the decrease of the relative residual with increasing values of  $n$ . This dependence is only shown for even multiples of  $n$ , but essentially equivalent results follow for odd multiples, so there is no need to add both.

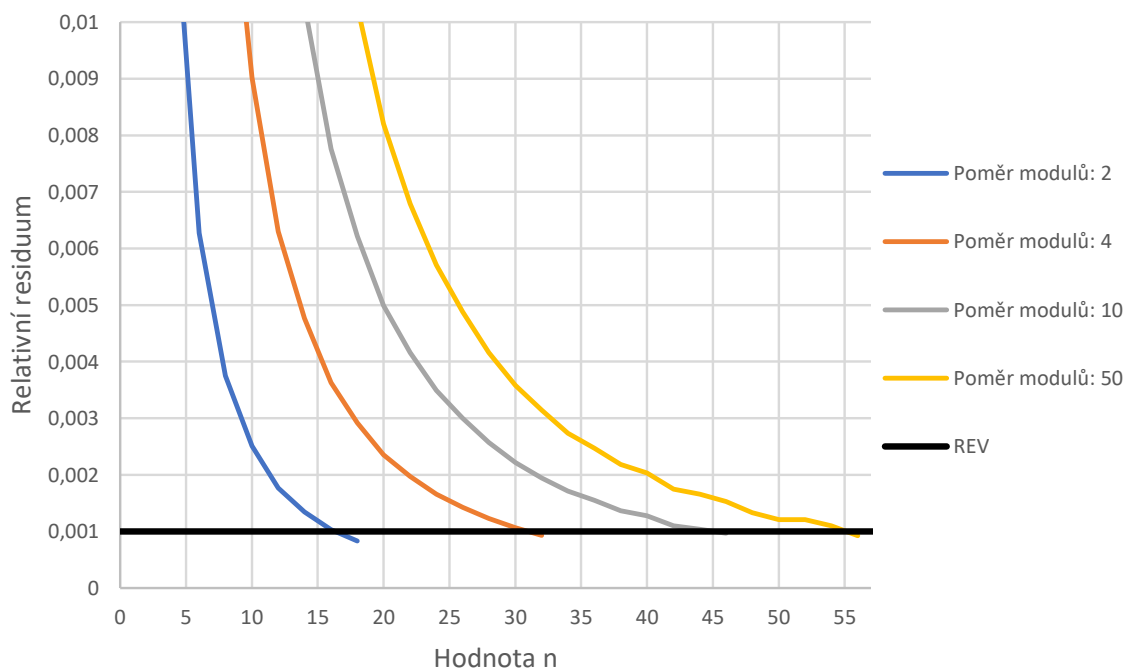


Figure 6.1: Chessboard: dependence of the rel. residue on the value of the REV dimension for even n

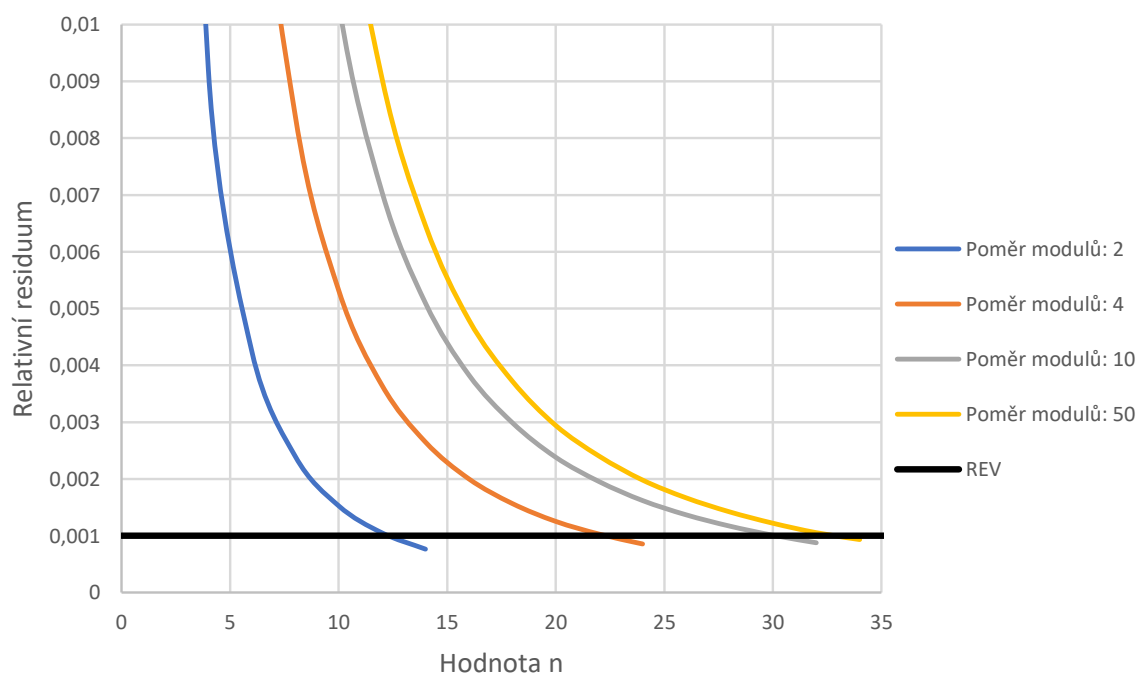


Figure 6.2: Sandwich: dependence of the rel. residue on the value of the REV dimension for even n

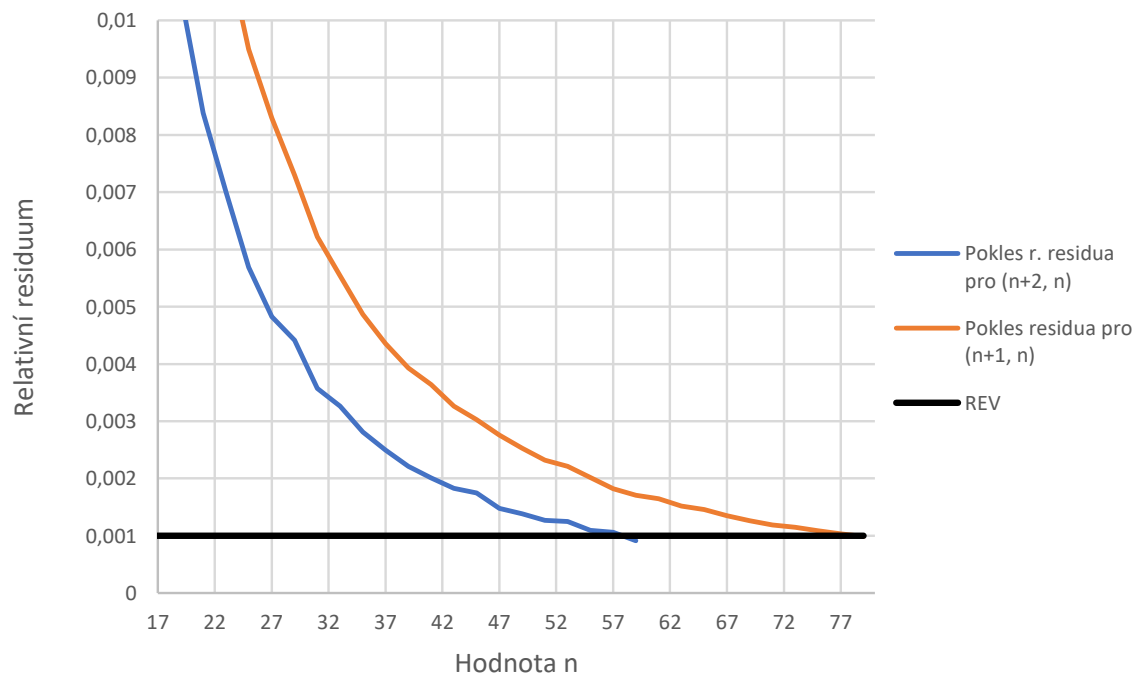


Figure 6.3: Comparison of the rate of decrease of the rel. residue with respect to the value of  $n$  when comparing  $(n+2, n)$  and  $(n+1, n)$  for the chessboard - odd multiples - modulus ratio 50

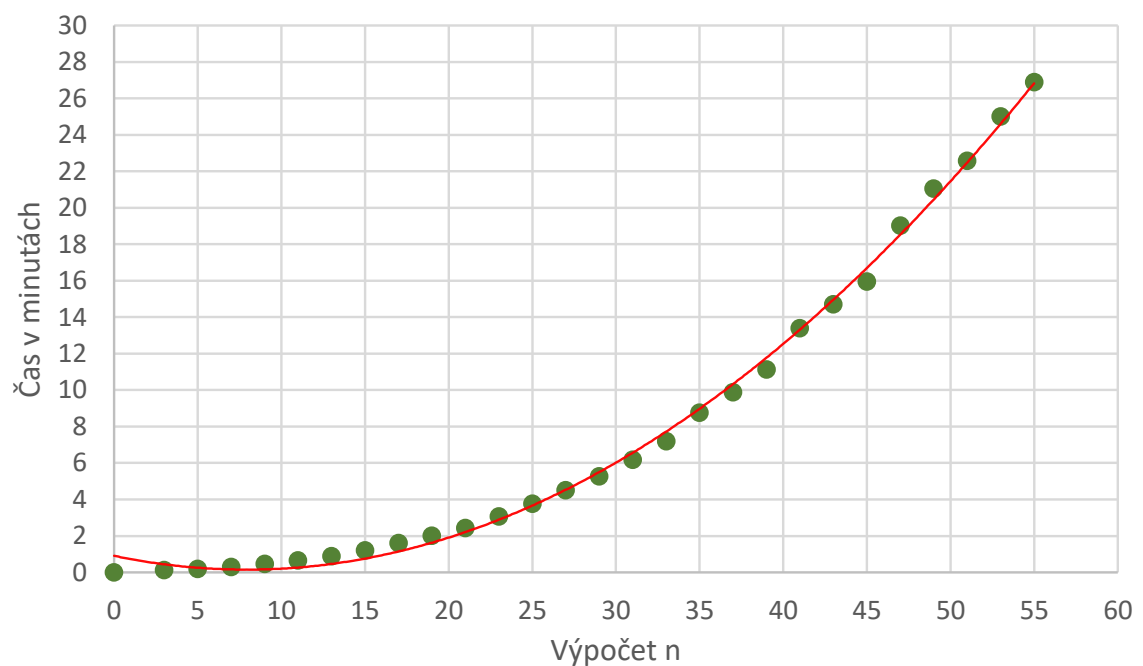


Figure 6.4: Graph of the dependence of the time required for convergence on the dimension - for a chessboard - modulus ratio 50 - the red dotted curve is a quadratic polynomial

## 6.4 Validation of results

In this section, we perform a "validation" of the results, i.e. check whether the measured results match the theoretical expectations.

### 6.4.1 Results for even and odd multiples of $n$

Theoretically, it should not matter how we approach REV. Therefore, it is necessary to verify that for the same values of Young's moduli, the resulting tensor comes out almost identical, regardless of whether we approach REV via even or odd multiples. To confirm the identity of the results, the following table lists the relative residuals between the tensors coming out for even and odd multiples of  $n$  in percentages for all ratios:

Table 6.3: Relative residuals between tensors for even and odd multiples of  $n$

Ratio	Chessboard	Sandwich
2	0,071062457%	1,093277233%
4	0,040265626%	1,340800827%
10	0,07103133%	1,431932456%
50	0,05272631%	1,910778411%

In the table 6.3 the difference of relative residues between the structures is visible. This difference is due to the values of  $n$  for which convergence has occurred. For the checkerboard there is convergence for very similar dimensions, see table 6.1, but for the sandwich the difference in dimensions is much larger, see table 6.2, and the difference increases with higher Young's modulus ratios.

### 6.4.2 Analysis of computed results using the mixture rule

For the sandwich, the obtained elements of the effective tensor correspond to the harmonic mean of the Young's moduli. This is consistent with the empirical expectation, which is the so-called **inverse mixture rule**, which is used in composites to predict various properties. We will not discuss this rule in detail here, for more details see (Roylance, 2000), or (Wikipedia contributors, 2024).

For the chessboard, the values of the effective tensor are in the range  $Y_1$  and  $Y_2$ , but tend to be closer to the value of  $Y_2$ , especially for large fractions of  $Y_1$  and  $Y_2$ . This is probably due to the fact that the individual chessboard arrays with high stiffness are connected only at the corners.

## 6.5 Summary of results and observations

Z vypočtených výsledků, grafů a pozorování jsou patrné určité trendy či závislosti, které jsou sumarizované v následujícím souhrnu:

Summary of findings and observations based on calculated results
Higher ratio of Young's moduli $\rightarrow$ Higher value of the REV dimension
Higher ratio of Young's moduli $\rightarrow$ Higher values of effective coefficients
Odd multiples of $n$ always converge more slowly
For a chessboard, the first and second elements on the main diagonal are almost equal, while the third element is half the size compared to them
For a sandwich, the elements on the main diagonal gradually decrease
The values of the effective elastic coefficients are generally similar for even and odd multiples of $n$ (maximum deviation 1.9%)
The relative residuals of the tensors for even and odd multiples of $n$ for a chessboard are orders of magnitude smaller than for a sandwich
The relative residuals of the tensors for even and odd multiples of $n$ for the sandwich increase with larger Young's modulus ratios
The effective elastic coefficients are higher for the sandwich than for the chessboard
The obtained elements for both the checkerboard and the sandwich lie within the range of values of Young's moduli
For the sandwich, the obtained elements correspond to the harmonic mean
For the checkerboard, the elements obtained tend to be closer to the lower value of Young's modulus
The time for each calculation grows approximately quadratically with $n$

## 7 Conclusion

The aim of this work was to implement an efficient computational tool to obtain the mechanical properties of difficult to describe microheterogeneous materials by numerical homogenization.

First, we theoretically discussed the problem of numerical homogenization, including the basic physical laws, mathematical equations and boundary conditions necessary to understand the problem. Next, we analyzed the different microstructures and types of materials to which this problem applies.

The theoretical part and the analysis of microstructures provided the necessary knowledge for the design and implementation of a computational algorithm addressing this problem. A detailed description of this program, from installation to its use to calculate the required tensors, completes this work.

In the last section we evaluate the results obtained and their interpretation. The results were consistent with empirical expectations, confirming the correctness of the procedures used.

In conclusion, numerical homogenization provides an efficient way to obtain mechanical properties of materials, which is crucial for deeper understanding and further research not only in engineering but also in other fields.

The motivation for this work was to understand the problem of microheterogeneous materials, their characteristics and to obtain their properties in two dimensions. As a natural next step, this work could be extended to three dimensions or possibly analyzing more complex microstructures, for example those containing random cracks.



## Bibliography

- BRDIČKA, Miroslav, Ladislav SAMEK, and Bruno SOPKO, 2005. *Mechanika continua*. Academia.
- CARNEIRO, Vitor Hugo, José MEIRELES, and Hélder PUGA, 2013. Auxetic materials—A review. *Materials Science-Poland*. Vol. 31, pp. 561–571.
- FLOW123D, 2024. *Flow123d*. Available also from: <https://flow123d.github.io/>. [Online; navštíveno 5. 05. 2024].
- MIN, Ki-Bok and Lanru JING, 2003. *Numerical determination of the equivalent elastic compliance tensor for fractured rock masses using the distinct element method*. Vol. 40. No. 6. ISSN 1365-1609. Available from DOI: [https://doi.org/10.1016/S1365-1609\(03\)00038-8](https://doi.org/10.1016/S1365-1609(03)00038-8).
- POVEY, Rhys G, 2023. Voigt transforms.
- ROYLANCE, David, 2000. *INTRODUCTION TO COMPOSITE MATERIALS*. Available also from: [https://ocw.mit.edu/courses/3-91-mechanical-behavior-of-plastics-spring-2007/a4d6750d1f2a3f8834338507b209657b\\_\\_03\\_\\_composites.pdf](https://ocw.mit.edu/courses/3-91-mechanical-behavior-of-plastics-spring-2007/a4d6750d1f2a3f8834338507b209657b__03__composites.pdf). [Online; navštíveno 19. 05. 2024].
- SCHAFER, M., 2006. *Computational Engineering - Introduction to Numerical Methods*. Springer.
- SIDDALL, R., 2024. *Odkaz na GitHub repozitář s kódem*. Available also from: <https://github.com/RonaldSiddall>. [Online; navštíveno 5. 05. 2024].
- WIKIPEDIA CONTRIBUTORS, 2024. *Rule of mixtures — Wikipedia, The Free Encyclopedia*. Available also from: [https://en.wikipedia.org/w/index.php?title=Rule\\_of\\_mixtures&oldid=1224132421](https://en.wikipedia.org/w/index.php?title=Rule_of_mixtures&oldid=1224132421). [Online; navštíveno 19.05.2024].
- ZOHDI, T. I. and P. WRIGGERS, 2005. *An Introduction to Computational Micromechanics*. Springer.