Department of Structural Engineering
Faculty of Engineering Science and Technology
NTNU- Norwegian University of Science and Technology

ACCESSIBILITY

MASTER THESIS 2011

SUBJECT AREA:	DATE:	NO. OF PAGES:
Structural Engineering	09.06.2011	118

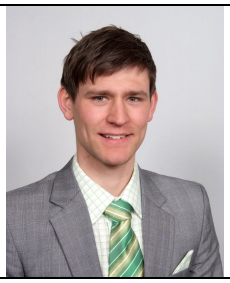
TITLE:

Computational Homogenization with Non-matching Grids treated with Localized Lagrange Multipliers

Numerisk homogenisering med forskjellig diskretisering av de representative volumelementene behandlet med lokaliserte Lagrange multiplikatorer

BY:

Olav Haukvik



SUMMARY:
Solving heterogeneous material problems are of importance in many fields. If the heterogeneities are small compared to the scale of the whole problem, a standard finite element analysis often becomes computationally too large. Multi-scale homogenization is a technique that reduces the amount of calculations, but still manages to capture the heterogeneous properties. The domain of the problem is divided into Representative Volume Elements(RVEs), which in turn are discretized through ordinary finite elements. Periodic boundary conditions have to be applied to the RVEs for homogenization to be possible, and a common way to maintain these boundary conditions is by Multi-Point Constraints(MPC). A limitation with MPC is that it does not maintain the periodic boundary conditions in a correct manner when the RVE boundary nodes are non-matching, which they in general are.

In this thesis, Localized Lagrange Multipliers(LLM) are used to maintain the periodic boundary conditions over the RVE, in order to handle RVEs with non-matching grids. Mathematical homogenization theory in 2D and derivations of MPC and the LLM method are given. A computer program solving two-scale computational homogenization problems in 2D using both LLM and MPC has been implemented in MATLAB. Several RVEs with different boundary situations and material compositions are analysed, and the results from the LLM and MPC analysis are compared.

The results show that LLM is more suitable than MPC to handle the periodic boundary conditions in multi-scale homogenization. LLM deals with all situations that MPC does. In addition, it produces reliable solutions when the boundary nodes are non-matching.

RESPONSIBLE TEACHER: Kjell Magne Mathisen

SUPERVISOR(S): Kjell Magne Mathisen and Trond Kvamsdal

CARRIED OUT AT: Department of Structural Engineering

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY NTNU
FACULTY OF ENGINEERING SCIENCES AND TECHNOLOGY
DEPARTMENT OF STRUCTURAL ENGINEERING

TKT4915 Beregningsmekanikk, masteroppgave

Masteroppgave 2011

for

Olav Haukvik

Computational Homogenization with Non-matching Grids treated with Localized Lagrange Multipliers

Numerisk homogenisering med forskjellig diskretisering av de representative volumelementene behandlet med lokaliserte Lagrange multiplikatorer

To predict the macroscopic behaviour of heterogeneous materials various homogenization techniques are typically used. A general approach to account for the effect of the material substructure in constitutive modelling is to carry out computational homogenization on a representative volume element (RVE). This approach presumes complete scale separation such that the subscale solutions interact only via their homogenized results on the macro scale, typically via equilibrium of macro scale stresses. To obtain representative results, the size of the RVE must be sufficiently larger than the characteristic length of the subscale structure, e.g., the particle spacing in a particle reinforced composite.

When solving the RVE problem macro scale boundary conditions should be applied. Periodic boundary conditions are consistent with the periodicity assumptions inherent in the formulation. However, practical problems very often lead to different discretization of the interfaces of the RVE, such that the interface coupling will not be kinematically consistent.

The purpose of this master thesis is to study and demonstrate how Localized Lagrange Multipliers (LLM) may be used to handle coupling of non-matching grids for 2D elasticity problems. The thesis should also provide a review of numerical homogenization and techniques for coupling of non-matching grids in general. The study should emphasize theory and computational formulation of the LLM method as well as demonstrate how LLM compares to other methods.

The master thesis should be organized as a research report. It is emphasized that clarity and structure together with precise references are central requirements in writing a scientific report.

Advisors: Kjell Magne Mathisen and Trond Kvamsdal

The master thesis should be handed in at the Department of Structural Engineering within June 14, 2011.

NTNU, January 17, 2011
Kjell Magne Mathisen
Principal Advisor

Preface

This Master's thesis is carried out at the Department of Structural Engineering at the Norwegian University of Science and Technology, NTNU. The thesis is the final work in the Civil and Environmental Master Program at NTNU, and has been written over a period of 20 weeks from January to June 2011.

The first three months were mainly spent programming the MATLAB code. A code in MATLAB can be hard to understand when not written yourself, but I have tried to write the code as a reflection of the theory. If some of the theory feels hard to grasp, my advise is to look in the code to see how it is done in practice. I hope that the program is fairly easy to operate also for others than myself.

I would like to express my gratitude to Kjell Magne Mathisen and Trond Kvamsdal, who both have been my advisers. They have been available when I have asked for meetings, where they have given me challenges and answered my questions in a constructive manner. I also want to thank Kari Marie Børset Skjerve from Statoil Research Center at Rotvoll for guidance concerning the theory behind the wave velocity plots.

Trondheim, June 2011 Olav Haukvik

Abstract

Solving heterogeneous material problems are of importance in many fields. If the heterogeneities are small compared to the scale of the whole problem, a standard finite element analysis often becomes computationally too large. Multi-scale homogenization is a technique that reduces the amount of calculations, but still manages to capture the heterogeneous properties. The domain of the problem is divided into Representative Volume Elements(RVEs), which in turn are discretized through ordinary finite elements. Periodic boundary conditions have to be applied to the RVEs for homogenization to be possible, and a common way to maintain these boundary conditions is by Multi-Point Constraints(MPC). A limitation with MPC is that it does not maintain the periodic boundary conditions in a correct manner when the RVE boundary nodes are non-matching, which they in general are.

In this thesis, Localized Lagrange Multipliers(LLM) are used to maintain the periodic boundary conditions over the RVE, in order to handle RVEs with non-matching grids. Mathematical homogenization theory in 2D and derivations of MPC and the LLM method are given. A computer program solving two-scale computational homogenization problems in 2D using both LLM and MPC has been implemented in MATLAB. Several RVEs with different boundary situations and material compositions are analysed, and the results from the LLM and MPC analysis are compared.

The results show that LLM is more suitable than MPC to handle the periodic boundary conditions in multi-scale homogenization. LLM deals with all situations that MPC does. In addition, it produces reliable solutions when the boundary nodes are non-matching.

Contents

Chapter 1

Introduction

Heterogeneous materials are found everywhere. Actually, materials that normally are considered homogeneous, are heterogeneous on a sufficiently small scale, such as steel. The components that create the heterogeneities in a material are often much smaller in size than the size of the whole material component. For instance in reservoir simulation, a field of current interest, the geomechanical model is typically built from units that are at least 100 times larger in volume than the units of the geological model [21]. The seabed consists of relatively thin layers with different rigidity compared to the size of the reservoirs and the depth down to the reservoirs, which yield the difference in scale [19]. Composite materials as reinforced concrete, plastics and wood are also good examples of heterogeneous materials where the computational model should capture the heterogeneous properties.

Consider a problem of various length scales, where the material is heterogeneous on the smallest scale. Solving this problem with a standard finite element analysis requires that the size of the elements correspond with the size of the smallest heterogeneity. For large and complex problems, this leads to a computationally large analysis of a size beyond the capacity of the computing machines of the near future. To handle this problem in a more efficient way, various techniques have been proposed. A promising approach developed in the recent years is Multi-scale Computational Homogenization. This procedure does not yield closed form over all constitutive equations. Instead it computes stress-strain relations at points of interest on the macroscopic field via a detailed modelling of the microscopic area linked to the point [19].

Computational homogenization introduces a coarser element grid to lower the computational effort. These elements consist themselves of several ordinary finite elements and are named Representative Volume Elements(RVEs), implying that they describe the heterogeneities within their range. Two independent problems need then to be solved; one microscopic problem and one macroscopic problem. In first order linear computational homogenization, a material will be considered heterogeneous at micro scale and homogeneous at macro scale. The macroscopic behaviour of the material is

predicted based on the microscopic properties, and it is highly important to preserve the information that is being transferred between the different grids and to make the system satisfy physical principles and conservation laws at all levels [14].

The RVEs are assumed to vary periodically, and the periodicity is maintained in the calculations by boundary conditions in the micro scale problem, also known as the RVE problem. A common way to force periodicity over the RVE is by Multi-Point Constraints(MPC), where the different RVE boundaries are selected to be master or slave boundaries, and the slave boundaries are forced to follow the master boundaries. If the boundary nodes are matching, MPC should handle the coupling just fine, but if the boundary nodes are non-matching, as they generally are, MPC is no longer able to cope with the situation.

There are other possible interface treatments than MPC, and they can be divided into three groups depending on if they introduce Additional Interface Variables(AIVs) to the equation system [6];

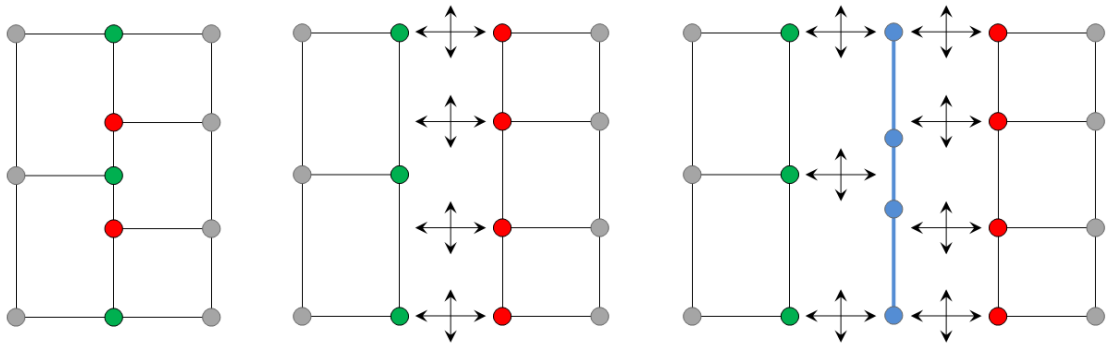
Primal: No AIVs

Dual: AIVs are dual variables, also known as multipliers.

Primal-Dual: AIVs include both dual and primal variables. Primal variables are for instance displacements.

The three interface treatment classes are illustrated in figure 1.1. All treatments have their advantages and criteria they fulfil. In the RVE problem, the most important criteria to satisfy in the coupling are energy conservation, correct stress transmission and kinematics, in that order. The primal treatment does not satisfy energy consistency, and is out of question. An example of a dual treatment is Mortar, whereas Localized Lagrange Multipliers(LLM) is a primal-dual method. Both mortar and LLM satisfy the energy criterion, however, the mortar method generally violates the force patch test, and the LLM method cannot simultaneously pass the force and motion patch tests [6]. Since correct stress transmission is more important than correct displacements, LLM is the most suitable method for multi-scale homogenization, and is the method described and used in this thesis.

In chapter 2, first order linear mathematical homogenization in 2D is described, where only the most important aspects regarding the method have been written down. This thesis focuses on the micro scale problem, and how to calculate the homogeneous properties of an RVE in a best possible way. The accuracy of the result depends on what type of boundary treatment used in the calculation, and the transformation method(MPC) and the LLM method are described in chapter 3. A computer program for solving two-scale linear homogenization problems in 2D, using both MPC and LLM, has been implemented in MATLAB. Chapter 4 explains how the program is built up, and the most important functions regarding the RVE problem are described in detail. The MATLAB code is in chapter 5 applied on different RVEs with both matching and non-matching grids, and the homogenized constitutive matrices yielding from the



(a) *Primal; no AIVs*

(b) *Dual; the AIVs are multipliers*

(c) *Primal-dual; the AIVs are multipliers and displacements*

Figure 1.1: *Three different interface treatment classes. The arrows are indicating multipliers.*

MPC and LLM analysis are compared. The comparison is done by calculating the wave velocities in the homogenized material for different propagation angles, and the wave velocities are graphically depicted in plots.

Most of the theory regarding computational homogenization has been taken from lecture notes by J. Fish [7], and the theory is also described in a master's thesis by K. Spildo [19]. Theory concerning the localized Lagrange multiplier method was mostly found in articles describing interface treatments for fluid-structure interactions written by C. A. Felippa, K. C. Park and M. R. Ross [5, 6, 17, 18], a Ph.D. thesis by M. R. Ross [16] and a pre-master's project by S. B. Raknes [14].

Chapter 2

Linear Homogenization in 2D

To model a heterogeneous material in a finite element analysis, the element grid has to be small enough to capture all the material properties in a satisfying way. The problem that can occur in such an analysis is that the calculations become too heavy for the computational machine. One possibility to minimize the computational effort is to make use of multi-scale mathematical homogenization. This method divides the problem into two independent problems; a micro scale problem and a macro scale problem. The micro scale problem takes care of the heterogeneities in the material, while the macro scale problem is considered as a homogeneous problem and makes use of average material properties that the micro scale problem results in.

2.1 Assumptions

In mathematical homogenization the material is assumed to be heterogeneous on micro scale and homogeneous on macro scale, as illustrated in figure 2.1. To make this separation of scales possible, the macroscopic length scale has to be much larger than

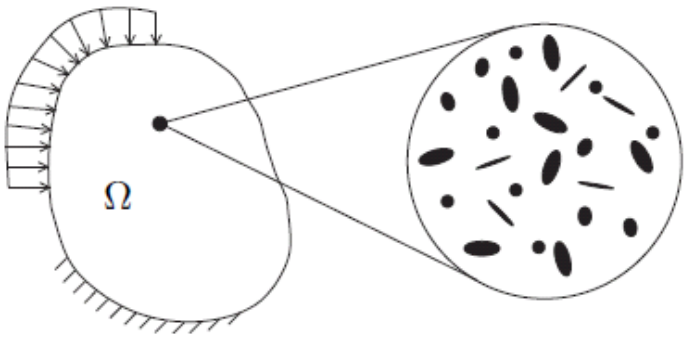


Figure 2.1: *Separation of scales [19].*

the microscopic length scale. The microscopic length scale must in turn be much larger than the molecular dimension to be able to describe the properties of the material in a satisfying manner, i.e. $l_{molecular} \ll l_{micro} \ll l_{macro}$ [11].

To describe the material on micro level, the material is divided into elements that on macro level are considered as points. The physical and geometrical properties around these points are described on micro level through Representative Volume Elements(RVEs). The properties of the RVEs are used as homogenized properties in the macro scale problem and should be of an appropriate size; large enough to represent the micro-structure without introducing non-existing properties. The following definition of an RVE is given in [11]:

> *An RVE is the smallest micro-structural volume that sufficiently accurately represents the overall macroscopic properties of interest.*

Figure 2.2 shows a deformed RVE.

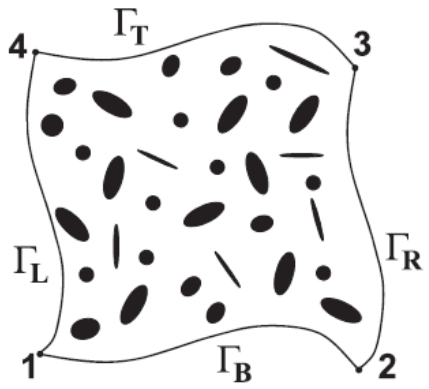


Figure 2.2: *A deformed Representative Volume Element - RVE [8].*

An important assumption that makes it possible to divide the material into RVEs, is that the heterogeneities vary periodically over the macroscopic domain. So-called global periodicity is though a limited assumption since the heterogeneities have to be the same in the whole domain. A more practical assumption is local periodicity, which allows the heterogeneities to vary over the macroscopic domain if it repeats itself in a neighbourhood around each macroscopic point [11], cf. figure 2.3.

The heterogeneous properties on micro scale vary periodically in such a small manner that they are not registered on macro scale, i.e. the macroscopic level is assumed homogeneous. Let x_i be the macro scale coordinate and $y_i = \dfrac{x_i}{\epsilon}$ be the micro scale coordinate, where $0 < \epsilon \ll 1$ denotes the size of a period or an RVE. A periodic

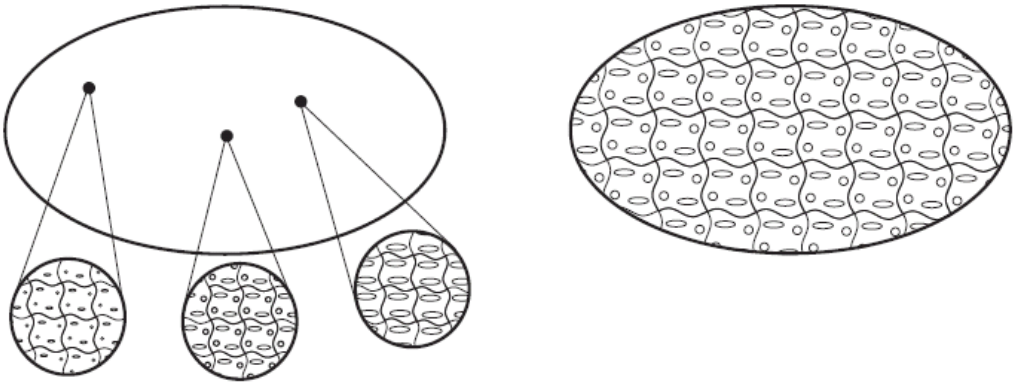


Figure 2.3: *Local periodicity (left) versus global periodicity (right) [11].*

function f can then be approximated as

$$f^{\epsilon}(x_i) = f(x_i, y(x_j)) = f(x_i, y_j + k\Theta_j), \quad k = 1, 2, \ldots$$

where Θ_j is a period of the variation or the length of an RVE, as seen in figure 2.4. An important property of any periodic function, which is exploited in the following mathematical derivation, is that the integral over a period is equal to zero.

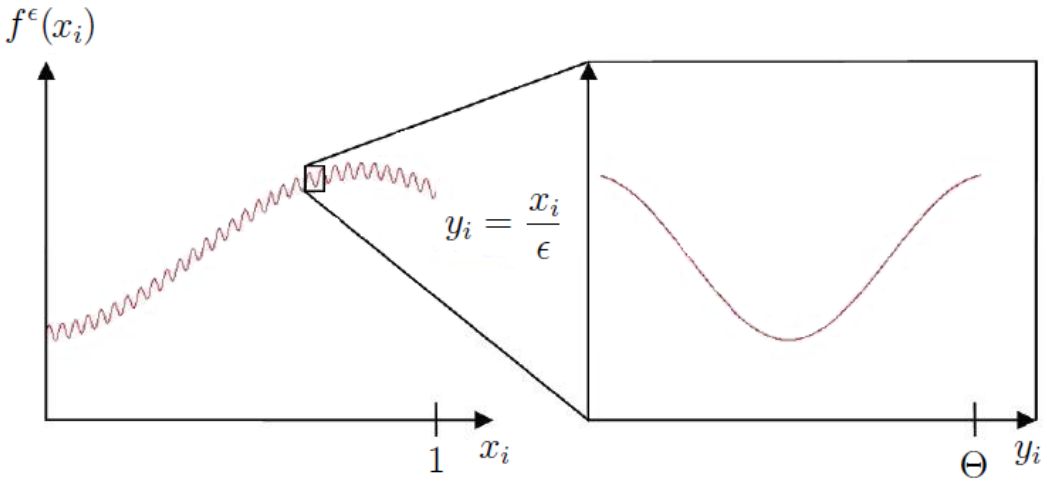


Figure 2.4: *Macro and micro scale coordinates. Modified from [19].*

2.2 First Order Mathematical Homogenization

2.2.1 Equilibrium Equations

Equilibrium of an infinitesimal element of the macro domain, Ω^ϵ, with body forces b_i, results in the equilibrium equation

$$\frac{\partial \sigma_{ij}^\epsilon}{\partial x_j} + b_i = 0 \qquad \text{on} \quad \Omega^\epsilon.$$

The stresses can be expressed by Hooke's law;

$$\sigma_{ij}^\epsilon = L_{ijkl}^\epsilon \varepsilon_{kl}^\epsilon \qquad \text{on} \quad \Omega^\epsilon,$$

where the strains are given as

$$\varepsilon_{kl}^\epsilon = \frac{1}{2}\left(\frac{\partial u_l^\epsilon}{\partial x_k} + \frac{\partial u_k^\epsilon}{\partial x_l}\right) \qquad \text{on} \quad \Omega^\epsilon.$$

On matrix form, Hooke's law looks like

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} L_{1111} & L_{1122} & L_{1112} \\ L_{2211} & L_{2222} & L_{2212} \\ L_{1211} & L_{1222} & L_{1212} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \end{Bmatrix}$$

where $\gamma_{12} = 2\varepsilon_{12}$.

Since the period, or the RVE, is small compared to the size of the whole system, double scale asymptotic expansion is used to approximate the displacement field [2]. The displacements can then be written as

$$u_i(x,y) = \epsilon^0 u_i^{(0)}(x,y) + \epsilon^1 u_i^{(1)}(x,y) + \epsilon^2 u_i^{(2)}(x,y) + \mathcal{O}(\epsilon^3),$$

where ϵ is the scale factor between micro and macro scale. The superscript in brackets is to identify the right part of the displacements to be multiplied with the relevant ϵ. Remember that x and y are the variables from macro and micro scale, respectively, and not the two-dimensional space coordinates.

Using the asymptotic expansion of the displacements, the expression for the strains and Hooke's law result in three equilibrium equations of different order [7];

$$\begin{aligned}
\mathcal{O}(\epsilon^{-2}): \quad & \frac{\partial \sigma_{ij}^{(-1)}}{\partial y_j} = 0, \\
\mathcal{O}(\epsilon^{-1}): \quad & \frac{\partial \sigma_{ij}^{(-1)}}{\partial x_j} + \frac{\partial \sigma_{ij}^{(0)}}{\partial y_j} = 0, \\
\mathcal{O}(\epsilon^{0}): \quad & \frac{\partial \sigma_{ij}^{(0)}}{\partial x_j} + \frac{\partial \sigma_{ij}^{(1)}}{\partial y_j} + b_i = 0.
\end{aligned}$$

2.2.2 $\mathcal{O}(\epsilon^{-2})$ Equilibrium Equation

The order $\mathcal{O}(\epsilon^{-2})$ equilibrium equation can, as shown in [14], be written as

$$\frac{\partial \sigma_{ij}^{(-1)}}{\partial y_j} = \frac{\partial}{\partial y_j}(L_{ijkl}\varepsilon_{kl}^{(-1)}) = \frac{\partial}{\partial y_j}\left[\frac{1}{2}L_{ijkl}\left(\frac{\partial u_l^{(0)}}{\partial y_k} + \frac{\partial u_k^{(0)}}{\partial y_l}\right)\right] = 0.$$

This second order differential equation is transformed to weak form by multiplying by $u_i^{(0)}$ and integrating over the RVE;

$$\int_\Theta u_i^{(0)} \frac{\partial}{\partial y_j}\left[\frac{1}{2}L_{ijkl}(y)\left(\frac{\partial u_l^{(0)}}{\partial y_k} + \frac{\partial u_k^{(0)}}{\partial y_l}\right)\right] \mathrm{d}y = 0.$$

Integration by parts and use of the divergence theorem yield

$$\underbrace{\int_{\partial\Theta} u_i^{(0)}\sigma_{ij}^{(-1)} n_j \,\mathrm{d}\Gamma}_{=0} - \int_\Theta \frac{\partial u_i^{(0)}}{\partial y_j}\left[\frac{1}{2}L_{ijkl}(y)\left(\frac{\partial u_l^{(0)}}{\partial y_k} + \frac{\partial u_k^{(0)}}{\partial y_l}\right)\right]\mathrm{d}y = 0, \tag{2.1}$$

where the first term is equal to zero because of the assumed periodicity of the RVEs. Looking at the first term, it can be seen that it is the displacements multiplied with the tractions integrated over the boundaries of the RVE, in other words it express the difference in energy at the boundaries. Assuming periodicity over the RVE is thus equivalent to require that the difference in energy at the boundaries is equal to zero. Rearranging the second term gives

$$\frac{1}{2}\int_\Theta \frac{\partial u_i^{(0)}}{\partial y_j} L_{ijkl}\left(\frac{\partial u_l^{(0)}}{\partial y_k} + \frac{\partial u_k^{(0)}}{\partial y_l}\right)\mathrm{d}y = 0,$$

where the only non-trivial solution, assuming L_{ijkl} to be positive definite, is

$$\frac{\partial u_i^{(0)}}{\partial y_j} = 0 \quad \Rightarrow u_i^{(0)} = u_i^{(0)}(x).$$

Thus, the leading term in the asymptotic expansion of the displacements is dependent of the macro scale coordinate only. From this it follows that

$$\sigma_{ij}^{(-1)} = \frac{1}{2}L_{ijkl}\left(\frac{\partial u_l^{(0)}}{\partial y_k} + \frac{\partial u_k^{(0)}}{\partial y_l}\right) = 0.$$

2.2.3 $\mathcal{O}(\epsilon^{-1})$ Equilibrium Equation

Using the result from the previous subsection, the $\mathcal{O}(\epsilon^{-1})$ equilibrium equation becomes

$$\underbrace{\frac{\partial \sigma_{ij}^{(-1)}}{\partial x_j}}_{=0} + \frac{\partial \sigma_{ij}^{(0)}}{\partial y_j} = \frac{\partial}{\partial y_j}(L_{ijkl}\varepsilon_{kl}^{(0)}) = 0.$$

Assume the following decomposition of $u_k^{(1)}$;

$$u_k^{(1)}(x,y) = H_{mnk}(y)\varepsilon_{mnx}^{(0)}(x), \quad H_{mnk}(y) \in \rho_\Theta$$
$$\rho_\Theta = \left\{ H_{mnk}(y) | H \in C^0, \Theta - periodic \right\},$$

where

$$\varepsilon_{mnx}^{(0)}(x) = \frac{1}{2}\left(\frac{\partial u_m^{(0)}}{\partial x_n} + \frac{\partial u_n^{(0)}}{\partial x_m} \right).$$

This assumption sets $u_k^{(1)}$ to be a function of the micro scale coordinate y, scaled with the first gradient of $u_k^{(0)}$, hence the name first order homogenization. Inserting into the equilibrium equation gives, after some manipulations [7],

$$\frac{\partial}{\partial y_j}[L_{ijkl}(\Psi_{klmn} + I_{klmn})] = 0,$$

where

$$\Psi_{klmn}(y) = \frac{1}{2}\left(\frac{\partial H_{mnk}(y)}{\partial y_l} + \frac{\partial H_{mnl}(y)}{\partial y_k} \right),$$
$$I_{klmn} = \frac{\delta_{km}\delta_{ln} + \delta_{lm}\delta_{kn}}{2}.$$

This second order differential equation can be solved with the finite element method in order to find $H_{mnk}(y)$, of which $u_k^{(1)}$ can be calculated.

2.2.4 $\mathcal{O}(\epsilon^0)$ Equilibrium Equation

Integrating the order $\mathcal{O}(\epsilon^0)$ equilibrium equation over the RVE and exploiting periodicity yield

$$\int_\Theta \left(\frac{\partial \sigma_{ij}^{(0)}}{\partial x_j} + \frac{\partial \sigma_{ij}^{(1)}}{\partial y_j} + b_i \right) \mathrm{d}\Theta = \int_\Theta \frac{\partial \sigma_{ij}^{(0)}}{\partial x_j}\,\mathrm{d}\Theta + \underbrace{\int_\Theta \frac{\partial \sigma_{ij}^{(1)}}{\partial y_j}\,\mathrm{d}\Theta}_{=0} + \int_\Theta b_i\,\mathrm{d}\Theta = 0.$$

This can be written as

$$\frac{\partial \tilde{\sigma}_{ij}}{\partial x_j} + \tilde{b}_i = 0, \tag{2.2}$$

where

$$\tilde{\sigma}_{ij} = \frac{1}{|\Theta|}\int_{\Theta} \sigma_{ij}^{(0)}\,\mathrm{d}\Theta$$
$$\text{and} \quad \tilde{b}_i = \frac{1}{|\Theta|}\int_{\Theta} b_i\,\mathrm{d}\Theta$$

are the general homogenized stress and body force, respectively.

Substituting

$$\sigma_{ij}^{(0)} = L_{ijkl}\varepsilon_{kl}^{(0)} = L_{ijkl}(\Psi_{klmn} + I_{klmn})\varepsilon_{mnx}^{(0)}$$

into (2.2) gives

$$\frac{\partial}{\partial x_j}\left[\frac{1}{|\Theta|}\int_{\Theta} L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta\,\varepsilon_{mnx}^{(0)}\right] + \tilde{b}_i = 0.$$

This transforms to

$$\frac{\partial}{\partial x_j}\left(\tilde{L}_{ijmn}\varepsilon_{mnx}^{(0)}\right) + \tilde{b}_i = 0,$$

where

$$\tilde{L}_{ijmn} = \frac{1}{|\Theta|}\int_{\Theta} L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta$$

is the homogenized constitutive tensor.

2.2.5 Properties of the Homogenized Constitutive Tensor

Two important properties of a stiffness tensor are that it is symmetric and positive definite. The homogenized constitutive tensor should also possess these properties. Let an expression for the RVE problem be added to the homogenized constitutive tensor [7];

$$\tilde{L}_{stmn} = \frac{1}{|\Theta|}\int_{\Theta} I_{stij}\,L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta$$
$$+$$
$$0 = \frac{1}{|\Theta|}\int_{\Theta} \Psi_{stij}L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta,$$

which becomes

$$\tilde{L}_{stmn} = \frac{1}{|\Theta|}\int_{\Theta} (\Psi_{stij} + I_{stij})L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta.$$

Thus, the homogenized constitutive tensor is symmetric and positive definite.

2.2.6 Chapter Summary

A linear elastic problem can be solved by splitting it into a micro scale problem and a macro scale problem. The micro scale problem is solved by means of the finite element method using periodic boundary conditions, and the material properties are averaged and used to solve the macro scale problem.

Micro Scale Problem - RVE
Find $H_{mnk}(y)$ on the RVE, Θ, such that

$$\begin{aligned}
&\frac{\partial}{\partial y_j}[L_{ijkl}(\Psi_{klmn} + I_{klmn})] = 0 \quad \text{on} \quad \Theta, \\
&H\ _{mnk}(y) = H_{mnk}(y + \Theta) \quad \text{on} \quad \Theta, \\
&H\ _{mnk}(y) = 0 \quad \text{on} \quad \partial\Theta_{\bar{u}},
\end{aligned}$$

where

$$\Psi_{klmn}(y) = \frac{1}{2}\left(\frac{\partial H_{mnk}(y)}{\partial y_l} + \frac{\partial H_{mnl}(y)}{\partial y_k}\right).$$

Macro Scale Problem
Find $u_i^{(0)}(x)$ on the domain Ω, such that

$$\begin{aligned}
&\frac{\partial}{\partial x_j}\left(\tilde{L}_{ijmn}\varepsilon^{(0)}_{mnx}\right) + \tilde{b}_i = 0 \quad \text{on} \quad \Omega, \\
&u_i^{(0)} = \bar{u}_i \quad \text{on} \quad \Gamma_u, \\
&\bar{\sigma}_{ij}\, n_j = \bar{t}_i \quad \text{on} \quad \Gamma_t,
\end{aligned}$$

where

$$\begin{aligned}
\tilde{L}_{ijmn} &= \frac{1}{|\Theta|}\int_\Theta \sigma_{ij}^{mn}\, \mathrm{d}\Theta, \\
\sigma_{ij}^{mn} &= L_{ijkl}(\Psi_{klmn} + I_{klmn}), \\
\Psi_{klmn}(y) &= \frac{1}{2}\left(\frac{\partial H_{mnk}(y)}{\partial y_l} + \frac{\partial H_{mnl}(y)}{\partial y_k}\right),
\end{aligned}$$

are found by solving the micro scale problem.

Chapter 3

Coupling of Non-matching Grids

An essential assumption which makes mathematical homogenization applicable, is to assume periodicity over the RVE. Equation (2.1) shows how this is utilized when the first term is set equal to zero. The periodicity is maintained by applying boundary conditions from macro scale in the micro scale problem. There are three appropriate methods; prescribed displacements, prescribed tractions or prescribed periodicity. It is shown in [11] that boundary conditions of prescribed periodicity give the best results and converge faster than the two other methods.

One way to force prescribed periodicity to the RVE is to require that the difference in displacement between two equal points on opposite boundaries should be constant, i.e. two opposite boundaries should have the same shape, as seen in figure 2.2. This can be taken care of by introducing master DOFs at one boundary and slave DOFs at the opposite boundary, and then force the slave boundary to follow the master boundary. The left grid in figure 3.1 shows an RVE with equally distributed nodes, and the right grid is an example of an RVE with non-matching grids. For matching boundary

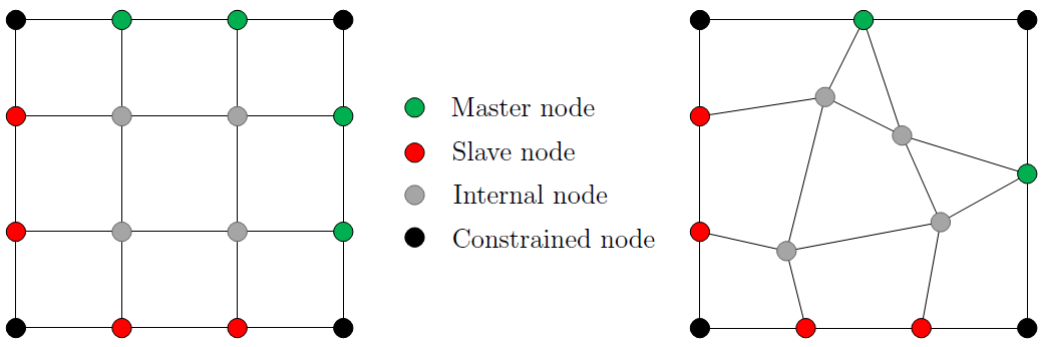


Figure 3.1: *RVE with matching boundary nodes (left) and non-matching boundary nodes (right).*

nodes, the relation between slave nodes and master nodes is straightforward, but for non-matching meshes the slave nodes do not correspond directly with the master nodes. The slave node condensation is done by translation and is described in detail in section 3.1.

Another way to look at periodicity is as explained in the text after equation (2.1); Assuming periodicity over the RVE is equivalent to require that the difference in energy at the boundaries is equal to zero. A method that ensures no loss in energy in a coupling is the localized Lagrange multiplier method, regardless of it is matching or non-matching grids. It is not possible to preserve the kinematics, the stresses and the energy simultaneously in a coupling [6], but at least this method ensures that the energy is conserved and hence a good choice to maintain periodicity. The method is described in section 3.2.

Figure 3.2 shows three RVEs with local periodicity. That is, the RVEs are varying with the same frequency, making it sufficient to calculate just one RVE and use this result for the other RVEs. In the following derivations and explanations in this chapter, the coupling is assumed to be between two adjacent RVEs, when it in practice is a coupling between two opposite boundaries on the same RVE. This is done for the simplicity of the figures and notation.

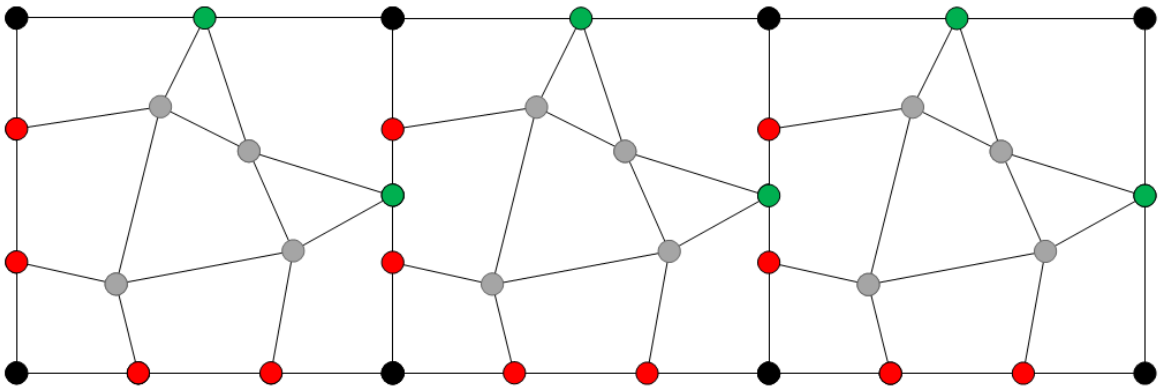


Figure 3.2: *Non-matching RVEs with local periodicity.*

3.1 The Transformation Method

Enforcing constraints by transformation is a common way to handle Multi-Point Constraints(MPC). The slave DOFs are related to the master DOFs through constraint equations and the original global equations, $\mathbf{Kd} = \mathbf{r}$, are modified so that the slave DOFs are eliminated.

3.1.1 Reduced Global Equations

The constraint equations can be written in the form [4]

$$\begin{bmatrix}\mathbf{G}_m & \mathbf{G}_s\end{bmatrix}\begin{Bmatrix}\mathbf{d}_m \\ \mathbf{d}_s\end{Bmatrix} = \mathbf{q},$$

where $\mathbf{d}_m$ and $\mathbf{d}_s$ are the master DOFs and the slave DOFs, respectively. $\mathbf{G}_m$, $\mathbf{G}_s$ and $\mathbf{q}$ contain constants. Since there are as many slave DOFs as there are constraint equations, $\mathbf{G}_s$ is square and non-singular. Solving for the slave DOFs result in

$$\mathbf{d}_s = \mathbf{G}_s^{-1}\big(\mathbf{q} - \mathbf{G}_m\mathbf{d}_m\big),$$

which gives the complete array of DOFs

$$\begin{Bmatrix}\mathbf{d}_m \\ \mathbf{d}_s\end{Bmatrix} = \begin{bmatrix}\mathbf{I} \\ -\mathbf{G}_s^{-1}\mathbf{G}_m\end{bmatrix}\mathbf{d}_m + \begin{Bmatrix}\mathbf{0} \\ \mathbf{G}_s^{-1}\mathbf{q}\end{Bmatrix}$$

or

$$\mathbf{d} = \mathbf{T}\mathbf{d}_m + \mathbf{q}_0, \tag{3.1}$$

where $\mathbf{I}$ is the identity matrix.

Rearranging the global equations according to the partitioning of $\mathbf{d}$, pre-multiplying by $\mathbf{T}^T$ and substituting for $\mathbf{d}$ from equation (3.1) yield

$$\mathbf{T}^T\mathbf{K}\big(\mathbf{T}\mathbf{d}_m + \mathbf{q}_0\big) = \mathbf{T}^T\mathbf{r}$$

or

$$\mathbf{K}_m\mathbf{d}_m = \mathbf{r}_m, \tag{3.2}$$

where

$$\begin{aligned}\mathbf{K}_m &= \mathbf{T}^T\mathbf{K}\mathbf{T}, \\ \mathbf{r}_m &= \mathbf{T}^T\big(\mathbf{r} - \mathbf{K}\mathbf{q}_0\big).\end{aligned}$$

After $\mathbf{d}_m$ is calculated from equation (3.2), the total node displacements can be calculated from equation (3.1).

3.1.2 Constraint Equations

The constraint equations ensure the correct correlation between the master DOFs and the slave DOFs. If the boundary nodes are matching, the slave node displacement equals the master node displacement. In general, the boundary nodes are non-matching, making the relation between master and slave node more complicated. A

common approximation is to make the slave node linearly dependent on the two adjacent master nodes, as illustrated in figure 3.3.

The constraint equations for the first two slave nodes of RVE 2 in figure 3.3 are

$$
\begin{aligned}
d^1_{sx} &= d^1_{mx}, \\
d^1_{sy} &= d^1_{my}, \\
d^2_{sx} &= (1-\alpha_2)d^1_{mx} + \alpha_2 d^2_{mx}, \\
d^2_{sy} &= (1-\alpha_2)d^1_{my} + \alpha_2 d^2_{my}.
\end{aligned}
$$

Written out on matrix form, $\begin{bmatrix}\mathbf{G}_m & \mathbf{G}_s\end{bmatrix}\begin{Bmatrix}\mathbf{d}_m \\ \mathbf{d}_s\end{Bmatrix} = \mathbf{q}$, gives

$$
\left[\begin{array}{ccccc:cccc:c}
1 & 0 & 0 & 0 & \dots & -1 & 0 & 0 & 0 & \dots \\
0 & 1 & 0 & 0 & \dots & 0 & -1 & 0 & 0 & \dots \\
1-\alpha_2 & 0 & \alpha_2 & 0 & \dots & 0 & 0 & -1 & 0 & \dots \\
0 & 1-\alpha_2 & 0 & \alpha_2 & \dots & 0 & 0 & 0 & -1 & \dots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}\right]
\begin{Bmatrix}
d^1_{mx} \\ d^1_{my} \\ d^2_{mx} \\ d^2_{my} \\ \vdots \\ \hline\hline \\ d^1_{sx} \\ d^1_{sy} \\ d^2_{sx} \\ d^2_{sy} \\ \vdots
\end{Bmatrix}
=
\begin{Bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ \vdots
\end{Bmatrix}.
$$

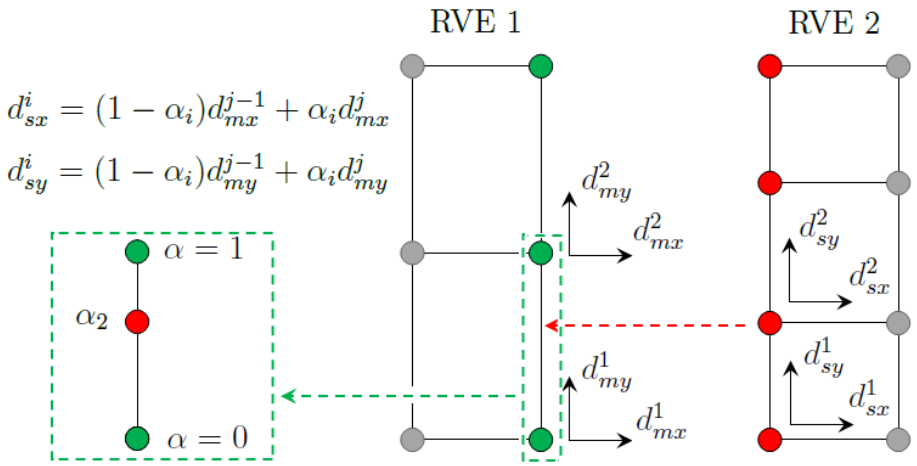


Figure 3.3: *The slave DOF is linearly related to the two adjacent master DOFs.*

3.1.3 Properties of the Transformation Method

In principle, the transformation method maintains the periodicity over the RVE by forcing the slave boundary to follow the master boundary. This is done by condensation, where the slave DOFs are condensed out. In addition, it introduces relations among DOFs through the constraint equations, which makes it different from static condensation. Static condensation uses only relations already contained in the global equations [4].

When the boundary nodes are matching, the transformation method gives the exact solution. The slave nodes follow the master nodes exactly, and it is done without any heavy calculations.

If the boundary nodes are non-matching, the slave boundary will not be correctly calculated. The approximation making the slave nodes linearly dependent on the master nodes does result in the slave nodes being correct, but the displacement field in between differs from the master boundary displacement, as figure 3.4 depicts.

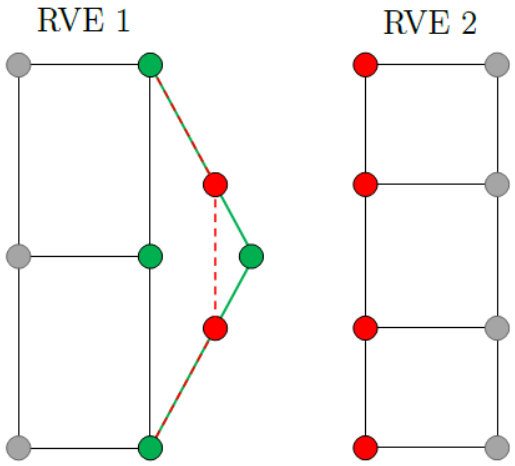


Figure 3.4: *The red dashed line between the slave DOFs shows how the slave displacements differ from the master displacements when using MPC for non-matching grids.*

3.2 The Localized Lagrange Multiplier Method

The Localized Lagrange Multiplier(LLM) method introduces a kinematic interface frame between the two RVEs. The RVEs are then separately connected to the interface frame by interaction forces, also known as Lagrange multiplier fields, as illustrated in figure 3.5. By doing this, each RVE only sees forces passed as Lagrange multipliers and does not need to know details of the opposite mesh, such as element shape functions [5]. The term *localized* means exactly that the multipliers are associated with only one of the RVEs. There is also no need to choose any master or slave nodes.

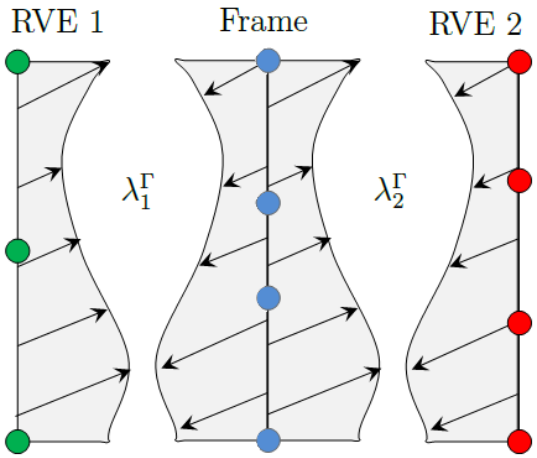


Figure 3.5: *The Lagrange multiplier fields between RVE 1 and the frame and RVE 2 and the frame.*

3.2.1 Equations of Motion in 2D

The equations of motion can be derived from the functional

$$\Pi = \Pi_1 + \Pi_2 + \Pi_b, \tag{3.3}$$

where Π_1 and Π_2 are the functionals of RVE 1 and RVE 2, respectively, and Π_b is the functional of the interface frame. For a structure, which in this case is an RVE, the potential energy is given as

$$\Pi_p = \frac{1}{2}\mathbf{d}^T\mathbf{K}\mathbf{d} - \mathbf{d}^T\mathbf{r},$$

where $\mathbf{d}$ is the nodal displacement vector of the RVE, $\mathbf{K}$ the stiffness matrix and $\mathbf{r}$ the force vector.

The energy of the frame is found by multiplying the displacements with the forces and integrating over the frame surface. Let $\mathbf{d}_1^\Gamma$ and $\mathbf{d}_2^\Gamma$ be the displacements at the boundary facing the frame of RVE 1 and RVE 2, respectively, and $\mathbf{d}_b^\Gamma$ the displacements of the interface frame. Further, let $\boldsymbol{\lambda}_1^\Gamma$ and $\boldsymbol{\lambda}_2^\Gamma$ be the interaction forces between RVE 1 and the frame and RVE 2 and the frame, respectively, as shown in figure 3.5. Π_b can then be expressed as

$$\Pi_b(\boldsymbol{\lambda}_1^\Gamma, \boldsymbol{\lambda}_2^\Gamma, \mathbf{d}_b^\Gamma) = \int_{\Gamma_b} \Big[(\boldsymbol{\lambda}_1^\Gamma)^T(\mathbf{d}_1^\Gamma - \mathbf{d}_b^\Gamma) + (\boldsymbol{\lambda}_2^\Gamma)^T(\mathbf{d}_2^\Gamma - \mathbf{d}_b^\Gamma)\Big]\mathrm{d}\Gamma, \tag{3.4}$$

where Γ_b denotes the interface surface [5]. Π_b becomes equal to zero if $\mathbf{d}_1^\Gamma = \mathbf{d}_b^\Gamma$ and $\mathbf{d}_2^\Gamma = \mathbf{d}_b^\Gamma$, i.e. the two RVEs are coupled together in a weak sense. The functional in equation (3.3) will then become the potential energy of the two RVEs only, as desirable.

In 2D, the displacements and forces may be discretized as

$$\mathbf{d}_1^\Gamma = \begin{Bmatrix} d_{1x}^\Gamma \\ d_{1y}^\Gamma \end{Bmatrix} = \begin{bmatrix} N_{1b}^1 & 0 & \dots & N_{1b}^{n_1} & 0 \\ 0 & N_{1b}^1 & \dots & 0 & N_{1b}^{n_1} \end{bmatrix} \begin{Bmatrix} d_{1x}^1 \\ d_{1y}^1 \\ \vdots \\ d_{1x}^{n_1} \\ d_{1y}^{n_1} \end{Bmatrix} = \mathbf{N}_{1b}\mathbf{d}_1,$$

$$\mathbf{d}_2^\Gamma = \begin{Bmatrix} d_{2x}^\Gamma \\ d_{2y}^\Gamma \end{Bmatrix} = \begin{bmatrix} N_{2b}^1 & 0 & \dots & N_{2b}^{n_2} & 0 \\ 0 & N_{2b}^1 & \dots & 0 & N_{2b}^{n_2} \end{bmatrix} \begin{Bmatrix} d_{2x}^1 \\ d_{2y}^1 \\ \vdots \\ d_{2x}^{n_2} \\ d_{2y}^{n_2} \end{Bmatrix} = \mathbf{N}_{2b}\mathbf{d}_2,$$

$$\boldsymbol{\lambda}_1^\Gamma = \begin{Bmatrix} \lambda_{1x} \\ \lambda_{1y} \end{Bmatrix} = \begin{bmatrix} N_{1\lambda}^1 & 0 & \dots & N_{1\lambda}^{n_{b1}} & 0 \\ 0 & N_{1\lambda}^1 & \dots & 0 & N_{1\lambda}^{n_{b1}} \end{bmatrix} \begin{Bmatrix} \lambda_{1x}^1 \\ \lambda_{1y}^1 \\ \vdots \\ \lambda_{1x}^{n_{b1}} \\ \lambda_{1y}^{n_{b1}} \end{Bmatrix} = \mathbf{N}_{1\lambda}\boldsymbol{\lambda}_1,$$

$$\boldsymbol{\lambda}_2^\Gamma = \begin{Bmatrix} \lambda_{2x} \\ \lambda_{2y} \end{Bmatrix} = \begin{bmatrix} N_{2\lambda}^1 & 0 & \dots & N_{2\lambda}^{n_{b2}} & 0 \\ 0 & N_{2\lambda}^1 & \dots & 0 & N_{2\lambda}^{n_{b2}} \end{bmatrix} \begin{Bmatrix} \lambda_{2x}^1 \\ \lambda_{2y}^1 \\ \vdots \\ \lambda_{2x}^{n_{b2}} \\ \lambda_{2y}^{n_{b2}} \end{Bmatrix} = \mathbf{N}_{2\lambda}\boldsymbol{\lambda}_2,$$

$$\mathbf{d}_b^\Gamma = \begin{Bmatrix} d_{bx} \\ d_{by} \end{Bmatrix} = \begin{bmatrix} N_b^1 & 0 & \dots & N_b^{n_b} & 0 \\ 0 & N_b^1 & \dots & 0 & N_b^{n_b} \end{bmatrix} \begin{Bmatrix} d_{bx}^1 \\ d_{by}^1 \\ \vdots \\ d_{bx}^{n_b} \\ d_{by}^{n_b} \end{Bmatrix} = \mathbf{N}_b\mathbf{d}_b,$$

where

n_1 = number of RVE 1 nodes,
n_2 = number of RVE 2 nodes,
n_{b1} = number of RVE 1 interface boundary nodes,
n_{b2} = number of RVE 2 interface boundary nodes,
n_b = number of interface frame nodes.

$\mathbf{N}_{1b}$ and $\mathbf{N}_{2b}$ contain the shape functions for the displacements of the interface boundary nodes of RVE 1 and RVE 2, respectively. $\mathbf{N}_b$ contains the shape functions for the displacements of the interface frame, whereas $\mathbf{N}_{1\lambda}$ and $\mathbf{N}_{2\lambda}$ contain the shape functions for the interaction forces on RVE 1 and RVE 2, respectively. Appropriate choices of

shape functions are given in section 3.2.2. $\mathbf{d}_1$ and $\mathbf{d}_2$ are the nodal displacements of RVE 1 and RVE 2, respectively, whereas $\mathbf{d}_b$ are the nodal displacements of the interface frame. $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are the nodal interaction forces between the nodes on RVE 1 and the frame and the nodes on RVE 2 and the frame, respectively, as shown in figure 3.6.

Inserting this discretization into equation (3.4) gives

$$\begin{aligned}\Pi_b(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \mathbf{d}_b) &= \int_{\Gamma_b} \Big[(\mathbf{N}_{1\lambda}\boldsymbol{\lambda}_1)^T(\mathbf{N}_{1b}\mathbf{d}_1 - \mathbf{N}_b\mathbf{d}_b) + (\mathbf{N}_{2\lambda}\boldsymbol{\lambda}_2)^T(\mathbf{N}_{2b}\mathbf{d}_2 - \mathbf{N}_b\mathbf{d}_b)\Big]\mathrm{d}\Gamma \\ &= \boldsymbol{\lambda}_1^T(\mathbf{B}_1\mathbf{d}_1 - \mathbf{L}_1\mathbf{d}_b) + \boldsymbol{\lambda}_2^T(\mathbf{B}_2\mathbf{d}_2 - \mathbf{L}_2\mathbf{d}_b),\end{aligned}$$

where

$$\begin{aligned}\mathbf{B}_1 &= \int_{\Gamma_b} \mathbf{N}_{1\lambda}^T\mathbf{N}_{1b}\,\mathrm{d}\Gamma, \quad \mathbf{B}_2 = \int_{\Gamma_b} \mathbf{N}_{2\lambda}^T\mathbf{N}_{2b}\,\mathrm{d}\Gamma, \\ \mathbf{L}_1 &= \int_{\Gamma_b} \mathbf{N}_{1\lambda}^T\mathbf{N}_{b}\,\mathrm{d}\Gamma, \quad \mathbf{L}_2 = \int_{\Gamma_b} \mathbf{N}_{2\lambda}^T\mathbf{N}_{b}\,\mathrm{d}\Gamma.\end{aligned}$$

Now, if the different potential energies are inserted into (3.3), the expression for the functional becomes

$$\begin{aligned}\Pi = &\frac{1}{2}\mathbf{d}_1^T\mathbf{K}_1\mathbf{d}_1 - \mathbf{d}_1^T\mathbf{r}_1 \\ &+ \frac{1}{2}\mathbf{d}_2^T\mathbf{K}_2\mathbf{d}_2 - \mathbf{d}_2^T\mathbf{r}_2 \\ &+ \boldsymbol{\lambda}_1^T(\mathbf{B}_1\mathbf{d}_1 - \mathbf{L}_1\mathbf{d}_b) + \boldsymbol{\lambda}_2^T(\mathbf{B}_2\mathbf{d}_2 - \mathbf{L}_2\mathbf{d}_b).\end{aligned}$$

By differentiating this functional with respect to all of the unknown variables and setting each equation equal to zero, five sets of equations emerge. These equations

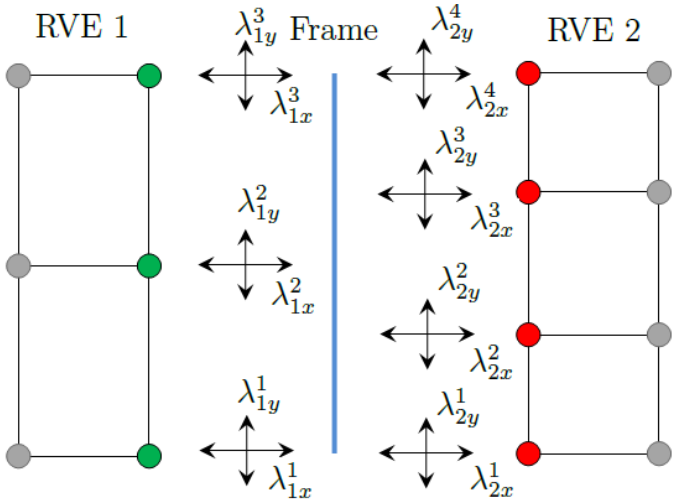


Figure 3.6: *The Lagrange multipliers are located on the RVE nodes.*

have to be satisfied in order to make the functional stationary.

$$\begin{aligned}
\frac{\delta\Pi}{\delta\mathbf{d}_1} = \mathbf{0} \quad &\Rightarrow \quad \mathbf{K}_1\mathbf{d}_1 + \mathbf{B}_1^T\boldsymbol{\lambda}_1 = \mathbf{r}_1 \\
\frac{\delta\Pi}{\delta\mathbf{d}_2} = \mathbf{0} \quad &\Rightarrow \quad \mathbf{K}_2\mathbf{d}_2 + \mathbf{B}_2^T\boldsymbol{\lambda}_2 = \mathbf{r}_2 \\
\frac{\delta\Pi}{\delta\boldsymbol{\lambda}_1} = \mathbf{0} \quad &\Rightarrow \quad \mathbf{B}_1\mathbf{d}_1 - \mathbf{L}_1\mathbf{d}_b = 0 \\
\frac{\delta\Pi}{\delta\boldsymbol{\lambda}_2} = \mathbf{0} \quad &\Rightarrow \quad \mathbf{B}_2\mathbf{d}_2 - \mathbf{L}_2\mathbf{d}_b = 0 \\
\frac{\delta\Pi}{\delta\mathbf{d}_b} = \mathbf{0} \quad &\Rightarrow \quad -\mathbf{L}_1^T\boldsymbol{\lambda}_1 - \mathbf{L}_2^T\boldsymbol{\lambda}_2 = 0
\end{aligned}$$

As seen from equation one and two above, $\mathbf{B}_1$ and $\mathbf{B}_2$ are matrices that map $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ onto the full set of node forces, and should not be confused with the RVE's strain-displacement matrix. Equation five shows that the interaction forces obey Newton's third law. $\mathbf{L}_1$ and $\mathbf{L}_2$ are matrices that relate the frame DOFs to the RVE boundary DOFs. It can hence be seen in equation three and four that the RVE displacements, $\mathbf{d}_1$ and $\mathbf{d}_2$, are equated to the interface frame displacements, $\mathbf{d}_b$ [12];

$$\begin{aligned}
\mathbf{d}_{1b} &= \mathbf{B}_1\mathbf{d}_1 = \mathbf{L}_1\mathbf{d}_b, \\
\mathbf{d}_{2b} &= \mathbf{B}_2\mathbf{d}_2 = \mathbf{L}_2\mathbf{d}_b,
\end{aligned}$$

where $\mathbf{d}_{1b}$ and $\mathbf{d}_{2b}$ are the displacements of the interface boundary nodes of RVE 1 and RVE 2, respectively, as shown in figure 3.7. On matrix form, the coupled equations of

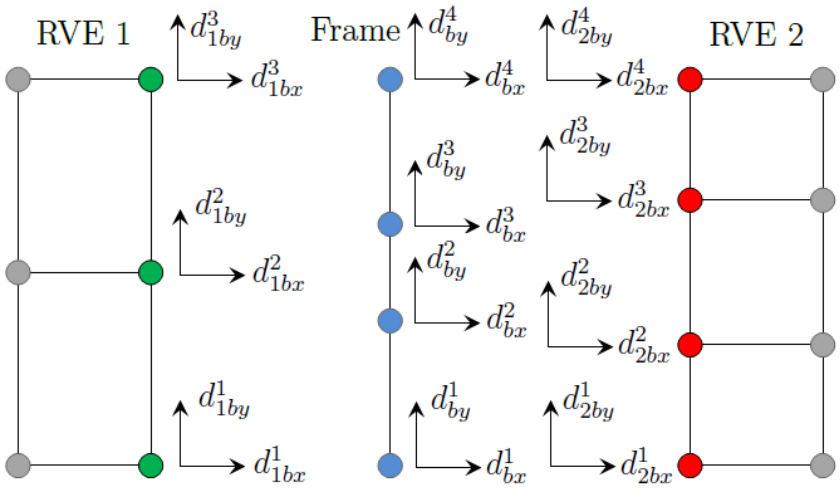


Figure 3.7: *The displacements of the interface boundary nodes of RVE 1 and RVE 2, and the nodal displacements of the interface frame.*

motion become

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} & \mathbf{B}_1^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 & \mathbf{0} & \mathbf{B}_2^T & \mathbf{0} \\ \mathbf{B}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_1 \\ \mathbf{0} & \mathbf{B}_2 & \mathbf{0} & \mathbf{0} & -\mathbf{L}_2 \\ \mathbf{0} & \mathbf{0} & -\mathbf{L}_1^T & -\mathbf{L}_2^T & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \mathbf{d}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}.$$

3.2.2 Choice of Shape Functions

The shape functions for the interface boundary displacements of the RVEs, $\mathbf{N}_{1b}$ and $\mathbf{N}_{2b}$, contain the existing shape functions that correspond with the interface boundary nodes, while the others are set to zero. It may thus be given as

$$N_{kb}^i = \begin{cases} N_k^i & \text{if } i \text{ is an interface boundary node,} \\ 0 & \text{otherwise,} \end{cases}$$

where

$$N_k^i = \text{the original shape function at node } i \text{ from RVE } k,$$
$$i = 1, 2, \ldots, n_k \quad \text{and} \quad k = 1, 2.$$

For a two-dimensional four-node quadrilateral element the shape functions will be linear.

If the RVEs consist of four-node quadrilateral elements, the interface frame will be divided into one-dimensional two-node elements. The shape functions for the interface frame, $\mathbf{N}_b$, are then piecewise linear C^0-continuous functions [5]. The locations of the frame nodes are discussed in section 3.2.3.

$\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{L}_1$ and $\mathbf{L}_2$ have either $\mathbf{N}_{1\lambda}$ or $\mathbf{N}_{2\lambda}$ in the integrand. To simplify the implementation of these integrals, the Lagrange multiplier shape functions, $\mathbf{N}_{1\lambda}$ and $\mathbf{N}_{2\lambda}$, are chosen to be Dirac delta functions located on the interface boundary nodes of the RVEs [5];

$$N_{k\lambda}^j = \delta(\xi - \xi_k^j) = \begin{cases} 1 & \text{if } \xi = \xi_k^j, \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\xi_k^j = \text{the coordinate of node } j \text{ from the interface boundary of RVE } k. \text{ See figure 3.8,}$$
$$j = 1, 2, \ldots, n_{bk} \quad \text{and} \quad k = 1, 2.$$

By implementing this discretization, the Lagrange multipliers become point forces that correspond with the RVE interface boundary DOFs.

Figure 3.8 shows an example of how to calculate the components in $\mathbf{L}_1$ that correspond with the middle node of RVE 1, with $N_{1\lambda}^2$ chosen to be a Dirac delta function. It shows how the Dirac delta function simplifies the integration, however, this does come with a sacrifice in the order of the discretization error [16].

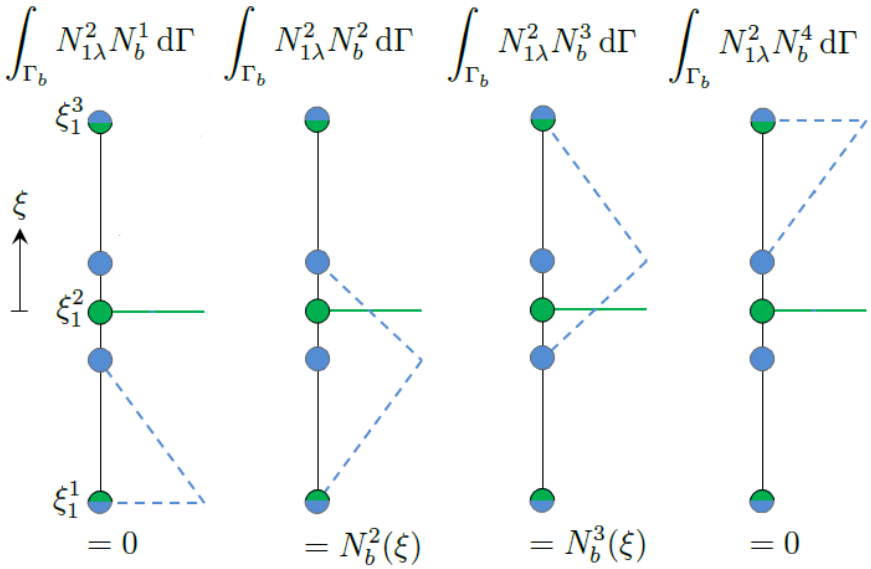


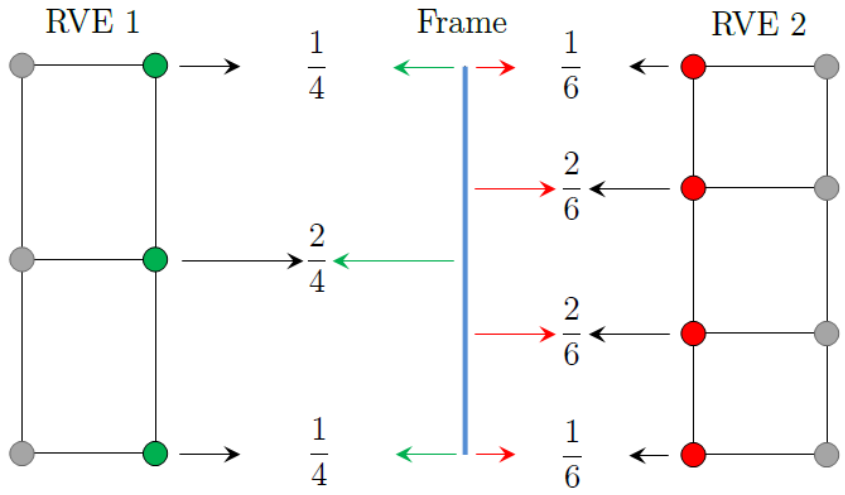
Figure 3.8: *Integration with $N^2_{1\lambda}$ as Dirac delta function and $\boldsymbol{N}_b$ as piecewise linear functions.*

3.2.3 The Zero Moment Rule

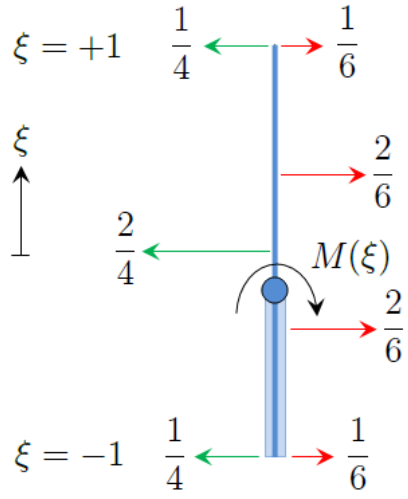
For non-matching meshes treated by the LLM method, the placement of frame nodes should obey conservation conditions that guarantee correct transmission of constant stress states across the interface. For a one-dimensional frame which separates two two-dimensional RVEs, the problem can be solved directly using the Zero Moment Rule (ZMR) [18].

First, points on the frame are mapped to the isoparametric dimensionless coordinate ξ that ranges from $\xi = -1$ to $\xi = +1$. A free body diagram of the frame is drawn, where the forces consist of the Lagrange multipliers that are normal to the frame and have a magnitude that create a constant stress state, as illustrated in figure 3.9(a). The frame is now considered an isolated object and the moment distribution, $M(\xi)$, can be calculated, as shown in figure 3.9(b). The points where the moment is zero are now possible frame node locations assuring that constant stress states are preserved [13].

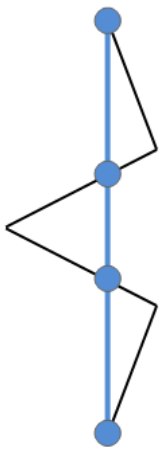
The ZMR does not say how many frame nodes to use, however they should be symmetrically distributed and they should not be collocated at the refined mesh nodes [17]. As seen in figure 3.9(c), the moment in this example is zero in four places and gives rise to three possible frame node configurations shown in figure 3.9(d). The most suitable configuration of frame nodes is problem dependent. A proof of the zero moment rule can be found in [13].



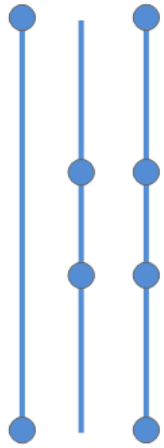
(a) *The Lagrange multipliers normal to the frame are mapped to the interface frame, with a magnitude assuring that they create a constant stress state.*



(b) *The moment is calculated from the forces along the shaded portion.*



(c) *Bending moment diagram; The frame nodes are located at the roots of* $M(\xi)$.



(d) *Possible frame node configurations.*

Figure 3.9: *The zero moment rule: An example.*

3.2.4 Properties of the LLM Method

As mentioned in the introduction, it seems impossible to simultaneously satisfy the kinematics, the stresses and the energy consistency when coupling non-matching meshes with one common frame configuration [6]. Which criteria that are the most important to satisfy, depends on the given problem. In homogenization theory it is most important to satisfy the energy consistency, since this is equivalent with periodic boundary conditions.

The LLM method conserves the energy in the coupling when the first variation of the functional is set equal to zero, resulting in five equations that have to be satisfied. The stresses are preserved if they are constant and the frame nodes are placed according to the zero moment rule. The kinematics is satisfied in a weak sense only.

The advantages using LLM are that the RVEs are completely uncoupled and there is no need to choose any master or slave boundaries. In addition, the Lagrange multipliers are physical quantities, as they are the forces required to hold the RVEs together.

The downsides with the LLM method are that it introduces a large number of additional interface variables and that the frame must be discretized trough special rules if certain conservation attributes should be enforced [5]. Also, the diagonal in the equation of motion-matrix contains null submatrices and the positive definiteness is thus lost. However, the equation system is non-singular if the $\mathbf{K}$s are positive definite [4].

Chapter 4

Computational Homogenization in MATLAB

The implementation of a two-scale linear homogenization problem may be divided into four steps:

1. The micro scale problem, also known as the RVE problem, must be solved for as many right-hand side vectors as unit strain components in the problem. The stresses each right-hand side vector creates are then calculated.
2. Calculate the homogenized constitutive matrix $\tilde{\mathbf{L}}$.
3. Solve the macro scale problem with $\tilde{\mathbf{L}}$ as material properties.
4. Post-process stresses in the RVEs of current interest.

The focus in this thesis is on the first two steps.

4.1 The RVE problem

The RVE problem is solved by means of the finite element method, and it has to be solved for as many right-hand side vectors as unit strain components in the problem. In two dimensions this is done by applying three different unit strain states. Each strain state results in a nodal displacement field, of which stresses in each element in the RVE can be calculated. The stresses calculated are then a column in the constitutive matrix $\mathbf{L}$ of the element. The unit strain vectors that are applied on each element are

$$\boldsymbol{\varepsilon}_{0,11} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}, \qquad \boldsymbol{\varepsilon}_{0,22} = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}, \qquad \boldsymbol{\varepsilon}_{0,12} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix},$$

and the resulting L-matrix builds up column by column

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} L_{1111} & L_{1122} & L_{1112} \\ L_{2211} & L_{2222} & L_{2212} \\ L_{1211} & L_{1222} & L_{1212} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{Bmatrix}.$$

The RVEs are assumed to vary periodically, making it sufficient to evaluate just one RVE using periodic boundary conditions. The boundaries which are going to be coupled are the opposite boundaries on the same RVE. Looking at figure 4.1, Γ_1 is going to be coupled with Γ_2, and Γ_3 is going to be coupled with Γ_4. The corners are constrained for displacements, because each corner node are related to four different RVEs.

Using the localized Lagrange multiplier method to handle the boundary conditions means solving the equation system

$$\underbrace{\begin{bmatrix} \mathbf{K} & \mathbf{B}_1^T & \mathbf{B}_2^T & \mathbf{B}_3^T & \mathbf{B}_4^T & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_1 & \mathbf{0} \\ \mathbf{B}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_2 & \mathbf{0} \\ \mathbf{B}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_3 \\ \mathbf{B}_4 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_4 \\ \mathbf{0} & -\mathbf{L}_1^T & -\mathbf{L}_2^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{L}_3^T & -\mathbf{L}_4^T & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{Bmatrix} \mathbf{d} \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_3 \\ \boldsymbol{\lambda}_4 \\ \mathbf{d}_{b12} \\ \mathbf{d}_{b34} \end{Bmatrix}}_{\mathbf{x}} = \underbrace{\begin{Bmatrix} \mathbf{r} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}}_{\mathbf{b}}.$$

In the system $\mathbf{Ax} = \mathbf{b}$, it is only the right-hand side $\mathbf{b}$ that changes for each unit strain state. The A-matrix is constant for the given RVE, except for different frame node configurations where the L-matrices are changed. When the x-matrix is solved for, the RVE node displacements are found in the vector $\mathbf{d}$, of which the stresses can be calculated.

If MPC is used to handle the periodic boundary conditions, the equation system to be solved is

$$\mathbf{T}^T\mathbf{K}\mathbf{T}\mathbf{d}_m = \mathbf{T}^T\mathbf{r},$$

where $\mathbf{q}$ is set equal to zero and $\mathbf{T} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_s^{-1}\mathbf{G}_m \end{bmatrix}$.

After solving for the master displacements $\mathbf{d}_m$, the total displacement vector can be found by

$$\mathbf{d} = \mathbf{T}\mathbf{d}_m.$$

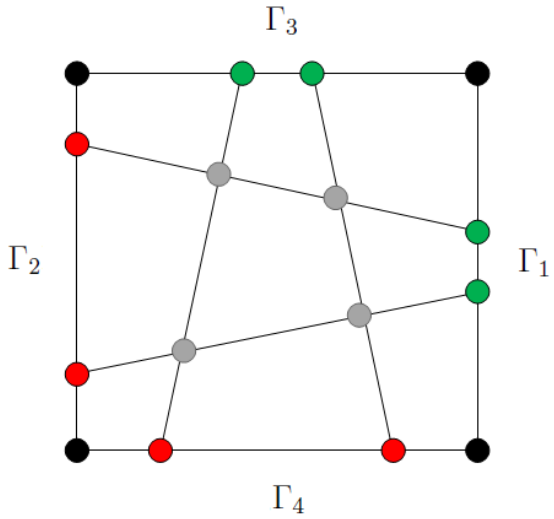


Figure 4.1: *RVE defining the four boundaries.*

4.2 The Homogenized Constitutive Matrix $\tilde{\mathbf{L}}$

The stresses found in the RVE problem is averaged over the entire RVE to get the homogenized material properties in $\tilde{\mathbf{L}}$;

$$\tilde{L}_{ijmn} = \frac{1}{|\Theta|}\int_{\Theta} L_{ijkl}(\Psi_{klmn} + I_{klmn})\,\mathrm{d}\Theta = \frac{1}{|\Theta|}\int_{\Theta} \sigma_{ij}^{mn}\,\mathrm{d}\Theta, \tag{4.1}$$

where σ_{ij}^{mn} are stress functions induced by the overall unit strain ε_{mn}, and Θ is the RVE volume [20]. The $\tilde{L}$-matrix is now used as the element constitutive matrix in the macro problem.

4.3 Implementation of LLM in MATLAB

A program for solving two-scale linear homogenization problems in 2D with localized Lagrange multipliers has been implemented in MATLAB. It is based on the RVE having four-node quadrilateral elements and that opposite RVE-boundaries consist of equal number of nodes, as illustrated in figure 4.1. Numerical integration in the program is carried out with 2×2 points Gaussian quadrature. The main script which runs the program is given in figure 4.2. This is where the geometry and properties of the RVE and macro problem are punched in.

The pre-process in the RVE problem includes calculating the element constitutive matrix $\mathbf{D}$, the stiffness matrix $\mathbf{K}$ and for each boundary *gamma*; $\mathbf{L}_\Gamma$, $\mathbf{B}_\Gamma$ and the frame coordinates. $\mathbf{K}$, $\mathbf{L}_\Gamma$ and $\mathbf{B}_\Gamma$ are then organized in $\mathbf{A}$.

Two-scale Linear Homogenization with LLM

Input Micro Problem
- Perimeter coordinates
- Young's Modulus and Poisson's Ratio for each element

Input Macro Problem
- Perimeter coordinates
- Boundary conditions - **BCmacro**
- Tractions - **rmacro**

RVE Analysis
- Calculate the constitutive matrix **D** for all elements
- Call K_matrix.m to get **K**

gamma = 1
- Call zmr.m to get the natural coordinates of the frame for boundary 1 and 2
- Call L_matrix.m to get **L1**
- Call B_matrix.m to get **B1**

gamma = 2
- Call L_matrix.m to get **L2**
- Call B_matrix.m to get **B2**

gamma = 3
- Call zmr.m to get the natural coordinates of the frame for boundary 3 and 4
- Call L_matrix.m to get **L3**
- Call B_matrix.m to get **B3**

gamma = 4
- Call L_matrix.m to get **L4**
- Call B_matrix.m to get **B4**

- Form the coupled equations of motion-matrix **A**

HomoL = zeros(3 , 3)
FOR m = 1 **TO** m = 3 **DO**
    eps = zeros(3 , 1)
    eps(m , 1) = 1
    - Call r_matrix.m to get **r**
    b = zeros(length(A) , 1)
    b(1:length(K)) = r
    x = A\b
    d = x(1:length(K))
    - Call sigmaHomo_vector.m to get **sigmaHomo**
    HomoL(: , m) = sigmaHomo
END DO

Macro Analysis

- Call macro_K_matrix.m to get **Kmacro**

u = Kmacro\rmacro
dmacro(BCmacro) = u

Figure 4.2: *Pseudocode for the main script using LLM.*

To find the homogenized constitutive matrix **HomoL**, the LLM equation system is solved three times, each time for a different unit strain **eps**. Then the macro problem is solved with **HomoL** as input in *macro_K_matrix.m*.

In the following subsections the functions *zmr.m, L_matrix.m, B_matrix.m* and *sigmaHomo_vector.m* are described in detail. *K_matrix.m, macro_K_matrix.m, r_matrix.m* and *A_matrix.m* are not mentioned further, since the latter function is just arranging matrices and the first three are not any different as for a normal finite element code. The code with all functions can be found in appendix B.

4.3.1 The Zero Moment Rule - zmr.m

The intention with the zero moment rule is to find the placement of the frame nodes. The frame nodes are treated in natural coordinates and range from $\xi = -1$ to $\xi = +1$ and are located where the moment created by the Lagrange multipliers is equal to zero, cf. figure 3.9. The first node, $\xi = -1$, is the bottom node for the frame between Γ_1 and Γ_2 and the left node for the frame between Γ_3 and Γ_4.

The forces, and the nodes where the forces work, have to be found before calculating the moment. The nodes where the forces work are the boundary nodes of the RVE, and they are named **lambdaNatcoord1** and **lambdaNatcoord2** as seen in the pseudocode given in figure 4.3. The magnitude of the forces is found calling the function *zmr_force.m*, where the stress is assumed to be constant along the RVE boundary with a value of 1. Each boundary element is treated with linear shape functions, and since the nodes are always at the element ends, each node has to take half the force. That is, the force is equal to the shape function area, as shown in figure 4.4(a).

By internal boundary nodes or internal multiplier nodes in the pseudocode, it is meant the RVE boundary nodes from both boundaries together except the end nodes, and they are arranged from low to high in array **b**. The moment will never change sign between the end node and the first internal multiplier, so the first moments calculated are at the second and third internal multiplier node. If the moments have different signs, the moment is equal to zero somewhere between those nodes, and the coordinate can be calculated as in figure 4.4(b). There should also be frame nodes where the boundary nodes are matching. The resulting frame nodes are given in the array **frameNatcoord**.

It can be convenient to change the frame node configuration in some occasions. The *zmr.m* has four built-in options of configurations which is chosen in the main script; All nodes, end nodes, internal nodes or every other node.

The Zero Moment Rule

IF gamma = 1 **THEN**
    id1 = node numbers for boundary 1
    id2 = node numbers for boundary 2
    xy1 = y-coordinates for boundary 1 nodes
    xy2 = y-coordinates for boundary 2 nodes
ELSE
    id1 = node numbers for boundary 3
    id2 = node numbers for boundary 4
    xy1 = x-coordinates for boundary 3 nodes
    xy2 = x-coordinates for boundary 4 nodes
END IF

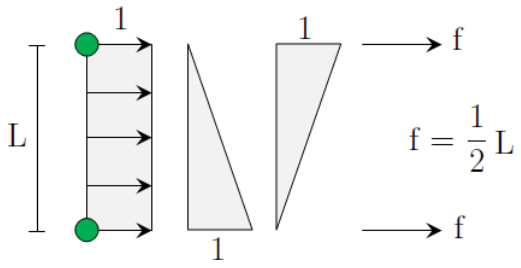
- lambdaNatcoord1 = Natural coordinates of boundary 1/3 nodes
- lambdaNatcoord2 = Natural coordinates of boundary 2/4 nodes

IF just two boundary nodes at both boundaries **THEN**
    frameNatcoord = $[-1, 1]$
    RETURN
END IF

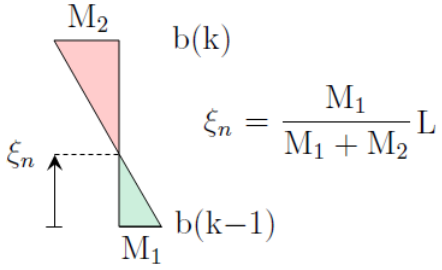
- Call zmr_force.m to get the Lagrange multipliers for both boundaries so that they create a constant stress state
- Arrange the internal boundary nodes from both boundaries in a single array **b** and the corresponding multipliers in a single array **force**

n = 2
FOR k = 2 **TO** k = length(b) **DO**
    - Calculate the moment at b(k−1) and b(k)
    IF moments have different signs **OR** b(k−1) = b(k) **THEN**
        - **frameNatcoord**(n) = natural coordinate where the moment is zero between b(k−1) and b(k)
        n = n + 1
    END IF
END DO
frameNatcoord(1) = −1
frameNatcoord(length(frameNatcoord) + 1) = 1

Figure 4.3: *Pseudocode for zmr.m*



(a) *The multipliers are calculated using linear shape functions to assure a constant stress state.*



(b) *The frame node are located where the moment changes sign.*

Figure 4.4: *Explanation of how the multipliers are calculated (a) and how the frame nodes are found (b).*

4.3.2 The L-matrix

Each RVE boundary has their own L-matrix which relate the boundary DOFs to the frame DOFs. For Γ_1 the expression reads

$$\mathbf{L}_1 = \int_{\Gamma_b} \mathbf{N}_{1\lambda}^T \mathbf{N}_b \, \mathrm{d}\Gamma.$$

Since the Lagrange multiplier shape functions are Dirac delta functions located at the boundary nodes, the integration is done directly by reading off the values of the frame node shape functions corresponding to the boundary node coordinate **xi**. The frame node shape functions are piecewise linear functions. An example of the integration is illustrated in figure 3.8 and the result is two rows in the following L-matrix.

$$\begin{bmatrix} N_{1\lambda}^1 & 0 \\ 0 & N_{1\lambda}^1 \\ N_{1\lambda}^2 & 0 \\ 0 & N_{1\lambda}^2 \\ N_{1\lambda}^3 & 0 \\ 0 & N_{1\lambda}^3 \end{bmatrix} \underbrace{\begin{bmatrix} N_b^1 & 0 & N_b^2 & 0 & N_b^3 & 0 & N_b^4 & 0 \\ 0 & N_b^1 & 0 & N_b^2 & 0 & N_b^3 & 0 & N_b^4 \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ 0 & 0 & N_b^2(\xi) & 0 & N_b^3(\xi) & 0 & 0 & 0 \\ 0 & 0 & 0 & N_b^2(\xi) & 0 & N_b^3(\xi) & 0 & 0 \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{bmatrix}}_{\mathbf{L}_1}$$

Figure 4.5 shows how the L-matrix is established in the program. The value **Nb** that goes into the matrix is found calling the function *shapefunc_frame.m*. **Nb** is equal to zero if **xi** is outside the frame shape function, otherwise the value is calculated from the linear frame shape function which has a value of 1 at frame node j and 0 at the two adjacent frame nodes.

The L-matrix

IF gamma = 1 **THEN**
    id = node numbers for boundary 1
    xy = y-coordinates for boundary 1 nodes
ELSEIF gamma = 2 **THEN**
    id = node numbers for boundary 2
    xy = y-coordinates for boundary 2 nodes
ELSEIF gamma = 3 **THEN**
    id = node numbers for boundary 3
    xy = x-coordinates for boundary 3 nodes
ELSE
    id = node numbers for boundary 4
    xy = x-coordinates for boundary 4 nodes
END IF

- lambdaNatcoord = Natural coordinates of boundary nodes

L = zeros(2∗number of boundary nodes , 2∗number of frame nodes)
FOR i = 1 **TO** i = number of boundary nodes **DO**
    FOR j = 1 **TO** j = number of frame nodes **DO**
        xi = lambdaNatcoord(i)
        - Call shapefunc_frame.m to get **Nb** = value of the shape function of frame node j at xi

        L(2∗i , 2∗j) = Nb
        L(2∗i−1 , 2∗j−1) = Nb
    END DO
END DO

Figure 4.5: *Pseudocode for L_matrix.m*

4.3.3 The B-matrix

The B-matrix relates the RVE boundary DOFs to all RVE DOFs. The matrix is unique for each boundary and the expression for Γ_1 is

$$\mathbf{B}_1 = \int_{\Gamma_b} \mathbf{N}_{1\lambda}^T \mathbf{N}_{1b} \, \mathrm{d}\Gamma.$$

Figure 4.6 explains how the B-matrix is created. Since the boundary nodes are a selection of the RVE nodes, the matrix becomes a Boolean matrix. That is, there is only zeroes except for the entries where the boundary node and the RVE node is the same. The value at that entry becomes 1, since the Dirac delta function lines up with the RVE shape function which is 1 at the node.

The B-matrix

IF gamma = 1 **THEN**
    id = node numbers for boundary 1
ELSEIF gamma = 2 **THEN**
    id = node numbers for boundary 2
ELSEIF gamma = 3 **THEN**
    id = node numbers for boundary 3
ELSE
    id = node numbers for boundary 4
END IF

B = zeros(2∗number of boundary nodes , 2∗number of RVE nodes)
FOR i = 1 **TO** i = number of boundary nodes **DO**
    B(2∗i , 2∗id(i)) = 1
    B(2∗i−1 , 2∗id(i)−1) = 1
END DO

- Eliminate the columns in B corresponding to the constrained corner DOFs

Figure 4.6: *Pseudocode for B_matrix.m*

In *B_matrix.m*, the current boundary node numbers are found in the vector **id**. The vector is used to place the ones in the correct entries in the matrix. For the boundary in figure 3.8, assuming that the whole RVE consists of six nodes, the boundary node numbers are $\mathbf{id} = [\,2 \quad 4 \quad 6\,]$. Hence the B-matrix becomes

$$\begin{bmatrix} N_{1\lambda}^1 & 0 \\ 0 & N_{1\lambda}^1 \\ N_{1\lambda}^2 & 0 \\ 0 & N_{1\lambda}^2 \\ N_{1\lambda}^3 & 0 \\ 0 & N_{1\lambda}^3 \end{bmatrix} \underbrace{\begin{bmatrix} N_{1b}^1 & 0 & N_{1b}^2 & 0 & N_{1b}^3 & 0 & N_{1b}^4 & 0 & N_{1b}^5 & 0 & N_{1b}^6 & 0 \\ 0 & N_{1b}^1 & 0 & N_{1b}^2 & 0 & N_{1b}^3 & 0 & N_{1b}^4 & 0 & N_{1b}^5 & 0 & N_{1b}^6 \end{bmatrix}}_{\mathbf{B}_1}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

To handle the constrained corners in the calculation, the rows and columns corresponding to the corner DOFs are eliminated in $\mathbf{K}$ and $\mathbf{r}$. Thus the columns in the B-matrices corresponding to the corner DOFs must also be eliminated for the A-matrix to end up correctly. For the B-matrix above, this means that the first four and the last four columns are eliminated.

4.3.4 The sigmaHomo-vector

After the displacement vector **d** is calculated, the stresses need to be found in order to calculate the homogenized constitutive matrix. Each component in the $\tilde{L}$-matrix is found from equation (4.1). The integration is done numerically as

$$\tilde{L}_{ijmn} = \frac{1}{|\Theta|} \sum_{k=1}^{n_G} \sigma_{ij}^{mn}(x_k, y_k) \mathrm{J}(x_k, y_k) \mathrm{w}(x_k, y_k),$$

where n_G is the number of Gauss points and $\sigma_{ij}^{mn}(x_k, y_k)$, $\mathrm{J}(x_k, y_k)$ and $\mathrm{w}(x_k, y_k)$ are respectively the stress, the Jacobian and the Gauss weight in the Gauss point (x_k, y_k). The Jacobian is the determinant of the Jacobian matrix and the Gauss weight has a value of 1 for the 2×2 Gauss integration. The sum in the expression above sums up the stress in each Gauss point for each stress component. This can also be written for

The sigmaHomo-vector

Input:
- eps = initial unit strain
- d = total displacement vector including corner DOFs

Gauss = $[-1\ 1\ 1\ -1\ ;\ -1\ -1\ 1\ 1\ ]\ /\sqrt{3}$
w = 1
vol = 0
sigma = zeros(3 , 1)
FOR i = 1 **TO** i = number of elements **DO**
    De = constitutive matrix for element i
    id = DOF numbers for element i
    de = d(id) = nodal displacements of element i
    FOR j = 1 **TO** j = number of Gauss points = 4 **DO**
        xi = Gauss(1 , j)
        eta = Gauss(2 , j)
        - Call dispstrain_B.m to get displacement-strain matrix **dsB** and Jacobian **J** for Gauss point j in element i

        sigmaGauss = De $*$ (dsB $*$ de + eps)
        sigma = sigma + sigmaGauss $*$ J $*$ w
        vol = vol + J
    END DO
END DO

sigmaHomo = sigma./vol

Figure 4.7: *Pseudocode for sigmaHomo_vector.m*

all three components for the given strain load case as

$$\boldsymbol{\sigma} = \begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \sum_{k=1}^{n_G} \boldsymbol{\sigma}_k \, \mathrm{J}_k \, \mathrm{w},$$

where $\boldsymbol{\sigma}_k$ and J_k are the stress vector and the Jacobian in every Gauss point k, respectively. The stress vector can be calculated from

$$\boldsymbol{\sigma}_k = \mathbf{D}^e\big(\mathbf{B}_k\mathbf{d}^e + \boldsymbol{\varepsilon}_0\big),$$

where $\mathbf{B}_k$ is the displacement-strain matrix in each Gauss point and the superscript e shows that the matrix or vector applies for the whole element and not only the Gauss point. $\boldsymbol{\varepsilon}_0$ is the initial unit strain vector. As seen in figure 4.7, $\boldsymbol{\sigma}_k$ is named **sigmaGauss**, $\boldsymbol{\sigma}$ is named **sigma**, $\mathbf{B}_k$ is named **dsB** and the Jacobian is named **J**.

The displacement-strain matrix **dsB** and the Jacobian **J** are found calling the function *dispstrain_B.m*. Inside this function the element is transformed into an isoparametric element with natural coordinates ξ and η, as shown in figure 4.8. This transformation makes it possible to calculate the dsB-matrix for all types of element shapes. The four Gauss points are located at $\pm\frac{1}{\sqrt{3}}$ and are given as input to *dispstrain_B.m* through **xi** and **eta**.

Finally, the vector **sigmaHomo** is calculated by dividing each component in **sigma** by the total RVE volume **vol**. The volume is found by summing up the Jacobian for each Gauss point. The sigmaHomo-vector is then given to the main script and placed as a column in **HomoL**. There are one sigmaHomo-vector for each unit strain state and together they make up the total homogenized constitutive matrix **HomoL**.

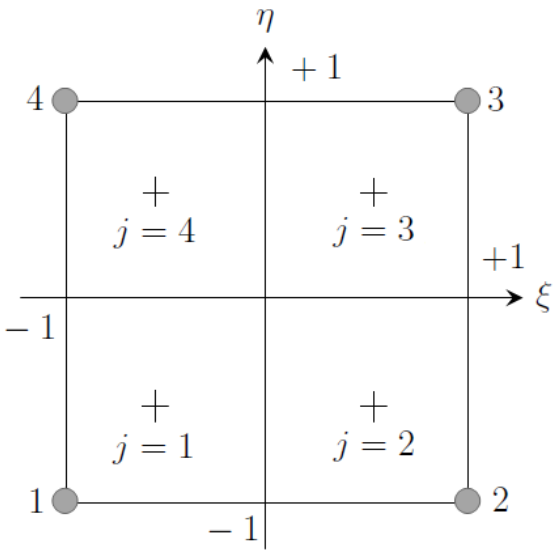


Figure 4.8: *Isoparametric element with the four Gauss points.*

4.4 Implementation of MPC in MATLAB

A program for solving two-scale linear homogenization problems with the transformation method has been implemented in MATLAB with the same template as with LLM. Figure 4.9 shows the main script. The pre-process is to calculate the constitutive matrix **D**, the transformation matrix **T** and the stiffness matrix **K**. The modified stiffness matrix and load vector corresponding to the master DOFs, $\mathbf{K}_m$ and $\mathbf{r}_m$, are calculated as derived in subsection 3.1.1. In the code they are named **Km** and **rm** respectively. All the functions are the same as the ones used in the LLM code, except for *T_matrix.m* which is described in detail in the following subsection.

4.4.1 The T-matrix

The K-matrix and the r-vector are transformed before the displacements are found, and the transformation is done with the T-matrix expressed as

$$T = \begin{bmatrix} \mathbf{I} \\ -\mathbf{G}_s^{-1}\mathbf{G}_m \end{bmatrix},$$

where **I** is the identity matrix. $\mathbf{G}_s$ and $\mathbf{G}_m$ contain the constants in the constraint equations that are in front of the slave DOFs and the master DOFs respectively, as explained in subsection 3.1.2.

In the pseudocode given in figure 4.10, it is seen that Γ_2 and Γ_4 are the slave boundaries and hence Γ_1 and Γ_3 the master boundaries. The first loop is over the boundary 4 slave nodes and the second is over the boundary 2 slave nodes, making **Gs** a diagonal matrix. First, inside the loops, the two slave DOF constants are placed in the G-matrix. Then the four master DOF constants are placed in the correct entries with the value **alpha** or **1−alpha**, depending on it is the upper or lower master node, respectively. The value **alpha** and the upper master node **masternode** are found calling the function *mpc_alpha.m*.

After the two loops is done, **Gs** and **Gm** are found and used to calculate the T-matrix. Finally, the columns corresponding to the constrained corner DOFs are eliminated.

Two-scale Linear Homogenization with MPC

Input Micro Problem
- Perimeter coordinates
- Young's Modulus and Poisson's Ratio for each element

Input Macro Problem
- Perimeter coordinates
- Boundary conditions - **BCmacro**
- Tractions - **rmacro**

RVE Analysis
- Calculate the constitutive matrix **D** for all elements
- Call T_matrix.m to get **T**
- Call K_matrix.m to get **K**

Km = T^T* K * T

HomoL = zeros(3 , 3)
FOR m = 1 **TO** m = 3 **DO**
    eps = zeros(3 , 1)
    eps(m , 1) = 1
    - Call r_matrix.m to get **r**
    rm = T^T* r
    dm = Km\rm
    d = T * dm
    - Call sigmaHomo_vector.m to get **sigmaHomo**
    HomoL(: , m) = sigmaHomo
END DO

Macro Analysis
- Call macro_K_matrix.m to get **Kmacro**

u = Kmacro\rmacro
dmacro(BCmacro) = u

Figure 4.9: *Pseudocode for the main script using MPC.*

The T-matrix

- id1 = node numbers for boundary 1
- id3 = node numbers for boundary 3
- ids2 = slave node numbers for boundary 2 (excluding corner nodes)
- ids4 = slave node numbers for boundary 4 (excluding corner nodes)
- idm = master node numbers (excluding corner nodes, including internal nodes)
- slavedofs = slave DOFs
- masterdofs = master DOFs

G = zeros(2∗number of slave nodes , 2∗number of RVE nodes)
FOR i = 1 **TO** i = number of boundary 4 slave nodes **DO**
G(2∗i−1 , 2∗ids4(i)−1) = −1
G(2∗i , 2∗ids4(i)) = −1
- Call mpc_alpha.m to get **alpha** and **masternode** = the upper masternode where alpha is equal 1

G(2∗i−1 , 2∗id3(masternode)−1) = alpha
G(2∗i−1 , 2∗id3(masternode−1)−1) = 1 − alpha
G(2∗i , 2∗id3(masternode)) = alpha
G(2∗i , 2∗id3(masternode−1)) = 1 − alpha
END DO
c = number of boundary 4 slave nodes
FOR i = 1 **TO** i = number of boundary 2 slave nodes **DO**
G(2∗(i + c)−1 , 2∗ids2(i)−1) = −1
G(2∗(i + c) , 2∗ids2(i)) = −1
- Call mpc_alpha.m to get **alpha** and **masternode** = the upper masternode where alpha is equal 1

G(2∗(i + c)−1 , 2∗id1(masternode)−1) = alpha
G(2∗(i + c)−1 , 2∗id1(masternode−1)−1) = 1 − alpha
G(2∗(i + c) , 2∗id1(masternode)) = alpha
G(2∗(i + c) , 2∗id1(masternode−1)) = 1 − alpha
END DO

Gs = G(: , slavedofs)
Gm = G(: , masterdofs)
T = zeros(2∗number of RVE nodes , 2∗number of master nodes)
T(masterdofs , :) = identity matrix
T(slavedofs , :) = −inv(Gs) ∗ Gm

- Eliminate the columns in T corresponding to the constrained corner DOFs

Figure 4.10: *Pseudocode for T_matrix.m*

Chapter 5

Numerical Examples

In the first section, the MATLAB code is verified by comparing the results to known analytical solutions. The second section derives a way to inspect the reliability of the results in a graphical way, when an analytical solution is not at hand. RVEs with different geometry and different content are then analysed with LLM and MPC, for both matching and non-matching nodes, in the last sections.

5.1 Verification of Code

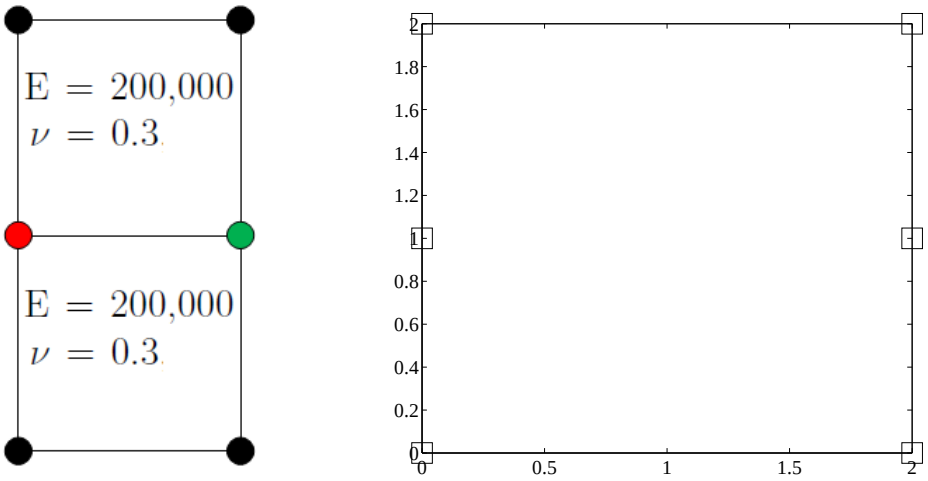
The code has to be verified before it can be used on numerical problems. If the calculated results in these basic problems are equal to the analytical solutions, the results of other more advanced numerical examples should be reliable as well.

5.1.1 Homogeneous Material with Matching Grids

An RVE that consists of elements with identical material properties is homogeneous, and the homogenized constitutive matrix $\tilde{\mathbf{L}}$ should become equal to the element constitutive matrix $\mathbf{D}$. Steel is a common construction material and has material properties E = 200,000 $\frac{N}{mm^2}$ and $\nu = 0.3$. The plane strain constitutive matrix for steel is thus

$$\mathbf{D}_{steel} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix}.$$

Figure 5.1(a) shows an RVE with two elements, both with the properties of steel. The RVE is modelled in MATLAB, as shown in figure 5.1(b), with the steel material,



(a) *Two elements with the same material properties.*

(b) *Grid used in MATLAB.*

Figure 5.1: *RVE with homogeneous material divided into two elements.*

and an RVE analysis with LLM and the transformation method(MPC) is carried out. Running with plane strain the homogenized constitutive matrices become

$$\tilde{\mathbf{L}}_{LLM} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}_{MPC} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix}.$$

The matrix from both methods is equal to $\mathbf{D}_{steel}$, verifying the code for homogeneous matching grids. The $\tilde{L}_{MPC}$-matrix is identical down to the last decimal, whereas the $\tilde{L}_{LLM}$-matrix is identical down to the sixth decimal. The discrepancy in the latter can be explained by the numerical number inserted in the diagonal of the A-matrix used in the LLM equation system. For the A-matrix to be invertible in MATLAB, the zeros in the diagonal are replaced by a small number. In this example the number is 10^{-18}, but it can also be chosen a lower number to get rid of the discrepancy.

5.1.2 Layered Material with Matching Grids

The RVE is heterogeneous if the elements have different material properties, and when the RVE is going to be treated as a single unit in the macro problem, the homogenized

constitutive matrix needs to be calculated correctly for the material properties to be preserved. Consider a material that consists of perfect layers with transverse isotropy. An analytical solution to such a problem can be found using Backus averaging. The Backus parameters in 2D are derived in appendix A and can be written

$$A = \left\langle a - \frac{f^2}{c} \right\rangle + \left\langle \frac{1}{c} \right\rangle^{-1} \left\langle \frac{f}{c} \right\rangle^2$$
$$C = \left\langle \frac{1}{c} \right\rangle^{-1}$$
$$F = \left\langle \frac{1}{c} \right\rangle^{-1} \left\langle \frac{f}{c} \right\rangle$$
$$L = \left\langle \frac{1}{l} \right\rangle^{-1}$$

where the brackets $\langle\rangle$ are the volume weighted average of the enclosed properties. That is, the enclosed properties are calculated for each layer and then averaged over all layers. a, c, f and l are components in the constitutive matrix for each layer, given as

$$\mathbf{D} = \begin{bmatrix} a & f & 0 \\ f & c & 0 \\ 0 & 0 & l \end{bmatrix}.$$

The single matrix that can describe the perfectly layered material correctly, and thus is the effective constitutive matrix for the material, becomes

$$\tilde{\mathbf{L}}_{Backus} = \begin{bmatrix} A & F & 0 \\ F & C & 0 \\ 0 & 0 & L \end{bmatrix}.$$

A code for calculating the Backus average, with Young's modulus and Poisson's ratio as input, has been implemented in MATLAB, and can be found in appendix B.5.

A ten layer RVE is shown in figure 5.2(a), where the material properties for each layer are constant and given in table 5.1. The resulting homogenized constitutive matrix using Backus averaging is

$$\tilde{\mathbf{L}}_{Backus} = \begin{bmatrix} 249.9792461 & 0.040642132 & 0 \\ 0.040642132 & 0.109754705 & 0 \\ 0 & 0 & 0.035351296 \end{bmatrix}.$$

Running the two main scripts with a ten layered model, illustrated in figure 5.2(b), and the layer properties as listed in table 5.1 give

$$\tilde{\mathbf{L}}_{LLM} = \begin{bmatrix} 249.9792461 & 0.040642132 & 0 \\ 0.040642132 & 0.109754705 & 0 \\ 0 & 0 & 0.035351296 \end{bmatrix}$$

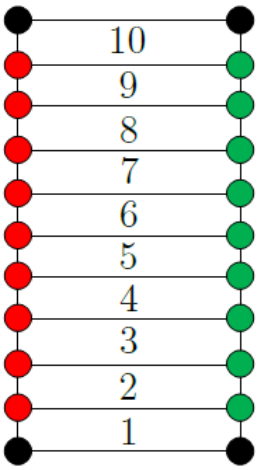
and

$$\tilde{\mathbf{L}}_{MPC} = \begin{bmatrix} 249.9792461 & 0.040642132 & 0 \\ 0.040642132 & 0.109754705 & 0 \\ 0 & 0 & 0.035351296 \end{bmatrix}.$$

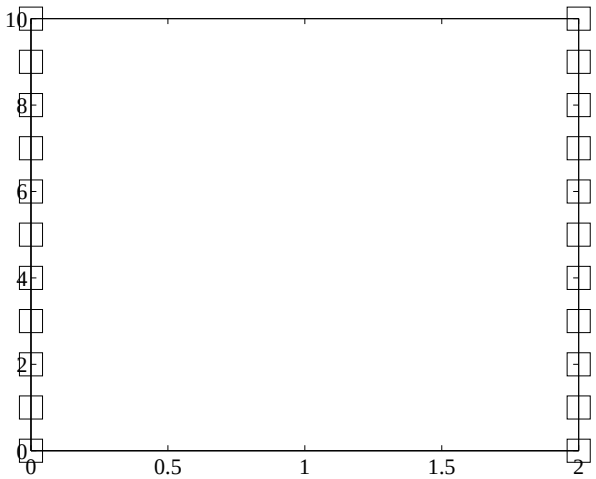
Both matrices are identical to the Backus solution down to the last decimal. This ensures that both codes work for heterogeneous RVEs with matching grids. It is also seen in figure 5.3 that the resulting nodal displacements using LLM or MPC are equal, despite that the kinematics in the LLM method is satisfied in a weak sense only. The reason is the matching grids, which makes the L-matrix for each boundary containing just ones since the frame nodes are located at the boundary nodes.

Table 5.1: *Layer properties.*

Layer	E $[\frac{N}{mm}]$	ν
1	100	0.450
2	1000	0.405
3	10	0.360
4	1	0.315
5	0.01	0.270
6	1000	0.225
7	0.1	0.180
8	10	0.135
9	100	0.090
10	1	0.045

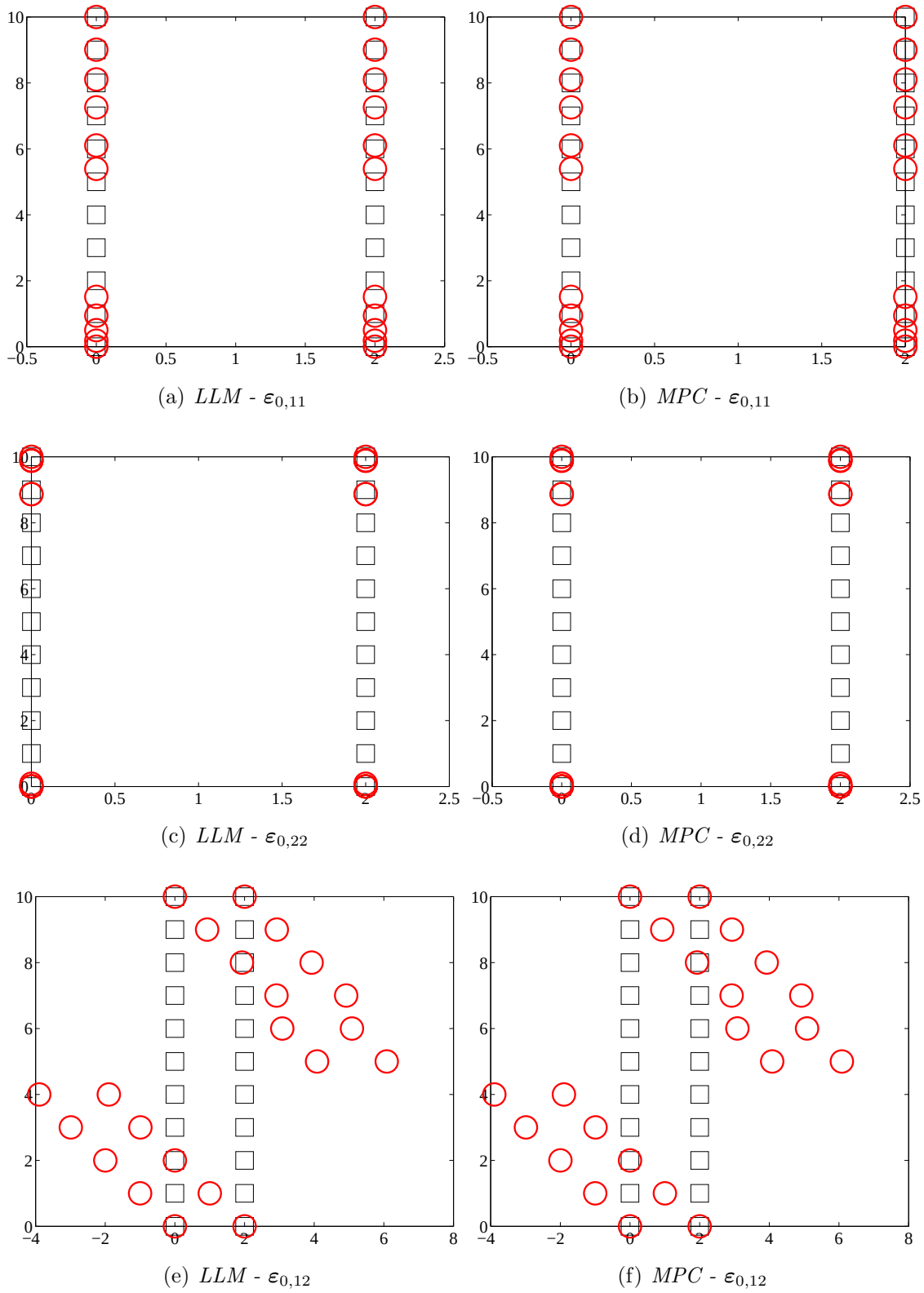


(a) *Ten layers with different material properties.*



(b) *Grid used in MATLAB.*

Figure 5.2: *RVE with ten different layers.*



(a) *LLM* - $\varepsilon_{0,11}$ (b) *MPC* - $\varepsilon_{0,11}$

(c) *LLM* - $\varepsilon_{0,22}$ (d) *MPC* - $\varepsilon_{0,22}$

(e) *LLM* - $\varepsilon_{0,12}$ (f) *MPC* - $\varepsilon_{0,12}$

Figure 5.3: *RVE node displacements due to the applied unit strain states using LLM and transformation(MPC).* $\varepsilon_{0,11}$ *loads in the horizontal x-direction and* $\varepsilon_{0,22}$ *in the vertical y-direction. The black squares and the red circles indicate the initial node coordinates and the displaced node coordinates, respectively.*

5.1.3 Homogeneous Material with Non-matching Grids

Assume the same homogeneous steel material and the same RVE as in subsection 5.1.1, but let the boundary nodes move along their boundaries independently of each other, as illustrated in figure 5.4. The hight of the RVE is 2 units, hence $\Delta_i \in \langle -1, 1 \rangle$. Four cases with non-matching boundary nodes are going to be analysed to examine to what degree the results are affected by different configurations of non-matching nodes. The homogenized constitutive matrices for each case are going to be compared to the steel element matrix, recall

$$\mathbf{D}_{steel} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix}.$$

Any discrepancy is measured with the relative difference in Frobenius norm, where the Frobenius norm is given as $\|\mathbf{L}\|_F = \sqrt{\sum_{i,j} L_{ij}^2}$ [15].

The four cases are

Case 1: $\Delta_1 = \Delta_2 = 0.1$

Case 2: $\Delta_1 = \Delta_2 = 0.5$

Case 3: $\Delta_1 = \Delta_2 = 0.9$

Case 4: $\Delta_1 = -0.1$ and $\Delta_2 = 0.9$

The four meshes are depicted in figure 5.5. Four-node quadrilateral elements do not handle trapezoidal shapes very well [4], so the RVEs are divided into smaller elements in the x-direction to minimize the error.

The mesh for case 1 is close to having matching boundary nodes. The analysis in MATLAB gives

$$\tilde{\mathbf{L}}^1_{LLM} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}^1_{MPC} = \begin{bmatrix} 264983.8155 & 113564.4924 & 37.2587 \\ 113564.4924 & 268450.7165 & 15.9680 \\ 37.2587 & 15.9680 & 76667.7422 \end{bmatrix}.$$

The superscript indicates which case the homogenized constitutive matrix belongs to. The LLM matrix is identical to the steel element matrix, while the MPC matrix deviates clearly. The relative difference in Frobenius norm is

$$\frac{\|\tilde{\mathbf{L}}^1_{MPC} - \mathbf{D}_{steel}\|_F}{\|\mathbf{D}_{steel}\|_F} = 0.01195.$$

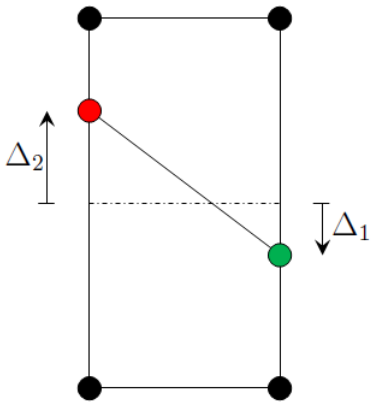
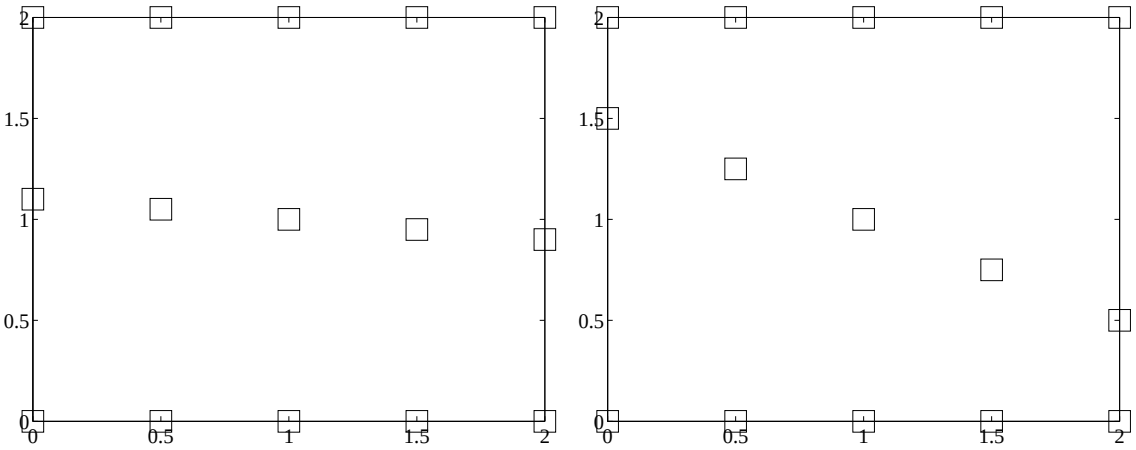


Figure 5.4: *RVE with homogeneous material and adjustable boundary nodes.*



(a) *Case 1* (b) *Case 2*

(c) *Case 3* (d) *Case 4*

Figure 5.5: *Four cases of non-matching grids.*

In the two following cases, the nodes will have a larger spacing to test the ruggedness of the LLM method and to see if the deviation using MPC gets any larger. The results for case 2 and 3 are

$$\tilde{\mathbf{L}}^2_{LLM} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix},$$

$$\tilde{\mathbf{L}}^2_{MPC} = \begin{bmatrix} 213323.8645 & 91424.5133 & 2287.9041 \\ 91424.5133 & 258962.1541 & 980.5303 \\ 2287.9041 & 980.5303 & 72716.0396 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}^3_{LLM} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix},$$

$$\tilde{\mathbf{L}}^3_{MPC} = \begin{bmatrix} 206347.7757 & 88434.7610 & 4563.7175 \\ 88434.7610 & 257680.8317 & 1955.8789 \\ 4563.7175 & 1955.8789 & 71995.7003 \end{bmatrix}.$$

The LLM matrices are the same as if it were matching grids. The relative differences for the MPC-matrices are

$$\frac{\|\tilde{\mathbf{L}}^2_{MPC} - \mathbf{D}_{steel}\|_F}{\|\mathbf{D}_{steel}\|_F} = 0.15760 \quad \text{and} \quad \frac{\|\tilde{\mathbf{L}}^3_{MPC} - \mathbf{D}_{steel}\|_F}{\|\mathbf{D}_{steel}\|_F} = 0.17783,$$

which confirm that making the slave nodes follow the master boundary is just an approximation and that the approximation works best when the boundary nodes are almost matching.

Case 4 is different from the three other cases in the way that the grid is not symmetric about the horizontal axis. The internal frame node will consequently not be placed in the middle of the boundary nodes, as is the case for the first three LLM cases. The frame node locations calculated in MATLAB for case 3 and 4 are shown in figure 5.6. The results for case 4 become

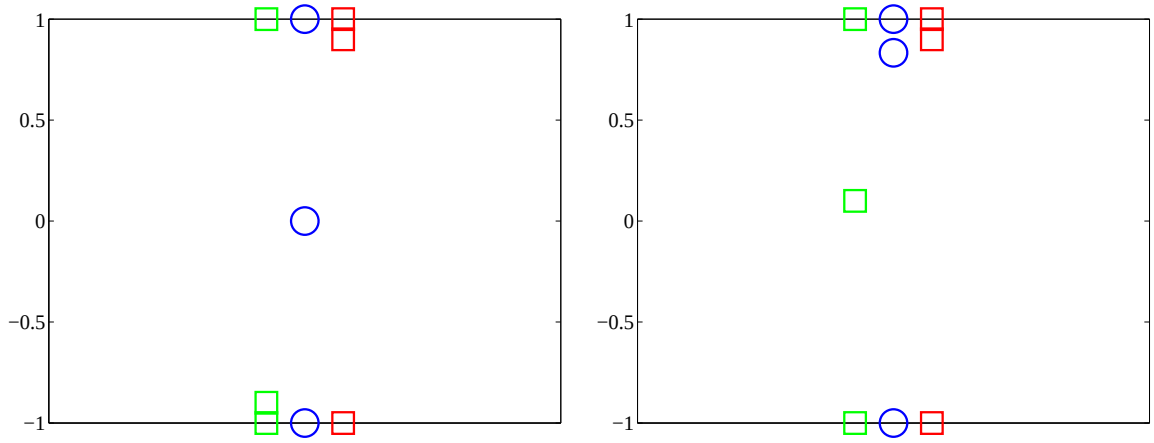
$$\tilde{\mathbf{L}}^4_{LLM} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix},$$

$$\tilde{\mathbf{L}}^4_{MPC} = \begin{bmatrix} 173380.6400 & 74305.9886 & 2911.1258 \\ 74305.9886 & 251625.6434 & 1247.6254 \\ 2911.1258 & 1247.6254 & 68251.7060 \end{bmatrix}.$$

LLM still gives the exact answer. The relative difference for the MPC matrix becomes

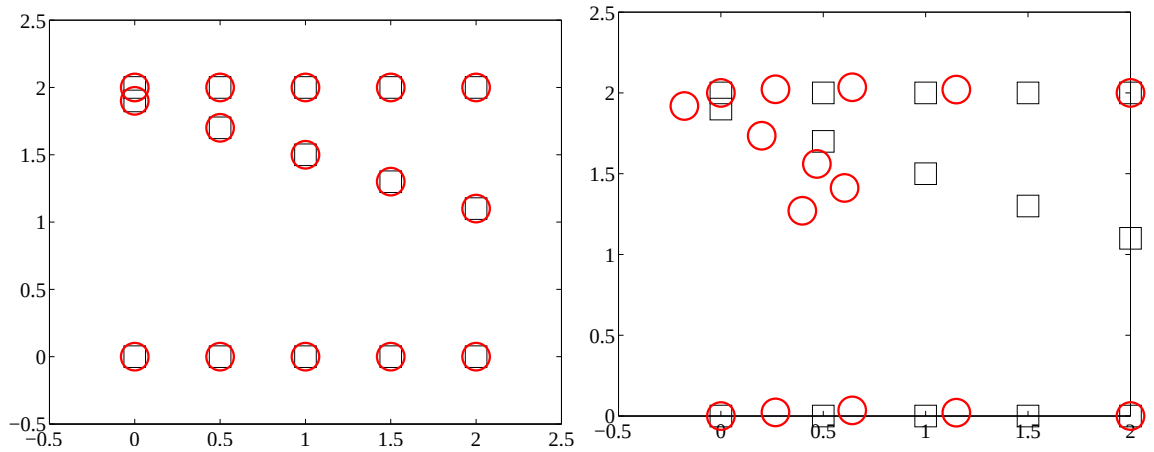
$$\frac{\|\tilde{\mathbf{L}}^4_{MPC} - \mathbf{D}_{steel}\|_F}{\|\mathbf{D}_{steel}\|_F} = 0.27028,$$

which is larger than in case 3, even though the spacing is smaller. The reason for the discrepancy when analysing non-matching grids with MPC is shown in figure 5.7(b), where it can be seen that the slave boundary does not follow the master boundary at all. The master boundary moves about 1.5 units to the left, while the slave boundary just moves a maximum of 0.2 units. The LLM method creates no displacements, because the RVE is in equilibrium with itself since the material is homogeneous.



(a) *Case 3 - frameNatcoord* $= [-1 \quad 0 \quad 1]$ (b) *Case 4 - frameNatcoord* $= [-1 \quad 0.8333 \quad 1]$

Figure 5.6: *Location of frame nodes for case 3 and 4. The green and red squares indicate the boundary 1 and 2 nodes respectively, whereas the blue circles mark the frame nodes.*



(a) *Case 4 - LLM* - $\varepsilon_{0,11}$ (b) *Case 4 - MPC* - $\varepsilon_{0,11}$

Figure 5.7: *Node displacements due to unit strain load in x-direction for case 4. The black squares and the red circles indicate the initial node coordinates and the displaced node coordinates, respectively.*

The frame nodes are placed according to the zero moment rule to guarantee correct transmission of constant stress states. When the material is homogeneous, the stresses are constant over the RVE, making the choice of frame node configuration insignificant. Running case 4 with the end frame nodes only, as shown in figure 5.8(a), results in

$$\tilde{\mathbf{L}}_{LLM}^{4,end} = \begin{bmatrix} 269230.7692 & 115384.6154 & 0 \\ 115384.6154 & 269230.7692 & 0 \\ 0 & 0 & 76923.0769 \end{bmatrix},$$

which is identical to $\mathbf{D}_{steel}$.

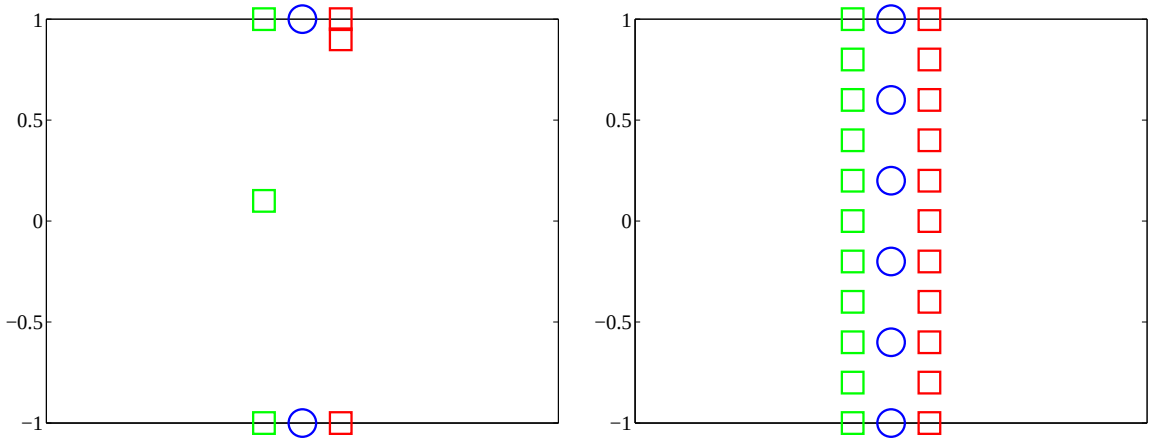
The layered RVE in subsection 5.1.2 is heterogeneous. Choosing frame node configurations as every other frame node, shown in figure 5.8(b), or just end nodes yield

$$\tilde{\mathbf{L}}_{LLM}^{everyother} = \begin{bmatrix} 252.5967575 & 5.6033042 & 0 \\ 5.6033042 & 15.5588765 & 0 \\ 0 & 0 & 5.0207867 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}_{LLM}^{end} = \begin{bmatrix} 389.0062792 & 218.3310325 & 0 \\ 218.3310325 & 389.0062792 & 0 \\ 0 & 0 & 85.3376233 \end{bmatrix}.$$

The matrices are not even comparable to $\tilde{\mathbf{L}}_{Backus}$, except for the $\tilde{L}_{1111}$-components. For a heterogeneous material, the stresses at the boundaries are varying, so the frame nodes are needed at every zero moment point to capture the change in stress between the different elements at the boundary.



(a) *Case 4 - end frame nodes* (b) *Ten layer material - every other frame node*

Figure 5.8: *Different frame node configurations. The green and red squares indicate the boundary 1 and 2 nodes respectively, whereas the blue circles mark the frame nodes.*

5.1.4 Non-matching Grids Inside Homogeneous Layers

Consider the the ten layer material from subsection 5.1.2. Let each of the two layers in the middle, layer 5 and 6, be divided into two elements, which have the same material properties as the layer. The nodes inside the two layers are non-matching, as seen from the boundaries in figure 5.9. The RVE is analysed with MPC and LLM, where the frame node configuration for the LLM method are chosen to be either all frame nodes or only frame nodes where the boundary nodes are matching, as seen in figure 5.9. The results become

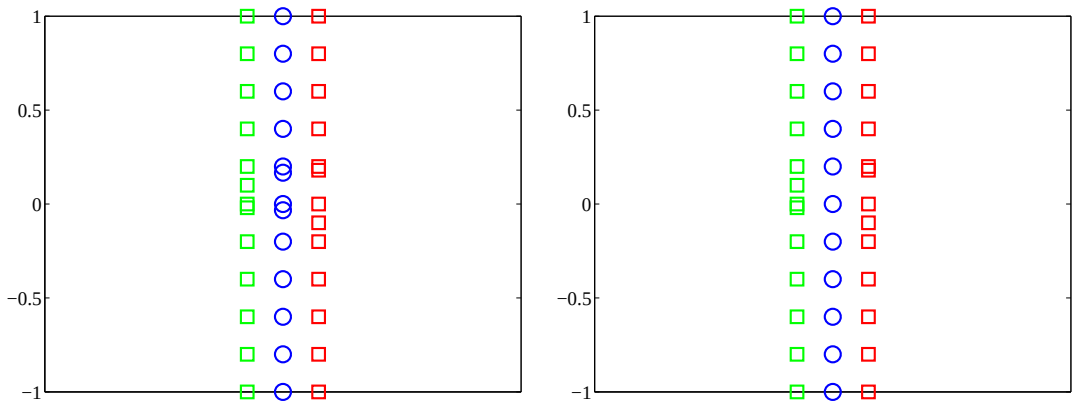
$$\tilde{\mathbf{L}}_{LLM}^{All} = \begin{bmatrix} 249.9792461 & 0.040642132 & 0 \\ 0.040642132 & 0.109754705 & 0 \\ 0 & 0 & 0.035351296 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Match} = \begin{bmatrix} 249.9792461 & 0.040642132 & 0 \\ 0.040642132 & 0.109754705 & 0 \\ 0 & 0 & 0.035351296 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}_{MPC} = \begin{bmatrix} 237.6427437 & 0.040096523 & 0.000012079 \\ 0.040096523 & 0.109527413 & -0.00001254 \\ 0.000012079 & -0.00001254 & 0.035259442 \end{bmatrix}.$$

Both of the LLM matrices are identical to $\tilde{\mathbf{L}}_{Backus}$, while MPC does not give the correct matrix. This shows that non-matching grids inside a homogeneous layer do not influence the result when using LLM, even if only the end frame nodes of the layers are used.



(a) *All possible frame nodes* (b) *Only frame nodes at matching nodes*

Figure 5.9: *Frame node configurations for the ten layer material where the two middle layers have non-matching nodes inside the layers. The green and red squares indicate the boundary 1 and 2 nodes respectively, whereas the blue circles mark the frame nodes.*

5.2 Wave Velocities

The homogenized constitutive matrix calculated in the RVE problem is suppose to be the best representation of the RVE when treated as one single unit. It can be difficult to see if the calculated matrix is correct or close to the correct solution, especially if the RVE consists of many layers and the layers are tilted. Without knowing if the homogenized constitutive matrix calculated is correct or not, it should at least lie in between an upper and a lower bound.

5.2.1 Upper and Lower Bounds

The elastic moduli can be approximated by assuming uniform strain in the material and averaging the relation for stress. Let S, E and L be the coarse scale stress, strain and elastic moduli respectively, and s, e and l be the same variables at fine scale. Then [9]

$$S = LE = \frac{1}{V}\int le \,\mathrm{d}V \approx \frac{1}{V}\int l \,\mathrm{d}V E = L_{Voigt}E,$$

where V is the volume. This was done by Voigt, and hence called the Voigt-moduli. The strain can be taken out from the integrand since it is assumed to be constant. In a similar way, done by Reuss, the elastic moduli can be approximated by assuming uniform stress in the material and averaging the expression for strain [9];

$$E = L^{-1}S = \frac{1}{V}\int l^{-1}s \,\mathrm{d}V \approx \frac{1}{V}\int l^{-1} \,\mathrm{d}V S = L^{-1}_{Reuss}S.$$

In an RVE, each element has constant material properties and the integral can thus be substituted by a sum, given as

$$L_{Voigt} = \frac{1}{V}\int l \,\mathrm{d}V = \sum_i \frac{1}{V}\int_{\Omega_i} l_i \,\mathrm{d}V_i = \sum_i l_i \frac{1}{V}\int_{\Omega_i} \mathrm{d}V_i = \sum_i l_i f_i,$$
$$L^{-1}_{Reuss} = \frac{1}{V}\int l^{-1} \,\mathrm{d}V = \sum_i \frac{1}{V}\int_{\Omega_i} l_i^{-1} \,\mathrm{d}V_i = \sum_i l_i^{-1} \frac{1}{V}\int_{\Omega_i} \mathrm{d}V_i = \sum_i l_i^{-1} f_i,$$

where $f_i = \frac{V_i}{V}$. The effective constitutive matrices for the material with Voigt and Reuss averaging, respectively, become

$$\tilde{\mathbf{L}}_{Voigt} = \sum_i \mathbf{l}_i f_i,$$
$$\tilde{\mathbf{L}}_{Reuss} = \frac{1}{\sum_i \frac{1}{\mathbf{l}_i} f_i},$$

where $\mathbf{l}_i$ is the element constitutive matrix known as $\mathbf{D}^e$. It can be seen that the Voigt average is a parallel model and that the Reuss average is a series model, hence

resulting in the two bounds for the constitutive matrix. The parallel model is stiff and the Voigt matrix is thus the upper bound, whereas the series model is more flexible and Reuss thus the lower bound. The proof that they really are the upper and lower bounds can be found in [9].

5.2.2 Christoffel Equation

It can be difficult and cumbersome to compare the homogenized constitutive matrix with the Voigt and Reuss matrices and to see if it lies inside the bounds. A more graphical way to compare the matrices is achieved by calculating the wave speeds in the material from each constitutive matrix, and then display the velocities in a plot. The velocities are calculated for different propagation angles angles.

The velocity in a medium can be found starting with the wave equation. Newton's second law, force equals mass times acceleration, leads to [10]

$$\rho(x)\partial_t^2 u_i(x,t) = \partial_{x_j}\sigma_{ij}(x,t),$$

written on index notation where subscripts repeated more than once in a term is summed upon, in this case subscript j. x is the spatial location, t is the time, ρ is the density and $\partial_x = \frac{\partial}{\partial x}$. Substituting with Hooke's law, $\sigma_{ij}(x,t) = L_{ijkl}(x)\varepsilon_{kl}(x,t) = L_{ijkl}(x)\partial_{x_l}u_k(x,t)$, yields

$$\rho(x)\partial_t^2 u_i(x,t) = \partial_{x_j}\Big(L_{ijkl}(x)\partial_{x_l}u_k(x,t)\Big).$$

For a homogeneous medium, $\mathbf{L}$ and ρ are independent of spatial position and the equation becomes

$$\rho\partial_t^2 u_i(x,t) = L_{ijkl}\partial_{x_j}\partial_{x_l}u_k(x,t). \tag{5.1}$$

If u is the velocity field and U is the polarization vector, the kth component of u can be written as

$$u_k = U_k \mathrm{e}^{-\mathrm{i}\omega\left(t - \frac{n_j x_j}{V(n)}\right)},$$

where ω is the angular frequency, $V(n)$ is the wave propagation velocity and n is a unit vector in the wave propagation direction. Inserting this into (5.1) results in [10]

$$(\Gamma_{ik} - \rho V^2\delta_{ik})U_k = 0 \tag{5.2}$$

where

$$\Gamma_{ik} = L_{ijkl}n_j n_l,$$

where the former is the Christoffel equation and the latter the Christoffel matrix. For the Christoffel equation to be valid, the expression inside the round brackets has to

be zero, making this a eigenvalue-eigenvector problem. ρV^2 are the eigenvalues of $\mathbf{\Gamma}$ and the polarization vectors $\mathbf{U}$ are the corresponding eigenvectors.

The Christoffel matrix can be calculated as

$$\mathbf{\Gamma} = \mathbf{D}^T \tilde{\mathbf{L}} \mathbf{D},$$

where

$$\mathbf{D}^T = \begin{bmatrix} n_1 & 0 & n_2 \\ 0 & n_2 & n_1 \end{bmatrix}$$

in 2D. $\mathbf{\Gamma}$ becomes a 2×2 matrix, which results in two eigenvalues ρV^2, of which two different wave velocities V can be calculated. n_1 and n_2 are the wave polarization directions, and since $\mathbf{\Gamma}$ is symmetric, as seen in the Chrisoffel equation (5.2), the two eigenvectors will be mutually orthogonal [10]. The wave with polarization in the propagation direction is called the P-wave, for pressure wave, and the wave with polarization perpendicular to the propagation direction is called the S-wave, for shear wave [3].

The wave velocities and the Voigt and Reuss bounds are calculated in MATLAB, and the codes can be found in appendix B.3.11.

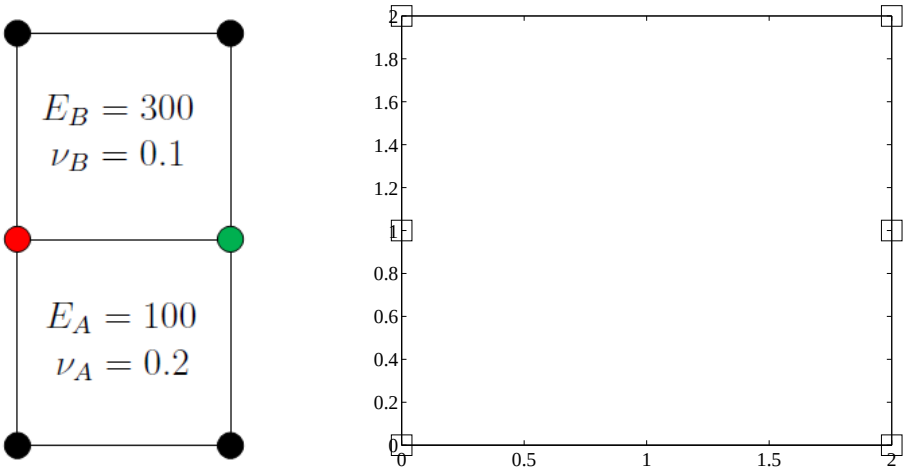
5.3 Flat Layered Materials

Consider a two layer RVE with matching grids. The two layers have different material properties, given as

$$\begin{aligned} E_A &= 100, \qquad \nu_A = 0.2, \\ E_B &= 300, \qquad \nu_B = 0.1. \end{aligned}$$

The RVE is shown together with the MATLAB model in figure 5.10. The resulting wave velocities for both LLM and MPC are graphically illustrated in figure 5.11, where it can be seen that the results for the two methods are identical for all propagation angles θ. The blue P-wave has the same velocity as the Voigt average for the angles $\theta = 0$ and $\theta = \pi$, which is naturally since the two materials then act in parallel. Similarly, when the wave direction is $\theta = \frac{\pi}{2}$ the materials act in series, thus the P-wave touches the Reuss bound.

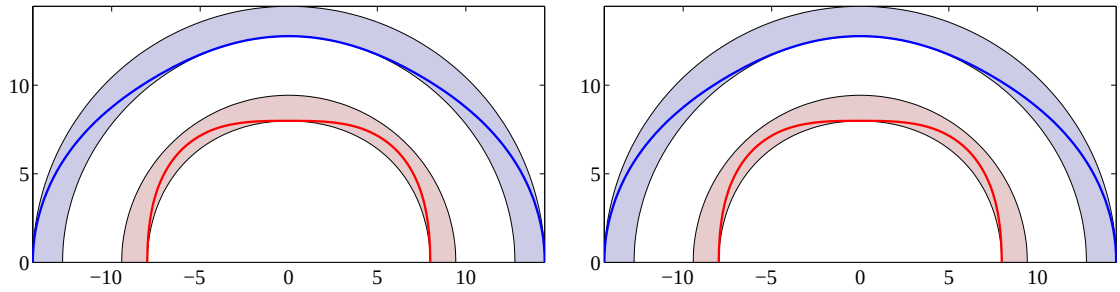
When the propagation direction is $\theta = 0$ and $\theta = \pi$, the S-wave will polarize in the vertical direction and hence act like the materials are in series. When the propagation direction is $\theta = \frac{\pi}{2}$, the S-wave polarizes in the horizontal direction. The S-wave will just go through material A before the wave is propagated half way, then it will go through material B only. In this way it acts like a series model, and hence touches the Reuss bound at $\theta = \frac{\pi}{2}$ as well.



(a) *Two layers with different material properties.*

(b) *Grid used in MATLAB.*

Figure 5.10: *RVE with two layers and matching grids.*



(a) *LLM*

(b) *MPC*

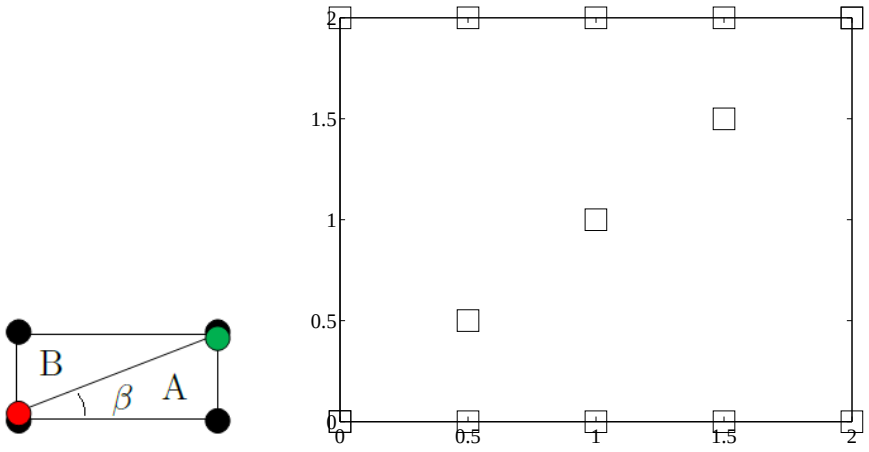
Figure 5.11: *Wave velocities for the perfectly layered RVE, shown for $\theta = [0, \pi]$. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

5.4 Tilted Layered Materials

Let an RVE consist of two different materials, material A and B, which are the same materials and thus have the same properties as in the previous section. The materials can be tilted in an angle $\beta \in \langle -90°, 90° \rangle$.

Figure 5.12 shows an RVE with 2 layers tilted $\beta = 45°$. The RVE is analysed with both LLM and MPC, and the homogenized constitutive matrices become

$$\tilde{\mathbf{L}}^2_{LLM} = \begin{bmatrix} 184.81 & 33.88 & 8.09 \\ 33.88 & 180.65 & 2.32 \\ 8.09 & 2.32 & 72.43 \end{bmatrix}$$



(a) *2 layers* (b) *Grid in MATLAB - 2 layers*

Figure 5.12: *RVE with 2 layers consisting of material A and B, and the corresponding grid in MATLAB with $\beta = 45°$. β is the angle of tilt. The two nodes that are close to the corner nodes is hard to see in the MATLAB grid, exactly because they are so close.*

and

$$\tilde{\mathbf{L}}^2_{MPC} = \begin{bmatrix} 161.03 & 30.02 & 6.27 \\ 30.02 & 181.14 & 2.71 \\ 6.27 & 2.71 & 70.13 \end{bmatrix},$$

where the superscript indicates the number of layers in the RVE. A reference solution for the two layer RVE has been calculated with 20×20 square elements and matching grids, where the 200 elements below the diagonal were selected to be material A. This yielded

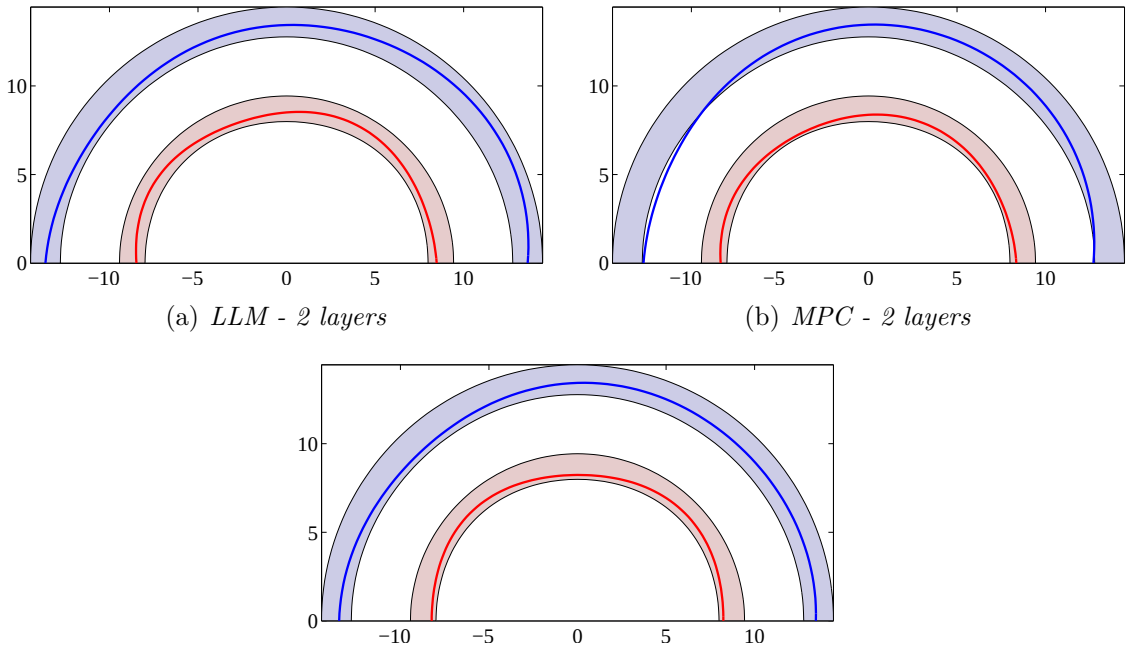
$$\tilde{\mathbf{L}}^2_{ref} = \begin{bmatrix} 181.12 & 33.34 & 2.78 \\ 33.34 & 180.40 & 2.74 \\ 2.78 & 2.74 & 67.90 \end{bmatrix}$$

using both LLM and MPC.

The wave velocities for the matrices are given in figure 5.13. It can be seen that the LLM wave velocity plot is fairly close to the reference plot. The relative difference in Frobenius norm is

$$\frac{\|\tilde{\mathbf{L}}^2_{LLM} - \tilde{\mathbf{L}}^2_{ref}\|_F}{\|\tilde{\mathbf{L}}^2_{ref}\|_F} = 0.03562.$$

It should be emphasized that the reference solution is not exact. The two layer RVE behaves the same if loaded in the x- or y-direction, and this is not the case for the reference solution, as seen in the reference solution matrix. The first diagonal component in the LLM matrix is relatively close to the reference solution, whereas in the



(a) *LLM - 2 layers* (b) *MPC - 2 layers*

(c) *Reference solution - 2 layers*

Figure 5.13: *Wave velocities for 2 layers where the materials are tilted 45°. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

MPC matrix it is not that close. The relative difference for the MPC matrix is

$$\frac{\|\tilde{\mathbf{L}}^2_{MPC} - \tilde{\mathbf{L}}^2_{ref}\|_F}{\|\tilde{\mathbf{L}}^2_{ref}\|_F} = 0.07945,$$

which is more than the double compared to the LLM matrix. The MPC P-wave velocity is also outside the bounds for some propagation angles.

At least some of the discrepancy shown in the LLM result can be explained by the defect of four-node quadrilateral elements when distorted as a trapezoid. The first and last element in figure 5.12(b) have a trapezoidal shape where one of the sides has a very short length. The partition of four elements along the x-axis is to minimize the error in the first place, but if the x-axis is divided into eight elements, as seen in figure 5.14, the result for LLM becomes

$$\tilde{\mathbf{L}}^2_{LLM,16elements} = \begin{bmatrix} 184.09 & 33.72 & 7.87 \\ 33.72 & 180.70 & 2.35 \\ 7.87 & 2.35 & 72.09 \end{bmatrix}.$$

The relative difference in Frobenius norm is

$$\frac{\|\tilde{\mathbf{L}}^2_{LLM,16elements} - \tilde{\mathbf{L}}^2_{ref}\|_F}{\|\tilde{\mathbf{L}}^2_{ref}\|_F} = 0.03261.$$

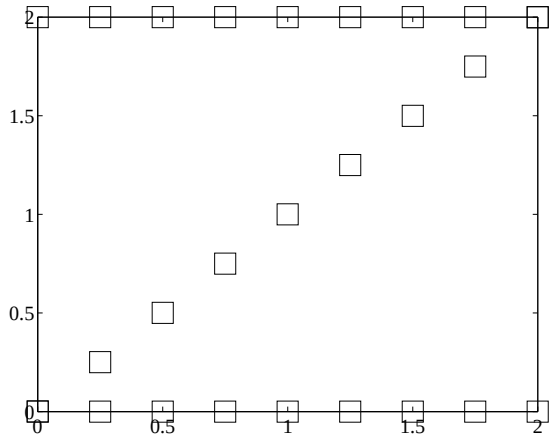


Figure 5.14: *RVE with 2 layers consisting 16 elements. The partition in the x-direction is to minimize the error from the trapezoidal shape of the quadrilateral elements.*

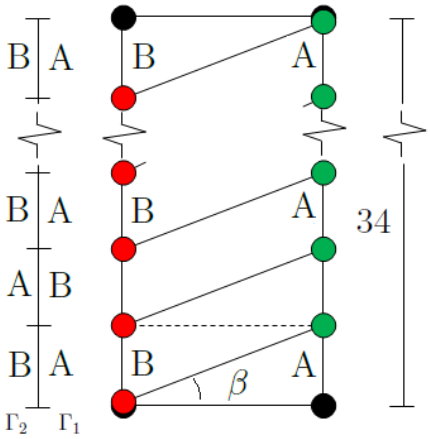
It is seen that the model with 16 elements is closer to the reference solution than the original model with 8 elements, but the difference between those two is not large, as seen when comparing the calculated relative differences in Frobenius norm. Figure 5.14 also shows that the elements get more narrow when partitioning in the x-direction, and hence obtain a more extreme trapezoidal shape in the other direction. So there is not much to gain by dividing the x-axis by more than four elements.

Now, consider an RVE that is 2 units wide and 34 units high, with layers consisting of every other material A and B, and a tilt of $\beta = 45°$. Three meshes, where the coupled boundaries have different composition of nodes and material, are going to be examined.

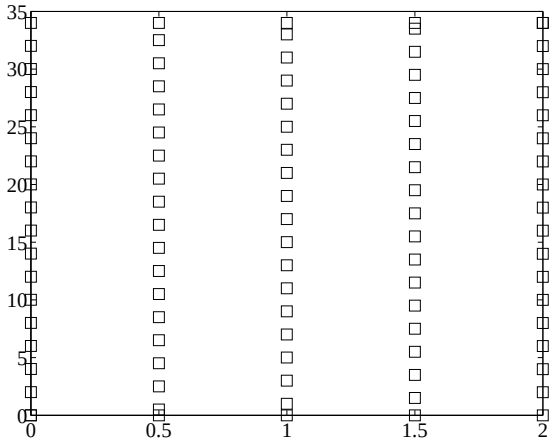
Mesh 1: The boundary nodes are matching, but the materials at the boundaries are not matching. That is, when Γ_1 consists of material A, Γ_2 consists of material B. The mesh is shown in figure 5.15(a). The MATLAB grid has to have 18 layers, all of which are 2 units thick, to describe this boundary situation for the given RVE and β.

Mesh 2: The boundary nodes are non-matching, and the materials are 50/50 matching/non-matching at the boundary. Figure 5.15(c) shows mesh 2 and how the boundaries are coupled. 10 layers with a thickness of 4 units are used in the analysis.

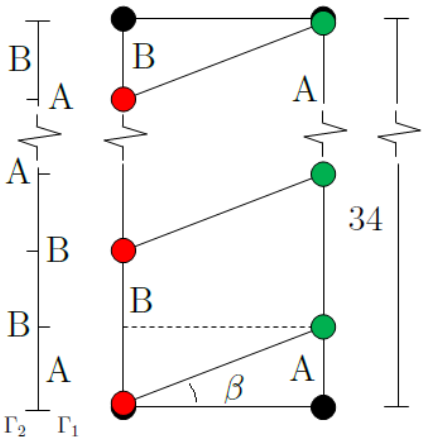
Mesh 3: The boundary nodes are matching, and the materials are matching. That is, when Γ_1 consists of material A, Γ_2 also consists of material A. It is seen in the mesh, given in figure 5.15(e), that the nodes are not matching in the end layers. This inaccuracy is a drawback that occurs when fulfilling the boundary situation. The MATLAB mesh has 34 layers, all of which are 1 unit thick.



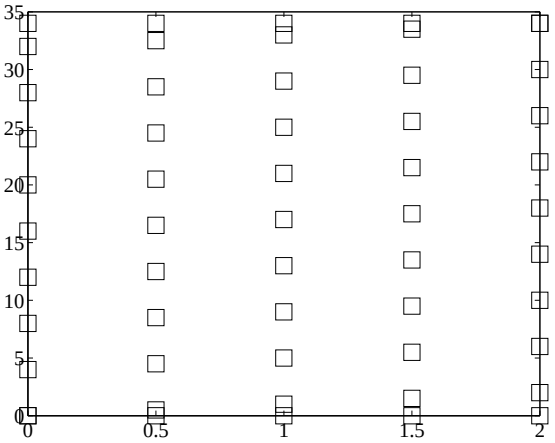
(a) *Mesh 1: Matching nodes, non-matching materials*



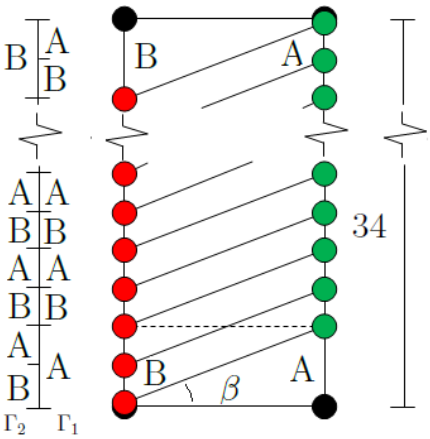
(b) *Grid in MATLAB - 18 layers*



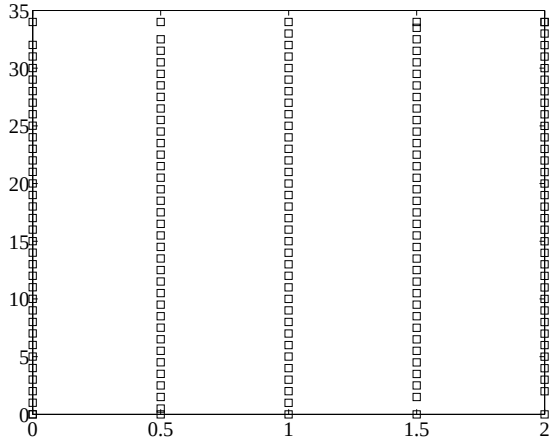
(c) *Mesh 2: Non-matching nodes, 50/50 matching materials*



(d) *Grid in MATLAB - 10 layers*



(e) *Mesh 3: Matching nodes, matching materials*



(f) *Grid in MATLAB - 34 layers*

Figure 5.15: *RVEs with different boundary situations, and their corresponding grids in MATLAB with $\beta = 45°$. The hight is 34 units and the width is 2 units. Each RVE has to have different thickness of the layers, and thus different number of layers, to create the different boundary situations without changing the angle of tilt and the size of the RVE. On the left, the materials at the boundaries are shown next to each other.*

Analysing the three meshes with LLM and MPC result in

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,45^\circ} = \begin{bmatrix} 172.54 & 43.35 & 11.02 \\ 43.35 & 171.26 & 10.96 \\ 11.02 & 10.96 & 77.78 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,1,\,45^\circ} = \begin{bmatrix} 189.53 & 34.89 & 6.29 \\ 34.89 & 185.97 & 8.06 \\ 6.29 & 8.06 & 81.21 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,45^\circ} = \begin{bmatrix} 172.02 & 42.82 & 10.27 \\ 42.82 & 171.09 & 10.49 \\ 10.27 & 10.49 & 77.08 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,2,\,45^\circ} = \begin{bmatrix} 180.30 & 32.54 & 3.98 \\ 32.54 & 183.52 & 5.64 \\ 3.98 & 5.64 & 77.67 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,45^\circ} = \begin{bmatrix} 172.72 & 43.42 & 11.12 \\ 43.42 & 171.34 & 11.08 \\ 11.12 & 11.08 & 77.93 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,3,\,45^\circ} = \begin{bmatrix} 169.97 & 42.03 & 10.64 \\ 42.03 & 171.58 & 10.26 \\ 10.64 & 10.26 & 77.22 \end{bmatrix}.$$

The corresponding wave velocity plots are depicted in figure 5.16.

First, it can be concluded from the similarity of the LLM matrices, that all three RVEs have the same homogenized behaviour despite different thickness of the layers and different number of layers, as long as the layers are every other material A and B, and the volume fractions of the two materials are the same. The similarity of the LLM matrices and wave velocity plots also show that the LLM method gives the same result regardless of boundary situation. The solution seems realistic as well, since the P- and S-wave velocity touch the Voigt bound and Reuss bound at $\theta = 45^\circ$, respectively, and both touch the Reuss bound at $\theta = 135^\circ$. This is where the materials act in parallel and series, respectively.

The variation among the LLM matrices can come from the influence the two end layers have in the three meshes. It can be seen in figure 5.15 that the end layers in mesh 3 are larger in size in proportion to the other layers in the mesh, whereas in mesh 2, the end layers are smaller.

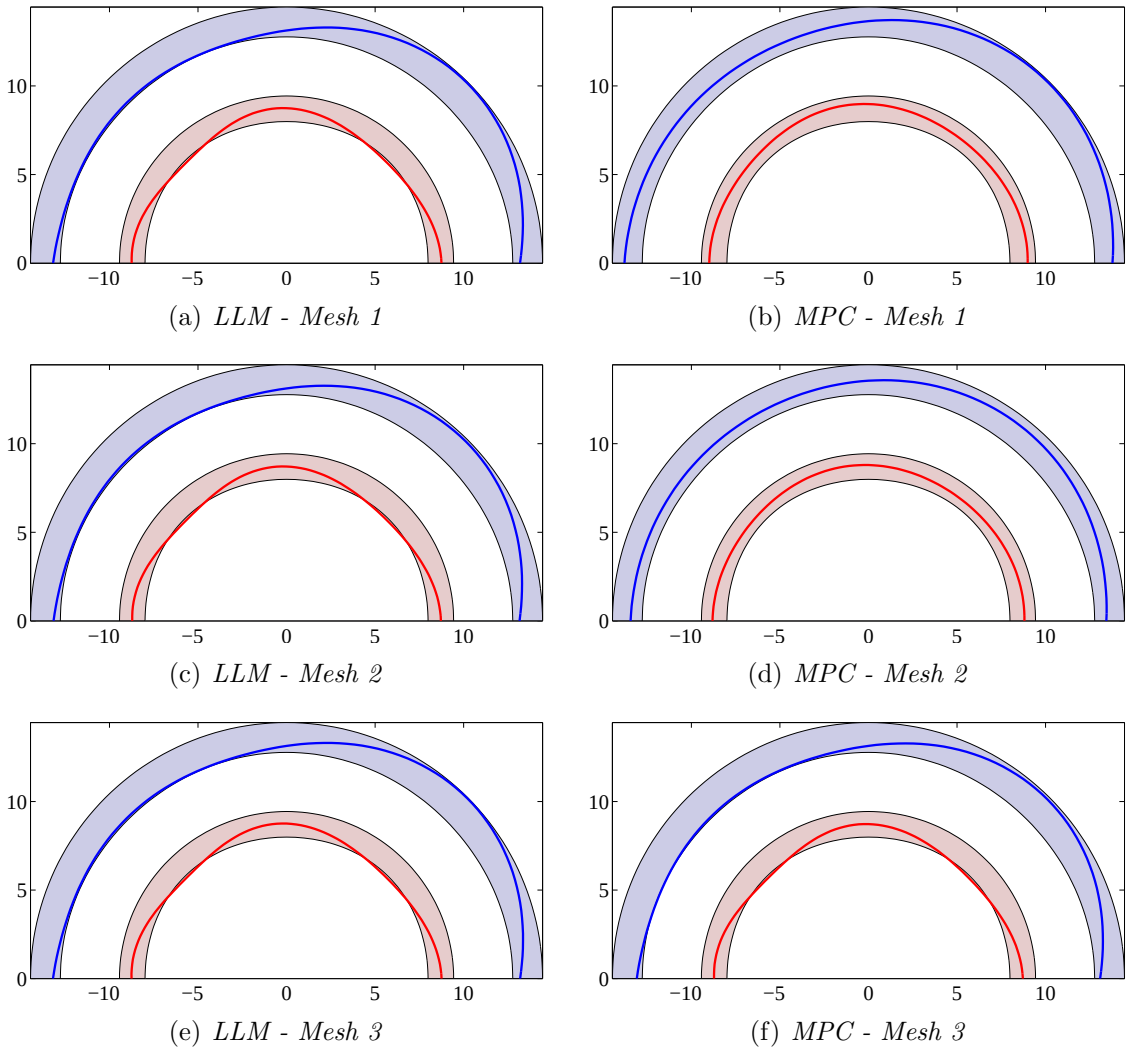
Second, the matrix resulting from the MPC analysis with mesh 3 seems much closer to the LLM results than for mesh 1 and 2. This is also seen in figure 5.16(f). The relative differences in Frobenius norm between the LLM matrix and MPC matrix for each mesh are

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,1,\,45^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,45^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,45^\circ}\|_F} = 0.10196,$$

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,2,\,45^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,45^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,45^\circ}\|_F} = 0.09023,$$

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,3,\,45^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,45^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,45^\circ}\|_F} = 0.01408,$$

which shows that the difference between LLM and MPC is much smaller for mesh 3, where both the boundary nodes and the materials match, than for mesh 1, where the



(a) *LLM - Mesh 1* (b) *MPC - Mesh 1*

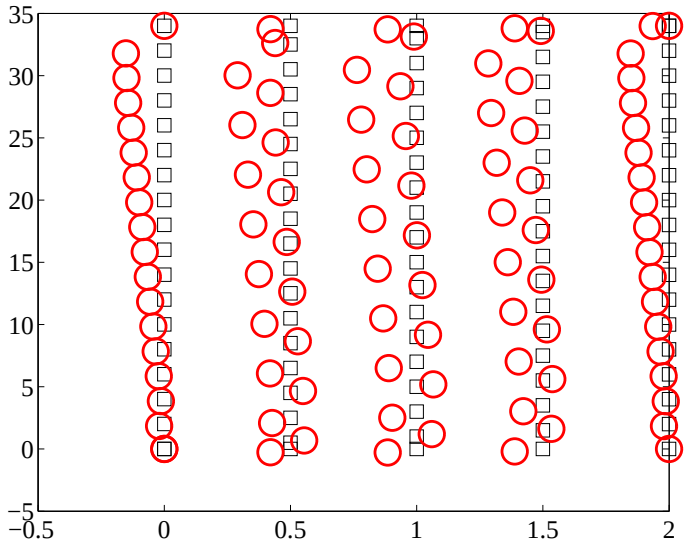
(c) *LLM - Mesh 2* (d) *MPC - Mesh 2*

(e) *LLM - Mesh 3* (f) *MPC - Mesh 3*

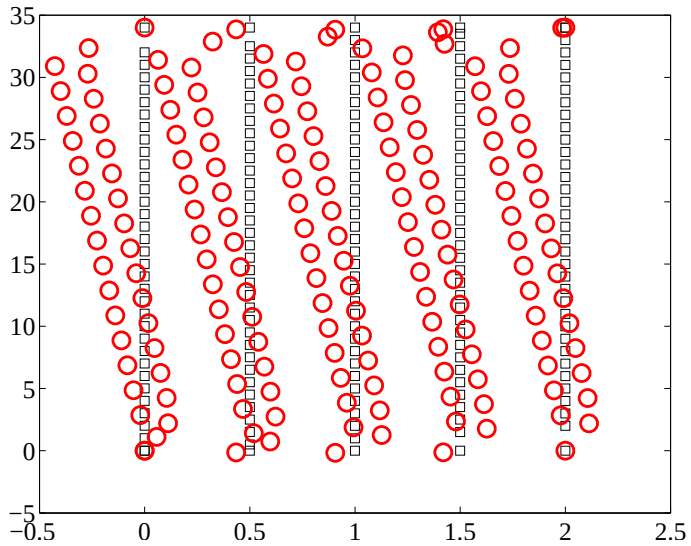
Figure 5.16: *Wave velocities for all three meshes, where the materials are tilted 45°. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

nodes are also matching, but not the materials. Even the difference for mesh 2, where the boundary nodes are non-matching, is smaller than for mesh 1.

The node displacements using MPC for mesh 1 are shown in figure 5.17(a), and it is seen that the slave boundary follows the master boundary exactly. This is also the case for mesh 3, as seen in figure 5.17(b). The displacements of the internal nodes for mesh 1 and 3 seem to roughly follow the same pattern. On the other hand, the boundary node displacements for mesh 1 are not comparable to the ones in mesh 3. The boundary node displacements for mesh 1 are on a straight line, because the coupling is the same for each layer. That is, the coupling is material A-B or B-A for every layer. In mesh 3, the coupling is either A-A or B-B, and hence the displacement



(a) *MPC - Mesh 1 -* $\varepsilon_{0,22}$



(b) *MPC - Mesh 3 -* $\varepsilon_{0,22}$

Figure 5.17: *Node displacements due to unit strain load in y-direction for mesh 1 and 3 using MPC. The displacement plots for the other two load states give the same displacement pattern. The black squares and the red circles indicate the initial node coordinates and the displaced node coordinates, respectively.*

patterns for the boundaries are equal to the displacement patterns for the internal nodes. This can be a reason why MPC gives the correct solution for mesh 3, but not for mesh 1.

The difference between the MPC matrix and the LLM matrix for mesh 3 can possibly be explained by the fact that the boundary nodes are not completely matching, as seen

in figure 5.15(e). It is seen in the lower left corner in figure 5.17(b) that the second slave node is located on the linear line between the two adjacent master nodes. The master node in the upper right corner is one of the three displaced nodes in the cluster at around 1.4 on the x-axis. It is the non-matching master node that stands out from the rest of the displaced nodes, and is affecting the result to some degree. However, the large number of layers makes the two non-matching nodes less significant and the result becomes fairly close to the LLM solution.

The three meshes, with the three different coupling situations, are now going to be analysed with $\beta = 26.565°$ and $\beta = 63.435°$. In the input code, the slope of the layers is $\frac{2}{x}$, so if the angle of tilt changes, it is the x-axis that will adjust. When $\beta = 45°$, the width of the RVE is 2 units. If $\beta = arctan\left(\frac{2}{4}\right) = 26.565°$, the width becomes 4 units, and if $\beta = arctan\left(\frac{2}{1}\right) = 63.435°$, the width becomes 1 unit. Thus, the RVEs in these cases are 34 units high and 4 units and 1 unit wide, respectively, and the meshes are given in figure 5.18.

The results of the 26.565° analysis become

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,26^\circ} = \begin{bmatrix} 190.26 & 38.34 & 15.29 \\ 38.34 & 163.11 & 2.17 \\ 15.29 & 2.17 & 72.81 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,1,\,26^\circ} = \begin{bmatrix} 196.52 & 34.56 & 9.65 \\ 34.56 & 174.97 & 3.24 \\ 9.65 & 3.24 & 77.87 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,26^\circ} = \begin{bmatrix} 188.46 & 38.23 & 14.45 \\ 38.23 & 163.15 & 2.18 \\ 14.45 & 2.18 & 72.57 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,2,\,26^\circ} = \begin{bmatrix} 189.35 & 32.99 & 7.04 \\ 32.99 & 173.30 & 2.07 \\ 7.04 & 2.07 & 75.11 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,26^\circ} = \begin{bmatrix} 190.75 & 38.35 & 15.43 \\ 38.35 & 163.12 & 2.15 \\ 15.43 & 2.15 & 72.83 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,3,\,26^\circ} = \begin{bmatrix} 188.61 & 37.52 & 14.74 \\ 37.52 & 163.61 & 1.78 \\ 14.74 & 1.78 & 72.51 \end{bmatrix},$$

where 26.565° is shortened to 26° in the superscript for notation simplicity. The wave velocity plots are given in figure 5.19, and the relative differences between the MPC and LLM matrix are

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,1,\,26^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,26^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,\,26^\circ}\|_F} = 0.06476,$$

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,2,\,26^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,26^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,\,26^\circ}\|_F} = 0.06236,$$

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,3,\,26^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,26^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,\,26^\circ}\|_F} = 0.01026.$$

The results for 63.435° are

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,63^\circ} = \begin{bmatrix} 164.25 & 38.59 & 2.31 \\ 38.59 & 189.94 & 15.68 \\ 2.31 & 15.68 & 73.04 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,1,63^\circ} = \begin{bmatrix} 186.41 & 32.47 & 2.47 \\ 32.47 & 199.04 & 8.38 \\ 2.47 & 8.38 & 79.46 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,63^\circ} = \begin{bmatrix} 164.39 & 38.37 & 2.20 \\ 38.37 & 189.37 & 15.01 \\ 2.20 & 15.01 & 72.73 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,2,63^\circ} = \begin{bmatrix} 169.23 & 28.91 & 0.95 \\ 28.91 & 196.28 & 5.85 \\ 0.95 & 5.85 & 75.62 \end{bmatrix},$$

$$\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,63^\circ} = \begin{bmatrix} 164.42 & 38.53 & 2.26 \\ 38.53 & 189.98 & 15.78 \\ 2.26 & 15.78 & 73.03 \end{bmatrix}, \quad \tilde{\mathbf{L}}_{MPC}^{Mesh\,3,63^\circ} = \begin{bmatrix} 160.74 & 37.29 & 2.18 \\ 37.29 & 189.84 & 14.87 \\ 2.18 & 14.87 & 72.35 \end{bmatrix},$$

where 63.435° is shortened to 63° for notation simplicity. These wave velocity plots are given in figure 5.20, and the relative differences between the MPC and LLM matrix are

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,1,63^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,1,63^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,1,63^\circ}\|_F} = 0.10528,$$

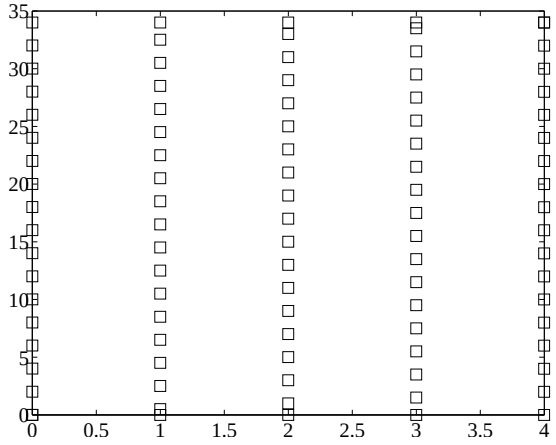
$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,2,63^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,2,63^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,2,63^\circ}\|_F} = 0.07745,$$

$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Mesh\,3,63^\circ} - \tilde{\mathbf{L}}_{LLM}^{Mesh\,3,63^\circ}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Mesh\,3,63^\circ}\|_F} = 0.01617.$$

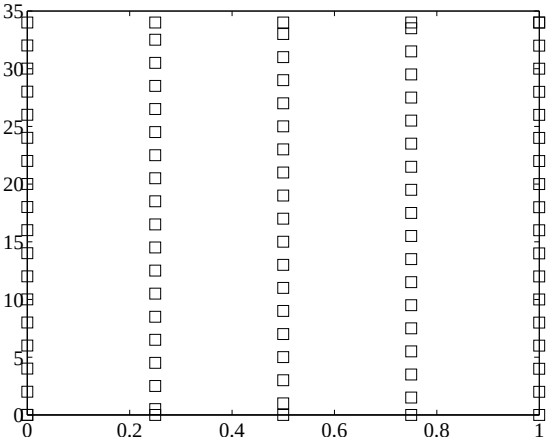
The outcomes are the same as for $\beta = 45^\circ$, in the way that the LLM matrices are fairly similar, and that the MPC matrix for mesh 3 is close to the LLM solutions. This is also seen from the wave velocity plots. It has to be emphasized how well the LLM method calculates the homogenized constitutive matrix regardless of boundary situation or angle of tilt. Looking at the LLM wave velocity plots, it is hard to tell the difference between the different meshes with the naked eye.

The relative differences in Frobenius norm for mesh 3 are 0.01026, 0.01408 and 0.01617 for angles 26°, 45° and 63°, respectively. This implies that MPC, when both the nodes and materials match, gets worse and worse for increasing angle of tilt. Another possibility is that the influence from the non-matching nodes in the end layers gets larger, since the spacing between the vertical nodes increases relatively to the width of the RVE.

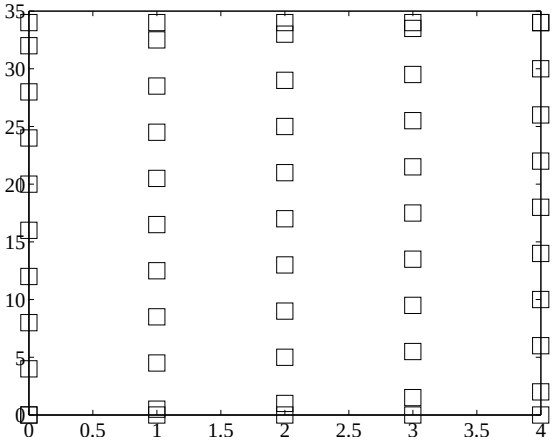
Mesh 2 is closer to the LLM solution for all tilt angles compared to mesh 1 when looking at the relative differences, even though mesh 1 has matching boundary nodes. However, when comparing the MPC wave velocity plots to the LLM plots, it can be seen that mesh 1 is closer for some propagation angles. If the propagation angle equals the tilt angle, i.e. $\theta = \beta$, the P-wave velocity for mesh 1 is exactly the same as the one



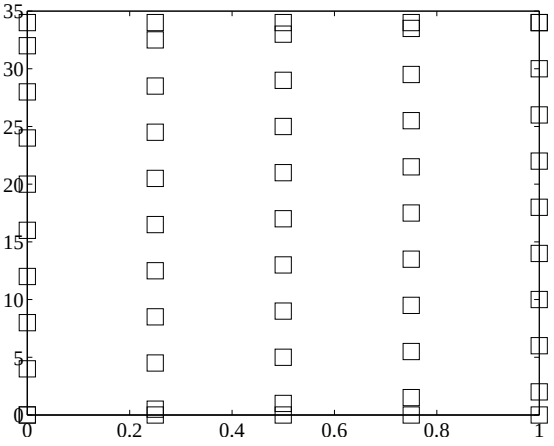
(a) *Grid in MATLAB - Mesh 1 - β = 26.565°*



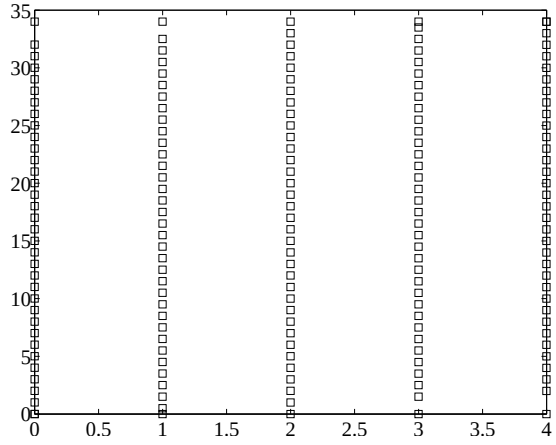
(b) *Grid in MATLAB - Mesh 1 - β = 63.435°*



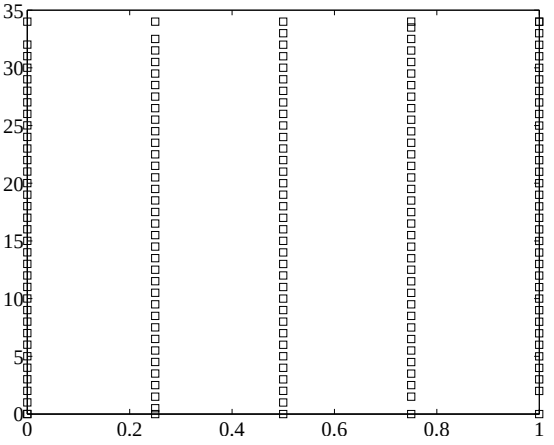
(c) *Grid in MATLAB - Mesh 2 - β = 26.565°*



(d) *Grid in MATLAB - Mesh 2 - β = 63.435°*

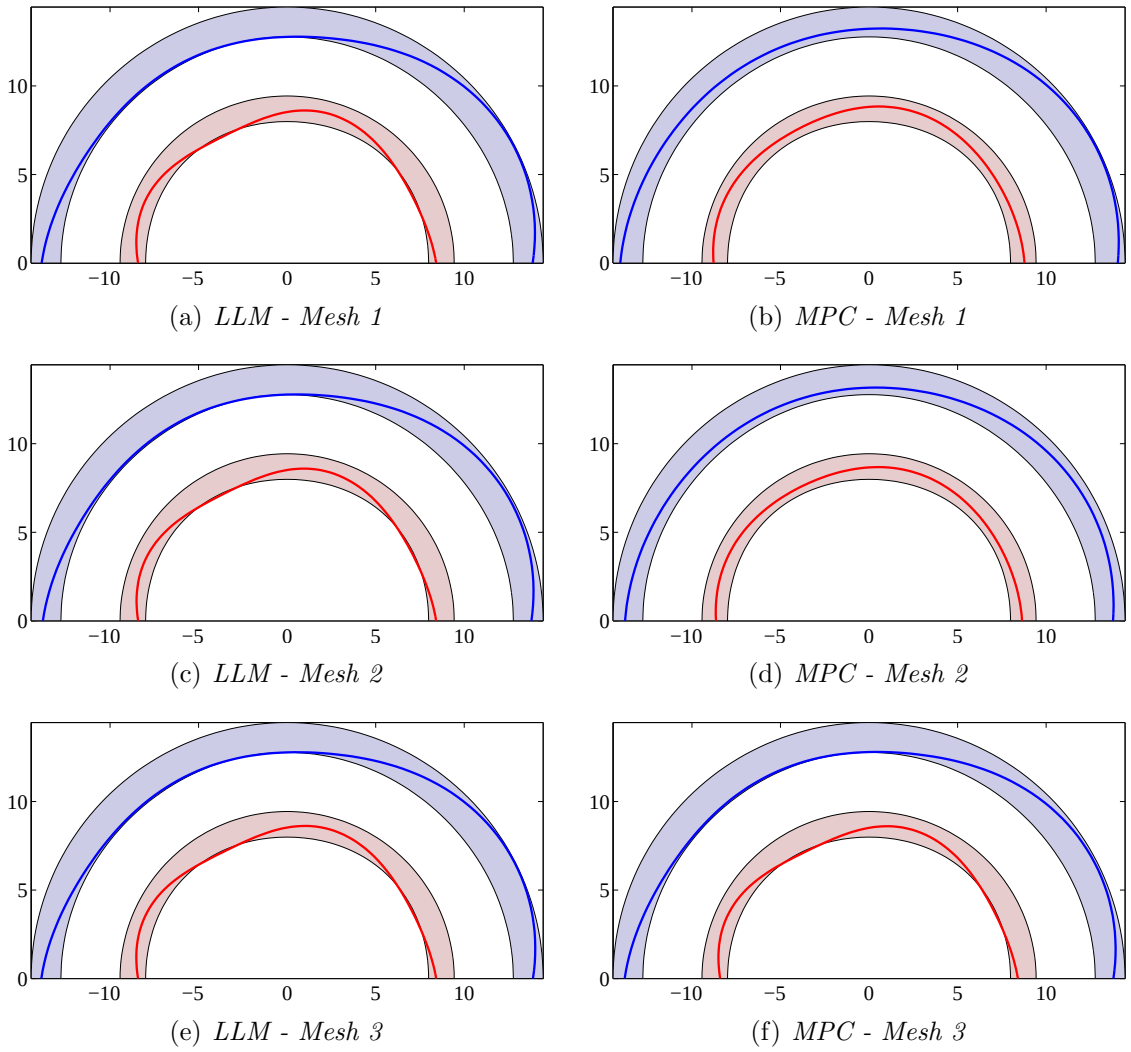


(e) *Grid in MATLAB - Mesh 3 - β = 26.565°*



(f) *Grid in MATLAB - Mesh 3 - β = 63.435°*

Figure 5.18: *Grids in MATLAB for the three meshes with β = 26.565° and β = 63.435° . The hight is 34 units and the width is 4 units and 1 unit, respectively. Each RVE has to have different thickness of the layers, and thus different number of layers, to create the different boundary situations without changing the angle of tilt and the size of the RVE.*



(a) *LLM - Mesh 1* (b) *MPC - Mesh 1*

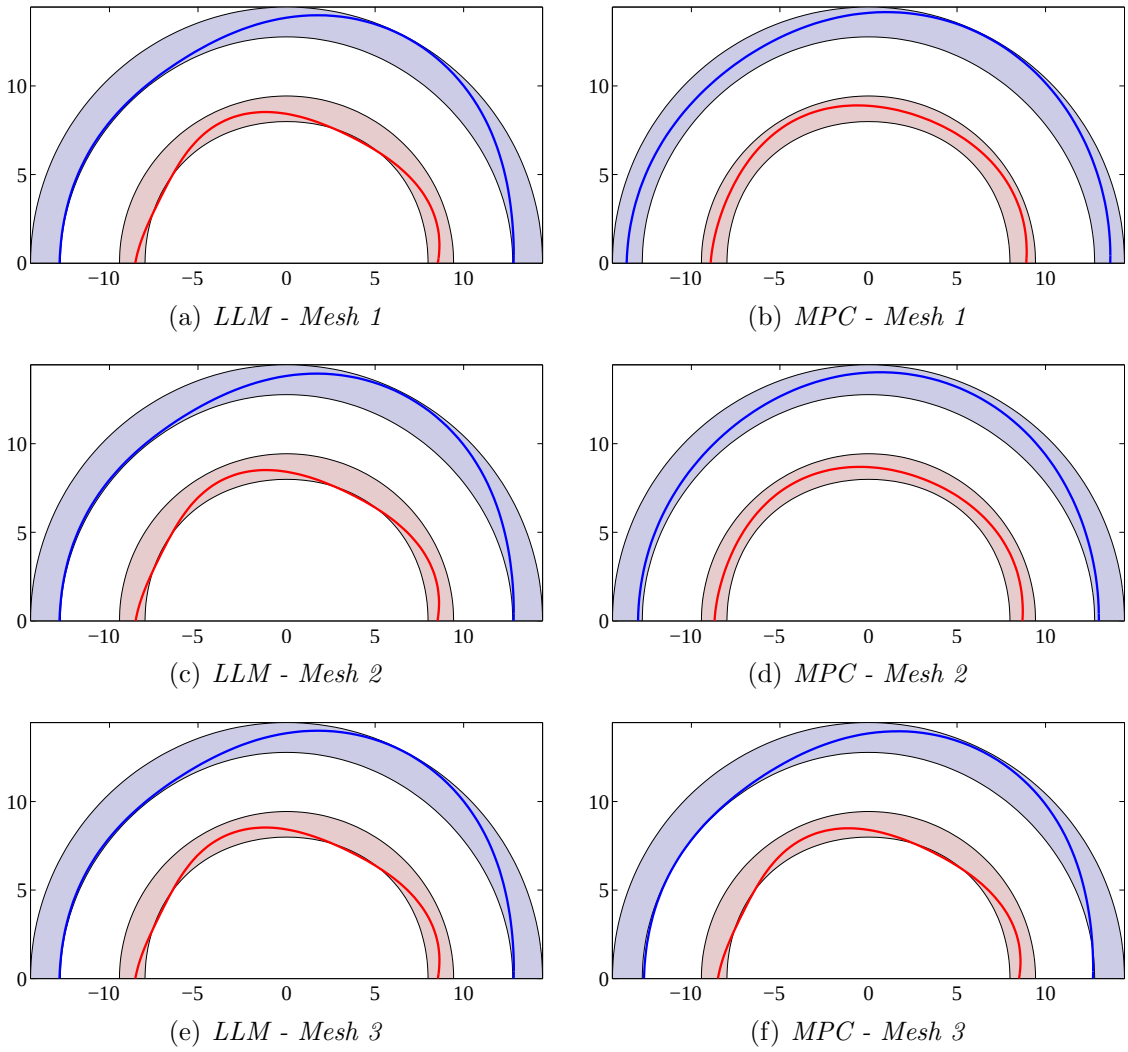
(c) *LLM - Mesh 2* (d) *MPC - Mesh 2*

(e) *LLM - Mesh 3* (f) *MPC - Mesh 3*

Figure 5.19: *Wave velocities for all three meshes, where the materials are tilted 26.565°. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

for LLM, for all tilt angles. On the other hand, mesh 2 is closer to the LLM P-wave plot when the propagation direction is perpendicular to the tilt angle, especially for larger tilt angles as seen in figure 5.20(d).

The LLM solutions seem, as the ones for $\beta = 45°$, realistic. The P-wave velocity touches the Voigt bound and the S-wave velocity touches the Reuss bound when the propagation direction is equal to the tilt angle. When the propagation direction is perpendicular to the tilt angle, both wave velocities touch the Reuss bound. Figure 5.21 shows the wave velocities for mesh 1, using LLM for all tilt angles, in polar plots. Displaying the wave plots in polar plots makes it easier to see that the velocities actually touch the boundaries at the correct angles. It can be seen that the P-wave



(a) *LLM - Mesh 1* (b) *MPC - Mesh 1*

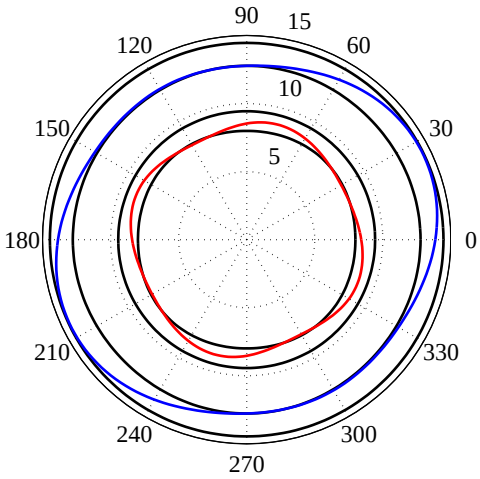
(c) *LLM - Mesh 2* (d) *MPC - Mesh 2*

(e) *LLM - Mesh 3* (f) *MPC - Mesh 3*

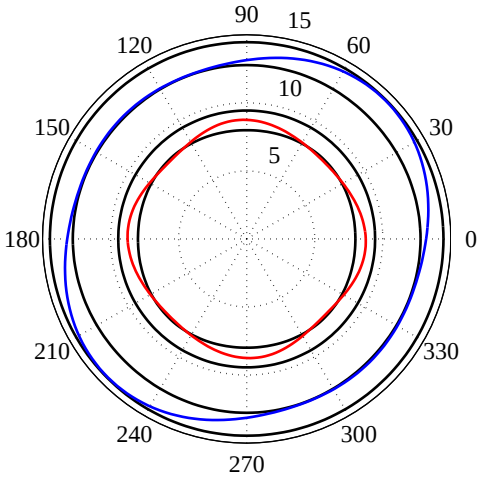
Figure 5.20: *Wave velocities for all three meshes, where the materials are tilted 63.435°. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

velocity is equal to the Voigt wave velocity at a propagation angle very close to 26°, 45° and 63° in these three plots.

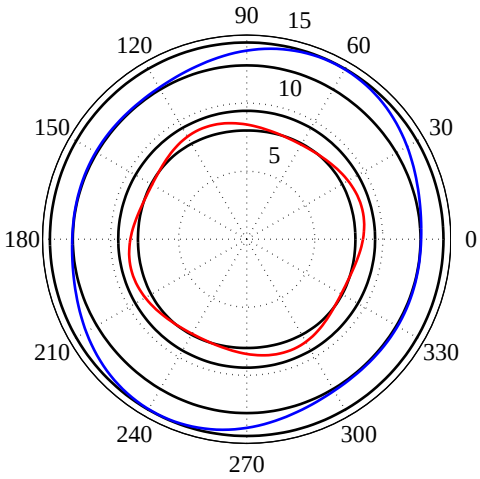
The S-wave velocity is for some propagation angles a small fraction slower than the Reuss wave velocity, as seen in figure 5.21. This is a very small deviation and can be due to the influence of the end layers of the RVE.



(a) *LLM - Mesh 1 - $\beta = 26.565°$*



(b) *LLM - Mesh 1 - $\beta = 45°$*



(c) *LLM - Mesh 1 - $\beta = 63.435°$*

Figure 5.21: *Polar plots for mesh 1 for all tilt angles using LLM, shown for propagation angles $\theta = [0, 2\pi]$. The blue and red line is V_P and V_S, respectively, and the black lines are the Voigt and Reuss bounds.*

5.5 Randomly Scattered Materials

Consider an RVE where the materials are randomly distributed, and not arranged in layers. Let material A and B have the properties

$$E_A = 100, \qquad \nu_A = 0.2,$$
$$E_B = 300, \qquad \nu_B = 0.1,$$

as before. Figure 5.22 shows an RVE with matching boundary nodes and nine elements, where five of them are material B and the remaining four is material A. Looking at the boundaries that are going to be coupled, the materials are matching some places and non-matching other places. That is, some boundary parts are material A-A or B-B in the coupling, others are non-matching with A-B connections.

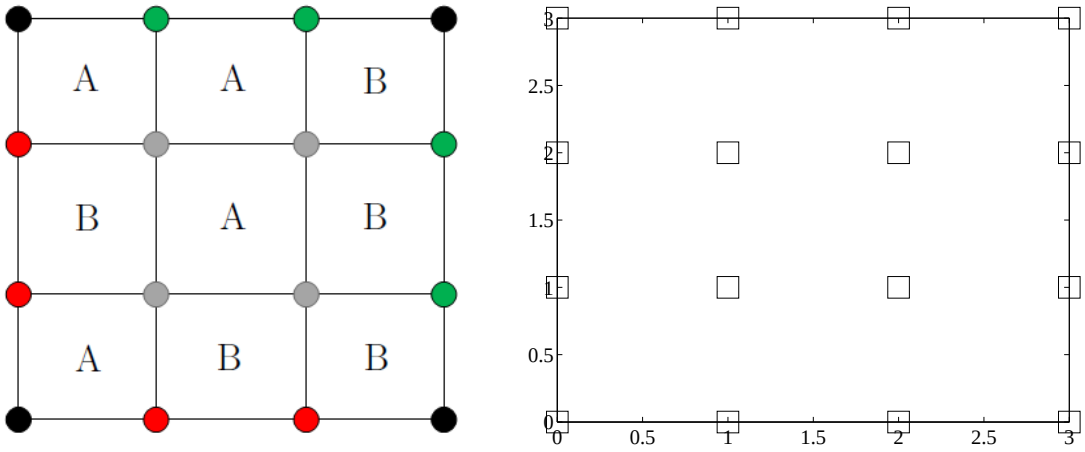
The RVE is analysed with LLM and MPC and the results become

$$\tilde{\mathbf{L}}_{LLM}^{Match} = \begin{bmatrix} 192.4093 & 32.1845 & -3.5525 \\ 32.1845 & 207.8698 & -3.1624 \\ -3.5525 & -3.1624 & 78.7497 \end{bmatrix}$$

and

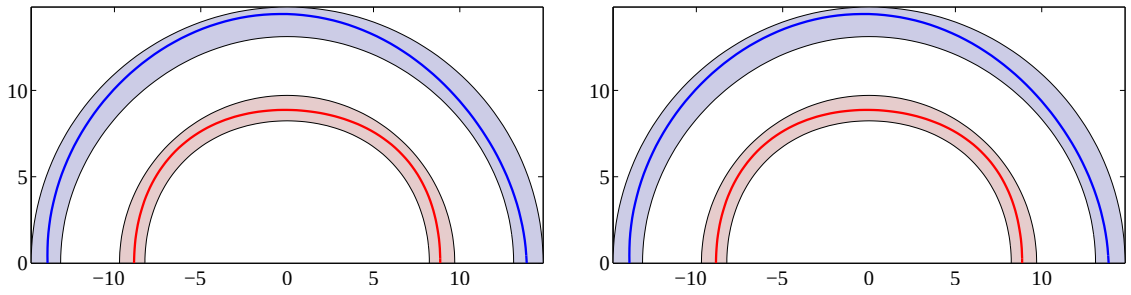
$$\tilde{\mathbf{L}}_{MPC}^{Match} = \begin{bmatrix} 192.4093 & 32.1845 & -3.5525 \\ 32.1845 & 207.8698 & -3.1624 \\ -3.5525 & -3.1624 & 78.7497 \end{bmatrix}.$$

The matrices are identical to each other, which is also seen form the wave velocity plots in figure 5.23. The fact that the materials are non-matching some places along



(a) *Randomly scattered materials* (b) *Grid in MATLAB*

Figure 5.22: *RVE with matching grids, where the materials A and B are irregularly distributed.*



(a) *LLM - Matching grids* (b) *MPC - Matching grids*

Figure 5.23: *Wave velocities for randomly distributed materials with matching grids. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*

the boundaries does not seem to influence the MPC result, as it did in the previous section.

Assume that the boundary nodes are relocated, leading to the RVE given in figure 5.24. The boundary nodes are now non-matching, but the elements consist of the same materials as before. The outcome using both LLM and MPC is

$$\tilde{\mathbf{L}}_{LLM}^{Non-match} = \begin{bmatrix} 184.3955 & 31.9305 & -1.2395 \\ 31.9305 & 193.4649 & 1.8788 \\ -1.2395 & 1.8788 & 73.5664 \end{bmatrix}$$

and

$$\tilde{\mathbf{L}}_{MPC}^{Non-match} = \begin{bmatrix} 141.0476 & 23.8653 & -10.8865 \\ 23.8653 & 192.9717 & -5.7153 \\ -10.8865 & -5.7153 & 67.2550 \end{bmatrix},$$

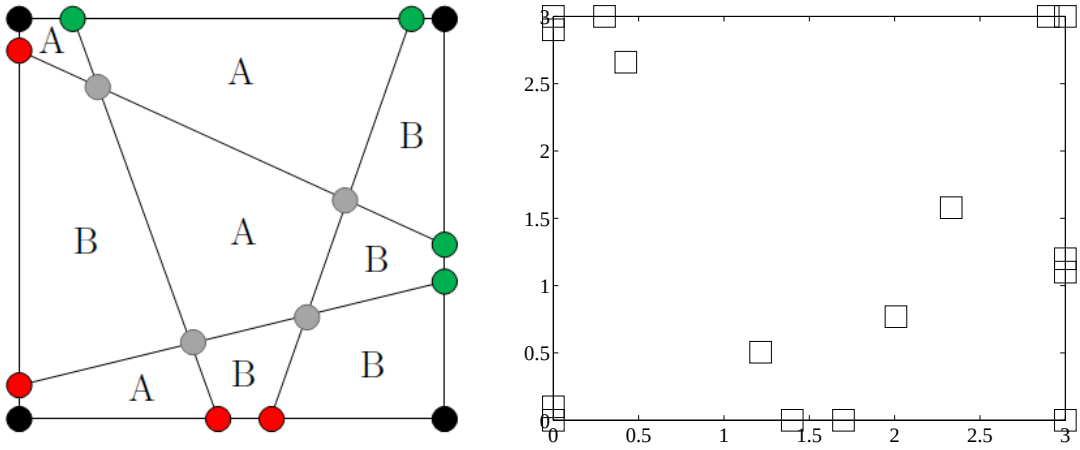
and the corresponding wave velocity plots can be seen in figure 5.25.

The LLM solution is within the Voigt and Reuss bounds by a large margin. This LLM matrix is not similar to the LLM matrix from the previous analysis where the nodes were matching, because the volume fraction and the location of the materials in the RVE are changed. The frame node locations for the two couplings are depicted in figure 5.26.

The P-wave velocity plot for the MPC matrix is way off for some propagation angles, and is not a reliable solution for this RVE. The relative difference in Frobenius norm between the LLM and MPC matrix is

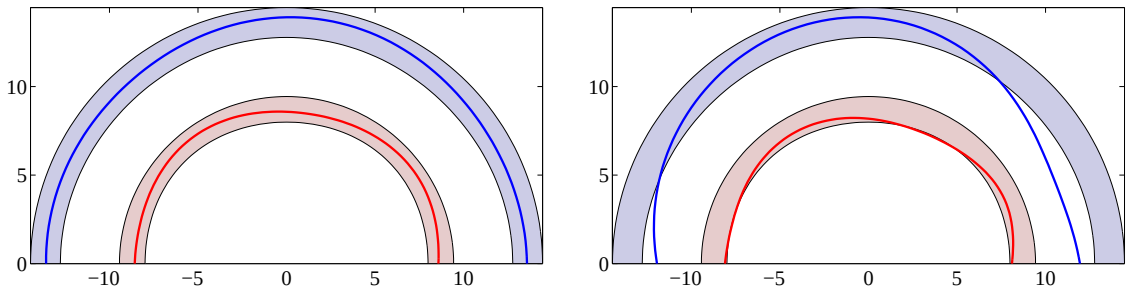
$$\frac{\|\tilde{\mathbf{L}}_{MPC}^{Non-match} - \tilde{\mathbf{L}}_{LLM}^{Non-match}\|_F}{\|\tilde{\mathbf{L}}_{LLM}^{Non-match}\|_F} = 0.17262.$$

A difference in the homogenized constitutive matrix of this magnitude can affect the final result of the two-scale homogenization problem substantially, if the matrix is used as input in the macro analysis.



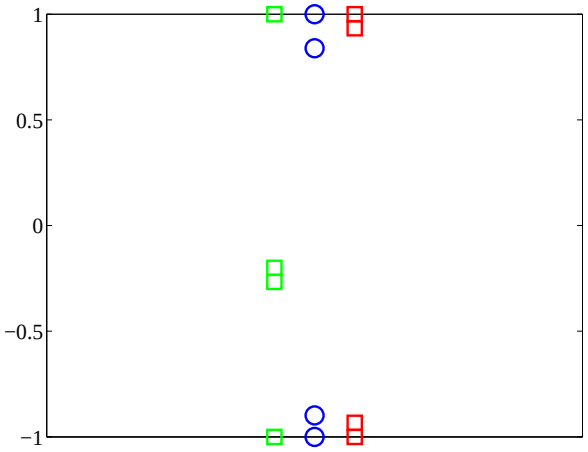
(a) *Randomly scattered material* (b) *Grid in MATLAB*

Figure 5.24: *RVE with non-matching grids, where the materials A and B are irregularly distributed.*

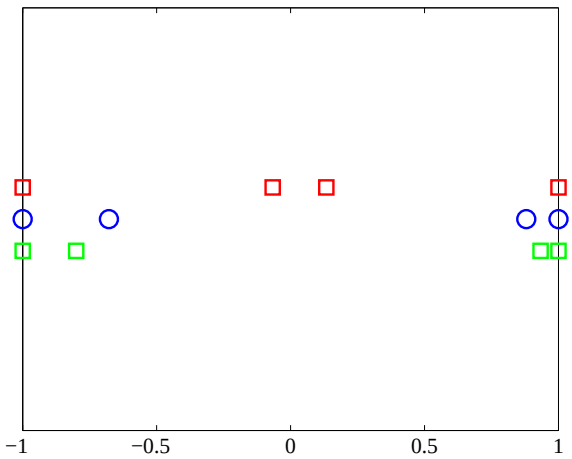


(a) *LLM - Non-matching grids* (b) *MPC - Non-matching grids*

Figure 5.25: *Wave velocities for randomly distributed materials with non-matching grids. The blue line is V_P whereas the red line is V_S. The blue and red shaded areas are the Voigt and Reuss bounds for V_P and V_S, respectively.*



(a) *Frame nodes between* Γ_1 *and* Γ_2



(b) *Frame nodes between* Γ_3 *and* Γ_4

Figure 5.26: *Frame nodes for the RVE with non-matching grids. The green squares indicate the boundary 1 and 3 nodes, and the red squares indicate the boundary 2 and 4 nodes, whereas the blue circles mark the frame nodes.*

Chapter 6

Concluding Remarks

It is highly important that the information from micro scale is preserved and transferred to the macro scale in a proper manner. This means in practice that the homogenized constitutive matrix for the RVE has to be correctly calculated. Several numerical analysis have been carried out, demonstrating that the LLM method produces the correct answer for RVEs with non-matching grids, as well as for matching grids. This is a major advantage in multi-scale homogenization compared to using MPC, which gives the correct solution only when the grids are matching.

In section 5.4, an RVE with three different meshes is analysed. The boundary compositions are different for each mesh, while the tilt angle of the layers and the size of the RVE are held constant. LLM gave practically the same result regardless of which mesh that was considered, which is pretty demonstrative concerning the reliability of the method. One out of the three meshes was correctly calculated by MPC, which was the mesh with matching grids and matching materials along the boundary.

The frame node configuration does not necessarily need to be all possible frame nodes. For a homogeneous RVE, the frame node configuration is insignificant for the result, even for non-matching grids. The zero moment rule calculates the location of the frame nodes based on the multipliers when they create a constant stress state. When an RVE is homogeneous, the coupled boundaries are in mutual equilibrium with constant stress. So the moment distribution along the coupling is the same as when calculating the frame nodes and hence the frame nodes are not needed in the calculation to guarantee constant stress in the coupling. For an RVE consisting of heterogeneous material, the stresses along the boundaries are not constant, and hence all frame nodes are needed. However, if the material is perfectly layered and the homogeneous layers are divided into smaller layers with the same properties, only the end frame nodes of the homogeneous layers are sufficient.

Chapter 7

Further Work

The code should be tested on actual industrial problems with realistic material properties as input. The macro scale analysis should also be included to see how much the results on macro scale are affected when different inputs from micro scale is used and to get the whole picture of the problem. Other elements than four-node quadrilateral elements, for instance triangular elements, can with advantage be implemented in the code to easier describe the geometry inside the RVEs.

A computer program solving two-scale computational homogenization using LLM in 3D should be implemented. Both mathematical homogenization and the LLM method are directly transferable to 3D, except how to find the frame node locations. Hence a zero moment rule for a 2D interface frame is an important step to make homogenization possible in 3D.

Bibliography

[1] G. E. Backus. Long-Wave Elastic Anisotropy Produced by Horizontal Layering. *Journal of Geophysical Research.* 1962;67(11):4427-4440.

[2] A. Bensoussan, J. L. Lions, G. Papanicolaou. *Asymptotic Analysis for Periodic Structures.* The Netherlands, North-Holland Publishing Company, 1978.

[3] K. Børset. *Multiscale Modelling of Elastic Parameters.* Master's thesis. Norwegian University of Science and Technology, 2008.

[4] R. D. Cook, D. S. Malkus, M. E. Plesha, R. J. Witt. *Concepts and Applications of Finite Element Analysis.* 4th ed. University of Wisconsin - Madison, John Wiley & Sons, Inc., 2002.

[5] C. A. Felippa, K. C. Park, M. R. Ross. A Classification of Interface Treatments for FSI.

[6] C. A. Felippa, K. C. Park, M. R. Ross. Recent Work on LLM and PFEM For Fluid-Structure Interaction. *Presentation at DFG FSI Workshop.* Munich, Germany, 2009.

[7] J. Fish. Short-course Lecture Notes. *Presentation at the Intensive Short-course and International Seminar on Multiscale Computational Science and Engineering.* Norwegian University of Science and Technology, Department of Structural Engineering, Trondheim, Norway, 2007.

[8] M. G. D. Geers, V. G. Kouznetsova, W. A. M. Brekelmans. Multi-scale modelling: Computational homogenization in solid mechanics. Eindhoven University of Technology.

[9] R. Hill. The elastic behaviour of a crystalline aggregate. *Proceedings of the Physical Society A.* 1952;65:349-354.

[10] L. T. Ikelle, L. Amundsen. *Introduction to Petroleum Seismology.* Society of Exploration Geophysics, Tulsa, Oklahoma, USA, 2005.

[11] V. G. Kouznetsova. *Computational homogenization for the multi-scale analysis of multi-phase materials.* PhD thesis. Technische Universiteit Eindhoven, 2002.

[12] K. C. Park, C. A. Felippa, R. Ohayon. Partitioned formulation of internal fluid-structure interaction problems by localized lagrange multipliers. *Comput. Methods Appl. Mech. Engrg.* 2001;190:2989-3007.

[13] K. C. Park, C. A. Felippa, G. Rebel. Interfacing Nonmatching FEM Meshes: The Zero Moment Rule. 2001.

[14] S. B. Raknes. *Methods for handling computational homogenization with non-matching grids.* Pre-master's project. Norwegian University of Science and Technology, 2010.

[15] R. L. Robertson. Determining Residual Stress from Boundary Measurements: A Linearized Approach. *Journal of Elasticity.* 1998;52:63-73.

[16] M. R. Ross. *Coupling and Simulation of Acoustic Fluid-Structure Interaction Systems Using Localized Lagrange Multipliers.* PhD thesis. University of Colorado at Boulder, 2006. Available at: http://www.colorado.edu/engineering/CAS/courses.d/FSI.d/FSI.CISM.d/Ross.Thesis.pdf

[17] M. R. Ross, M. R. Sprague, C. A. Felippa, K. C. Park. Treatment of acoustic fluidstructure interaction by localized Lagrange multipliers and comparison to alternative interface-coupling methods. 2008.

[18] M. R. Ross, C. A. Felippa, K. C. Park, M. R. Sprague. Treatment of acoustic fluidstructure interaction by localized Lagrange multipliers: Formulation. *Computer Methods in Applied Mechanics and Engineering.* 2008;197:3057-3079. DOI:10.1016/j.cma.2008.02.017. (accessed 26 February 2008).

[19] K. Spildo. *Computational homogenization for linear solid mechanics problems.* Master's thesis. Norwegian University of Science and Technology, 2009.

[20] Z. Yuan, J. Fish. Toward realization of computational homogenization in practice. *International Journal for Numerical Methods in Engineering.* 2008;73:361-380. DOI: 10.1002/nme.2074. (accessed 2 May 2007).

[21] W. Zijl, M. A. N. Henriks, C. M. P. Hart. Numerical Homogenization of the Rigidity Tensor in Hookes Law Using the Node-Based Finite Element Method. *Mathematical Geology.* 2002;34:3.

Appendix A

Backus Averaging in 2D

The Backus averaging technique is given in 3D in a paper by Backus [1]. The following derivation is in 2D and based on the derivation presented by Backus.

Let x_1, x_2, x_3 be the position coordinates and s_1, s_2, s_3 the components of displacements in 3D. The elastic properties of the medium are constant in x_1 and x_2 but may vary with x_3. Let $w(x_3)$ be any continuous weighting function that averages over a length l'. Then

$$\langle f \rangle(x_3) = \int_{-\infty}^{\infty} w(\zeta - x_3) f(\zeta)\, \mathrm{d}\zeta$$

is the average of f over a distance roughly l' around the position x_3. The average will be written $\langle f \rangle$. One approximation is done in the following derivation: if $f(x_3)$ is nearly constant when x_3 changes by no more than l', while $g(x_3)$ may vary much over the same distance, then

$$\langle fg \rangle = f \langle g \rangle. \tag{A.1}$$

Assume a medium that is transversely isotropic for each x_3 with a vertical axis of symmetry. The stress-strain relations in 2D can be found by setting $\varepsilon_{22} = \varepsilon_{12} = \varepsilon_{23} = 0$ in the 3D equations, i.e. plane strain in the x_2-direction. The stress-strain relations become

$$\begin{aligned} T_{11} &= a\frac{\partial s_1}{\partial x_1} + f\frac{\partial s_3}{\partial x_3}, \\ T_{33} &= f\frac{\partial s_1}{\partial x_1} + c\frac{\partial s_3}{\partial x_3}, \\ T_{13} &= l\left(\frac{\partial s_1}{\partial x_3} + \frac{\partial s_3}{\partial x_1}\right), \end{aligned} \tag{A.2}$$

where a, f, c and l are the elastic parameters varying over x_3. T_{22} is also non-zero, but not important in this derivation.

Consider an infinite horizontal slab of vertical thickness L' $\gg$ l'. The slab is divided into horizontal layers so thin that when averaged over a vertical distance l', all properties of the slab are almost independent of x_3. If the slab is subjected to the static stress T_{13} and T_{33} on the top and bottom, the derivatives $\dfrac{\partial s_i}{\partial x_3}$ and T_{11} will vary widely because of the different properties of the layers, whereas the derivatives $\dfrac{\partial s_i}{\partial x_1}$ and the applied stress components will vary slowly. Solving for the rapidly varying stress and displacement field variables from equation (A.2) yield

$$\begin{aligned}
\frac{\partial s_1}{\partial x_3} &= \frac{1}{l}T_{13} - \frac{\partial s_3}{\partial x_1}, \\
\frac{\partial s_3}{\partial x_3} &= \frac{1}{c}T_{33} - \frac{f}{c}\frac{\partial s_1}{\partial x_1}, \\
T_{11} &= \left(a - \frac{f^2}{c}\right)\frac{\partial s_1}{\partial x_1} + \frac{f}{c}T_{33}.
\end{aligned} \tag{A.3}$$

The rapid variations with x_3 in the left-hand side field variables are produced by the rapid variations of the elastic coefficients. Since there are no products of a rapid varying field variable and a rapid varying elastic parameter on the right-hand side in (A.3), formula (A.1) can be applied to compute the averages;

$$\begin{aligned}
\frac{\partial \langle s_1\rangle}{\partial x_3} &= \left\langle \frac{1}{l} \right\rangle \langle T_{13}\rangle - \frac{\partial \langle s_3\rangle}{\partial x_1}, \\
\frac{\partial \langle s_3\rangle}{\partial x_3} &= \left\langle \frac{1}{c} \right\rangle \langle T_{33}\rangle - \left\langle \frac{f}{c} \right\rangle \frac{\partial \langle s_1\rangle}{\partial x_1}, \\
\langle T_{11}\rangle &= \left\langle a - \frac{f^2}{c} \right\rangle \frac{\partial \langle s_1\rangle}{\partial x_1} + \left\langle \frac{f}{c} \right\rangle \langle T_{33}\rangle.
\end{aligned}$$

The averaged stresses can now be solved for, resulting in

$$\begin{aligned}
\langle T_{11}\rangle &= A\frac{\partial \langle s_1\rangle}{\partial x_1} + F\frac{\partial \langle s_3\rangle}{\partial x_3}, \\
\langle T_{33}\rangle &= F\frac{\partial \langle s_1\rangle}{\partial x_1} + C\frac{\partial \langle s_3\rangle}{\partial x_3}, \\
\langle T_{13}\rangle &= L\left(\frac{\partial \langle s_1\rangle}{\partial x_3} + \frac{\partial \langle s_3\rangle}{\partial x_1}\right),
\end{aligned}$$

where

$$
\begin{aligned}
A &= \left\langle a - \frac{f^2}{c} \right\rangle + \left\langle \frac{1}{c} \right\rangle^{-1} \left\langle \frac{f}{c} \right\rangle^2 \\
C &= \left\langle \frac{1}{c} \right\rangle^{-1} \\
F &= \left\langle \frac{1}{c} \right\rangle^{-1} \left\langle \frac{f}{c} \right\rangle \\
L &= \left\langle \frac{1}{l} \right\rangle^{-1}
\end{aligned}
$$

are the relations between the averaged stresses and the strains calculated from averaged displacements, and thus the components of the constitutive matrix;

$$
\begin{Bmatrix} \langle T_{11} \rangle \\ \langle T_{33} \rangle \\ \langle T_{13} \rangle \end{Bmatrix} = \begin{bmatrix} A & F & 0 \\ F & C & 0 \\ 0 & 0 & L \end{bmatrix} \begin{Bmatrix} \dfrac{\partial \langle s_1 \rangle}{\partial x_1} \\ \dfrac{\partial \langle s_3 \rangle}{\partial x_3} \\ 2\dfrac{\partial \langle s_1 \rangle}{\partial x_3} \end{Bmatrix}.
$$

Appendix B

The MATLAB Code

The MATLAB code for solving two-scale linear homogenization problems is given in the following sections. The mesh generators have some limitations that are important to know when using the program. The node coordinates can be punched in by hand if another mesh than the generators can produce is preferable, as long as they create square elements. The limitations for the mesh generator are;

- The RVE has to be square with 90° angle in each corner
- The number of boundary nodes have to be equal on opposite boundaries
- Straight lines are drawn between opposite boundary nodes, and internal nodes are created where the lines cross.

Figure B.1 illustrates how the mesh generator builds up the RVE.

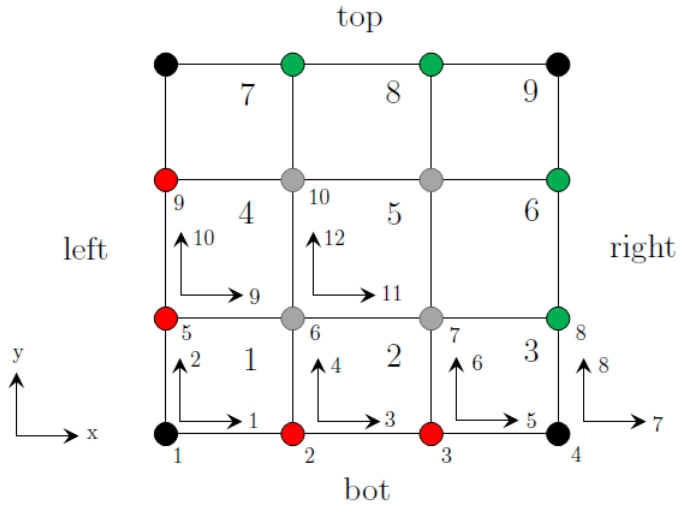


Figure B.1: *Illustration of how the elements, nodes and DOFs are counted in the RVEs and the macro problem in the code.*

The RVE problem inputs that need to be punched in are

bot, top: The x-coordinates of the bottom and top boundary nodes, respectively, including corner nodes. The vectors have to have the same length.

left, right: The y-coordinates of the left and right boundary nodes, respectively, including corner nodes. The vectors have to have the same length.

E, nu: Young's modulus and Poisson's ratio for each element. The first element in the vector is the property of element 1, and the second for element 2, etc. They have to have the same length as number of elements.

th: Thickness of the RVE. Default is 1 unit.

ElemType: Type of element. Q4 is the only option.

Frame_Nodes: Choice of frame node configuration. Alternatives: *all; internal; everyother; end.*

PlaneStrain: If *yes*, plane strain is used, if *no*, plane stress is used.

WavePlots: Wave velocity plots are drawn together with the Voigt and Reuss averages. Alternatives: *yes; no.*

Theta: The upper limit for the propagation angle in the wave velocity plots. The angle starts at 0 and stops at *Theta.*

The inputs for the macro problem are

BOT, TOP, LEFT, RIGHT: The same as for the RVE problem.

BCmacro(DOF numbers) = 0: Boundary conditions. The DOF numbers of the DOFs that should be constrained must be written in the brackets.

rmacro(DOF numbers) = value: Loads. The value of the load, and the DOF numbers of the DOFs the load should be applied to, must be given.

B.1 Input for RVEs in This Thesis

```
%% Input to Main_script_LLM.m and Main_script_MPC.m

%5.1.1 Homogeneous Material with Matching Grids
bot=  [0 2];        %x-nodes of bottom side
top=  [0 2];        %x-nodes of top side
left= [0 1 2];      %y-nodes of left side
right=[0 1 2];      %y-nodes of right side
[XYZ CON DOF] = mesh(bot,top,left,right);
E=ones(size(CON,1),1)*200000;   %Young's modulus [El1 El2 ...]
nu=ones(size(CON,1),1)*0.3;     %Poisson's ratio [El1 El2 ...]

%5.1.2 Layered Materials with Matching Grids
bot=  [0 2];
top=  [0 2];
left= [0 1 2 3 4 5 6 7 8 9 10];
right=[0 1 2 3 4 5 6 7 8 9 10];
[XYZ CON DOF] = mesh(bot,top,left,right);
E=[100 1000 10 1 0.01 1000 0.1 10 100 1];
nu=[0.45 0.405 0.36 0.315 0.27 0.225 0.18 0.135 0.09 0.045];

%5.1.3 Homogeneous Material with Non-matching Grids
delta1 = 0.1;
delta2 = 0.1;
bot=  [0 0.5 1 1.5 2];
top=  [0 0.5 1 1.5 2];
left= [0 1+delta2 2];
right=[0 1-delta1 2];
[XYZ CON DOF] = mesh(bot,top,left,right);
E=ones(size(CON,1),1)*200000;
nu=ones(size(CON,1),1)*0.3;

%5.1.4 Non-matching Grids Inside Homogeneous Layers
bot=  [0 1 2];
top=  [0 1 2];
left= [0 1 2 3 4 4.5 5 5.9 6 7 8 9 10];
right=[0 1 2 3 4 4.9 5 5.5 6 7 8 9 10];
[XYZ CON DOF] = mesh(bot,top,left,right);
E=[100 100 1000 1000 10 10 1 1 0.01 0.01 0.01 0.01 1000 1000 1000 ...
    1000  0.1 0.1 10 10 100 100 1 1];
nu=[0.45 0.45 0.405 0.405 0.36 0.36 0.315 0.315 0.27 0.27 0.27 ...
    0.27 0.225 0.225 0.225 0.225 0.18 0.18 0.135 0.135 0.09 0.09 ...
    0.045 0.045];

%5.3 Flat Layered Materials
bot=  [0 2];
top=  [0 2];
left= [0 1 2];
right=[0 1 2];
```

```
[XYZ CON DOF] = mesh(bot,top,left,right);
E=[100 300];
nu=[0.2 0.1];

%5.4 Tilted Layered Materials
beta=45;            %Tilt Angle: -90<beta<90, beta~=0
ElemMesh = '34,1';  %(Hight,mesh x) Alt: 2,1/34,1/34,2/34,3

bot = [0 0.5/tand(abs(beta)) 1/tand(abs(beta)) 1.5/tand(abs(beta)) ...
    2/tand(abs(beta))];
top = bot;
if strcmpi(ElemMesh,'2,1')
%2 layered RVE
    left= [0 0.001 2];
    right=[0 1.999 2];
%2 x 34 units RVE
elseif strcmpi(ElemMesh,'34,1')
    %Mesh 1
    left= [0 0.001 2.01 4.01 6.01 8.01 10.01 12.01 14.01 16.01 ...
        18.01 20.01 22.01 24.01 26.01 28.01 30.01 32.01 34];
    right=[0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 33.999 34];
elseif strcmpi(ElemMesh,'34,2')
    %Mesh 2
    left= [0 0.001 4 8 12 16 20 24 28 32 34];
    right=[0 2 6 10 14 18 22 26 30 33.999 34];
elseif strcmpi(ElemMesh,'34,3')
    %Mesh 3
    left= [0 0.001 1.01 2.01 3.01 4.01 5.01 6.01 7.01 8.01 9.01 ...
        10.01 11.01 12.01 13.01 14.01 15.01 16.01 17.01 18.01 ...
        19.01 20.01 21.01 22.01 23.01 24.01 25.01 26.01 27.01 ...
        28.01 29.01 30.01 31.01 32.01 34];
    right=[0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 ...
        22 23 24 25 26 27 28 29 30 31 32 33 33.99 34];
end
if sign(beta)==-1   %if negative beta
    rightx=right;
    right=left; left=rightx;
end
[XYZ CON DOF] = mesh(bot,top,left,right);
E=ones(size(CON,1),1)*100;
nu=ones(size(CON,1),1)*0.2;
Eb1=5:8:size(CON,1); Eb2=6:8:size(CON,1);
Eb3=7:8:size(CON,1); Eb4=8:8:size(CON,1);
E(Eb1)=300; nu(Eb1)=0.1; E(Eb2)=300; nu(Eb2)=0.1;
E(Eb3)=300; nu(Eb3)=0.1; E(Eb4)=300; nu(Eb4)=0.1;

%Reference Solution 2 layers - 400 elements, 45 deg
bot= [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 ...
    1.6 1.7 1.8 1.9 2];
top=  bot;
left= bot;
right=bot;
```

```
[XYZ CON DOF] = mesh(bot,top,left,right);
E=ones(size(CON,1),1)*100;
nu=ones(size(CON,1),1)*0.2;
Eb=[1:20 23:40 43:60 65:80 85:100 107:120 127:140 149:160 169:180 ...
    191:200 211:220 233:240 253:260 275:280 295:300 317:320 ...
    337:340 359:360 379:380];
E(Eb)= 300;
nu(Eb)=0.1;

%5.5 Randomly Scattered Material
%Matching grids
    bot=  [0 1 2 3];
    top=  [0 1 2 3];
    left= [0 1 2 3];
    right=[0 1 2 3];
%Non-matching grids
    % bot=  [0 1.4 1.7 3];
    % top=  [0 0.3 2.9 3];
    % left= [0 0.1 2.9 3];
    % right=[0 1.1 1.2 3];
[XYZ CON DOF] = mesh(bot,top,left,right);
E= [100 300 300    300 100 300     100 100 300];
nu=[0.2 0.1 0.1    0.1 0.2 0.1     0.2 0.2 0.1];
```

B.2 Mesh Generators

```
function [XYZ CON DOF] = mesh(bot,top,left,right)

nnodesx=length(bot);        %Number of horizontal nodes
nnodesy=length(left);       %Number of vertical nodes
nelx=nnodesx-1;             %Number of horizontal element
nely=nnodesy-1;             %Number of vertical element
nnodes=nnodesx*nnodesy;     %Number of nodes

lx=bot(nnodesx)-bot(1);
ly=left(nnodesy)-left(1);

XYZ=zeros(nnodes,2);        %Node coordinates
for i=1:nnodesy
    yl=left(i)-left(1);
    dy=right(i)-left(i);
    for j=1:nnodesx
        xb=bot(j)-bot(1);
        dx=top(j)-bot(j);

        x=(dx*yl+xb*ly)/(ly-dx*dy/lx);
        y=dy/lx*x+yl;

```

```
23          XYZ(j+(i-1)*nnodesx,1) = x + bot(1);
24          XYZ(j+(i-1)*nnodesx,2) = y + left(1);
25      end
26  end
27  nel=nelx*nely;                  %Number of elements
28  CON=zeros(nel,4);               %Node numbers for each element
29  for i=1:nely
30      for j=1:nelx
31          CON(j+(i-1)*nelx,:)=[j+(i-1)*nnodesx j+(i-1)*nnodesx+1 ...
                j+i*nnodesx+1 j+i*nnodesx];
32      end
33  end
34  DOF=zeros(nel,2*4);             %DOFs for each element
35  for i=1:nel
36      DOF(i,:)=[CON(i,1)*2-1 CON(i,1)*2 CON(i,2)*2-1 CON(i,2)*2 ...
            CON(i,3)*2-1 CON(i,3)*2 CON(i,4)*2-1 CON(i,4)*2];
37  end
38  figure(1)
39  plot(XYZ(:,1),XYZ(:,2),'sk')
40  title('RVE mesh')
41  end
```

```
1   function [CON DOF] = mesh_givenXYZ(XYZ,nelx,nely)
2
3   nnodesx=nelx+1;             %Number of horizontal nodes
4   nel=nelx*nely;              %Number of elements
5
6   CON=zeros(nel,4);           %Node numbers for each element
7   for i=1:nely
8       for j=1:nelx
9           CON(j+(i-1)*nelx,:)=[j+(i-1)*nnodesx j+(i-1)*nnodesx+1 ...
                j+i*nnodesx+1 j+i*nnodesx];
10      end
11  end
12  DOF=zeros(nel,2*4);         %DOFs for each element
13  for i=1:nel
14      DOF(i,:)=[CON(i,1)*2-1 CON(i,1)*2 CON(i,2)*2-1 CON(i,2)*2 ...
            CON(i,3)*2-1 CON(i,3)*2 CON(i,4)*2-1 CON(i,4)*2];
15  end
16  figure(1)
17  plot(XYZ(:,1),XYZ(:,2),'sk')
18  title('RVE mesh')
19  end
```

```
1   function [XYZ CON DOF] = macro_mesh(BOT,TOP,LEFT,RIGHT)
2
3   nnodesx=length(BOT);
4   nnodesy=length(LEFT);
5   nelx=nnodesx-1;
```

```
nely=nnodesy-1;
nnodes=nnodesx*nnodesy;

lx=BOT(nnodesx)-BOT(1);
ly=LEFT(nnodesy)-LEFT(1);

XYZ=zeros(nnodes,2);
for i=1:nnodesy
    yl=LEFT(i)-LEFT(1);
    dy=RIGHT(i)-LEFT(i);
    for j=1:nnodesx
        xb=BOT(j)-BOT(1);
        dx=TOP(j)-BOT(j);

        x=(dx*yl+xb*ly)/(ly-dx*dy/lx);
        y=dy/lx*x+yl;

        XYZ(j+(i-1)*nnodesx,1) = x + BOT(1);
        XYZ(j+(i-1)*nnodesx,2) = y + LEFT(1);
    end
end
nel=nelx*nely;
CON=zeros(nel,4);
for i=1:nely
    for j=1:nelx
        CON(j+(i-1)*nelx,:)=[j+(i-1)*nnodesx j+(i-1)*nnodesx+1 ...
            j+i*nnodesx+1 j+i*nnodesx];
    end
end
DOF=zeros(nel,2*4);
for i=1:nel
    DOF(i,:)=[CON(i,1)*2-1 CON(i,1)*2 CON(i,2)*2-1 CON(i,2)*2 ...
        CON(i,3)*2-1 CON(i,3)*2 CON(i,4)*2-1 CON(i,4)*2];
end
figure(2)
plot(XYZ(:,1),XYZ(:,2),'ok')
title('Macro problem mesh')
end
```

```
function [CON DOF] = macro_mesh_givenXYZ(XYZ,nelx,nely)

nnodesx=nelx+1;
nel=nelx*nely;
CON=zeros(nel,4);
for i=1:nely
    for j=1:nelx
        CON(j+(i-1)*nelx,:)=[j+(i-1)*nnodesx j+(i-1)*nnodesx+1 ...
            j+i*nnodesx+1 j+i*nnodesx];
    end
end
DOF=zeros(nel,2*4);
```

```
for i=1:nel
    DOF(i,:)=[CON(i,1)*2-1 CON(i,1)*2 CON(i,2)*2-1 CON(i,2)*2 ...
        CON(i,3)*2-1 CON(i,3)*2 CON(i,4)*2-1 CON(i,4)*2];
end
figure(2)
plot(XYZ(:,1),XYZ(:,2),'ok')
title('Macro problem mesh')
end
```

B.3 Main Script Using LLM

```
clear all
clc
%% INPUT MICRO

%5.1.1 Homogeneous Material with Matching Grids
bot=  [0 2];         %x-nodes of bottom side
top=  [0 2];         %x-nodes of top side
left= [0 1 2];       %y-nodes of left side
right=[0 1 2];       %y-nodes of right side
[XYZ CON DOF] = mesh(bot,top,left,right);
E=ones(size(CON,1),1)*200000;    %Young's modulus [El1 El2 ...]
nu=ones(size(CON,1),1)*0.3;      %Poisson's ratio [El1 El2 ...]

th=1;                %Thickness
ElemType='Q4';       %Element type

%Choice of frame node configuration
    Frame_Nodes = 'all';
    %Frame_Nodes = 'internal';
    %Frame_Nodes = 'everyother';
    %Frame_Nodes = 'end';

PlaneStrain='yes';  %Plane strain: yes / Plane stress: no

WavePlots = 'no';    %Want wave velocity plots: yes / no
Theta = pi;          %Max propagation angle

%% INPUT MACRO

BOT=  [0 2 4];
TOP=  [0 2 4];
LEFT= [0 2 4];
RIGHT=[0 2 4];

[XYZma CONma DOFma] = macro_mesh(BOT,TOP,LEFT,RIGHT);
%[CONma DOFma] = macro_mesh_givenXYZ(XYZ,nelx,nely);

```

```
38 BCmacro = ones(2*length(XYZma),1);
39 rmacro = zeros(2*length(XYZma),1);
40
41 %Boundary constraints; give constrained DOFs
42 BCmacro([1 7 8 13])=0;
43
44 %Load; give value and DOFs that the value are applied to
45 rmacro([5 17]) = 5;
46 rmacro(11) = 10;
47
48 BCmacro=find(BCmacro);
49 rmacro=rmacro(BCmacro);
50
51 %% MICRO ANALYSIS
52
53 if strcmpi(PlaneStrain,'yes')
54     [D] = plane_strain(E,nu);
55 else
56     [D] = plane_stress(E,nu);
57 end
58
59 [BCcorner] = BCcorner(XYZ);   %All DOFs except corner DOFs
60
61 [K] = K_matrix(XYZ,CON,DOF,D,ElemType,th,BCcorner);
62
63 gamma=1;
64 [frameNatcoord12] = zmr(XYZ,gamma,Frame_Nodes);
65 [L1] = L_matrix(XYZ,frameNatcoord12,gamma);
66 [B1] = B_matrix(XYZ,gamma,BCcorner);
67 gamma=2;
68 [L2] = L_matrix(XYZ,frameNatcoord12,gamma);
69 [B2] = B_matrix(XYZ,gamma,BCcorner);
70 gamma=3;
71 [frameNatcoord34] = zmr(XYZ,gamma,Frame_Nodes);
72 [L3] = L_matrix(XYZ,frameNatcoord34,gamma);
73 [B3] = B_matrix(XYZ,gamma,BCcorner);
74 gamma=4;
75 [L4] = L_matrix(XYZ,frameNatcoord34,gamma);
76 [B4] = B_matrix(XYZ,gamma,BCcorner);
77
78 [A] = A_matrix(K,B1,B2,B3,B4,L1,L2,L3,L4);
79
80 ndofs=2*length(XYZ);        %Number of DOFs
81 HomoL=zeros(3,3);
82 for m=1:3
83     eps=zeros(3,1);
84     eps(m,1)=1;
85
86     [r] = r_vector(XYZ,CON,DOF,D,ElemType,th,eps,BCcorner);
87
88     b=zeros(length(A),1);
89     b(1:length(K))=r;
```

```
90
91      x=A\b;
92
93      d=zeros(ndofs,1);
94      d(BCcorner)=x(1:length(K));      %Total displacement vector
95
96      [sigmaHomo] = sigmaHomo_vector(XYZ,CON,DOF,D,eps,d);
97      HomoL(:,m)=sigmaHomo;
98
99      figure(4+m)
100     plot(XYZ(:,1),XYZ(:,2),'sk'); hold on
101     plot(XYZ(:,1)+d(1:2:length(d)-1),XYZ(:,2)+d(2:2:length(d)),'or')
102     title(['Displacment plot for unit strain state ', num2str(m)])
103 end
104 display('The homogenized constitutive matrix is')
105 HomoLllm=HomoL
106
107 if strcmpi(WavePlots,'yes')      %If wave velocity plots
108     theta=0:pi/100:Theta;
109     rho=1;
110
111     VoigtReuss      %Runs VoigtReuss script
112
113     [speeds] = christoffelspeeds(HomoLllm,theta,rho);
114
115     figure(8)
116     polar(theta',speeds(:,1),'b'); hold on;
117     title('P-wave velocity polar plot')
118
119     figure(9)
120     polar(theta',speeds(:,2),'r'); hold on;
121     title('S-wave velocity polar plot')
122
123     [xp,yp] = pol2cart(theta,speeds(:,1)');
124     [xs,ys] = pol2cart(theta,speeds(:,2)');
125     figure(10);
126     plot(xp,yp,'b'); hold on
127     plot(xs,ys,'r'); hold on
128     title('Wave velocity plots')
129 end
130 %% MACRO ANALYSIS
131
132 [Kmacro] = macro_K_matrix(XYZma,CONma,DOFma,HomoL,th,BCmacro);
133
134 u=Kmacro\rmacro;
135
136 dmacro=zeros(2*length(XYZma),1);
137 dmacro(BCmacro) = u;    %Total macro displacement vector
138
139 figure(11)
140 plot(XYZma(:,1),XYZma(:,2),'ok');hold on
141 plot(XYZma(:,1)+dmacro(1:2:length(dmacro)-1), ...
```

```
142     XYZma(:,2)+dmacro(2:2:length(dmacro)),'or')
143 title('Macro scale displacement plot')
```

B.3.1 plane_strain.m

```
1  function [D] = plane_strain(E,nu)
2
3  D=zeros(length(E)*3,3);
4  for i=1:length(E)
5      Ee=E(i);         %Element Young's modulus
6      nue=nu(i);       %Element Poisson's ratio
7
8      D(i*3-2:i*3,1:3) = Ee/((1-2*nue)*(1+nue))*[1-nue nue 0;nue ...
           1-nue 0;0 0 (1-2*nue)/2];
9  end
```

B.3.2 plane_stress.m

```
1  function [D] = plane_stress(E,nu)
2
3  D=zeros(length(E)*3,3);
4  for i=1:length(E)
5      Ee=E(i);         %Element Young's modulus
6      nue=nu(i);       %Element Poisson's ratio
7
8      D(i*3-2:i*3,1:3) = Ee/(1-nue^2)*[1 nue 0;nue 1 0;0 0 (1-nue)/2];
9  end
```

B.3.3 BCcorner.m

```
1  function [BCcorner] = BCcorner(XYZ)
2
3  ndof=2*length(XYZ);
4  dofNumb(1:ndof)=1:ndof;
5
6  id1=find(abs(XYZ(:,1)-max(XYZ(:,1)))<10^-10);
7  id2=find(abs(XYZ(:,1)-min(XYZ(:,1)))<10^-10);
8  nb1nodes=length(id1); nb2nodes=length(id2);
9
10 idcorner(1)=id2(1);          %Find corner DOFs
11 idcorner(2)=id1(1);
```

```
idcorner(3)=id2(nb2nodes);
idcorner(4)=id1(nb1nodes);

dofNumb(idcorner*2)=0;       %Make corner DOFs zero
dofNumb(idcorner*2-1)=0;

BCcorner=find(dofNumb);      %All DOFs except corner DOFs
```

B.3.4 K_matrix.m

```
function [K] = K_matrix(XYZ,CON,DOF,D,ElemType,th,BCcorner)

nel=size(CON,1);             %Number of elements
ndof=2*length(XYZ);          %Number of DOFs

K=zeros(ndof,ndof);
for i=1:nel
    id=DOF(i,:);             %Element DOF numers
    xyz=XYZ(CON(i,:),:);     %Element node coordinates
    De=D(i*3-2:i*3,1:3);     %Constitutive element matrix

    [ke]=feval(ElemType,xyz,De,th);

    K(id,id)=K(id,id) + ke;
end
K=K(BCcorner,BCcorner);
end
```

ElemType - Q4.m

```
function [ke re] = Q4(xyz,De,th,eps)

ke=zeros(8,8);       %Element stiffness matrix
re=zeros(8,1);       %Element load vector

a=1/sqrt(3);
w=1;
Gauss=[-a a a -a;-a -a a a];  %4-point Gauss

for i = 1:4
    xi=Gauss(1,i);
    eta=Gauss(2,i);

    [dsB J]=dispstrain_B(xyz,xi,eta);

```

```
    if nargin==3
    ke=ke+dsB'*De*dsB*J*th*w;
    end

    if nargin==4
    re=re+dsB'*De*eps*J*th*w;
    end
end
end
```

dispstrain_B.m

```
function [dsB J] = dispstrain_B(xyz,xi,eta)

natcoord=[-1 1 1 -1;-1 -1 1 1];     %Natural coordinates

dNdnat(1,:)=1/4*natcoord(1,:).*(1+natcoord(2,:)*eta);
dNdnat(2,:)=1/4*natcoord(2,:).*(1+natcoord(1,:)*xi);

Jmat=dNdnat*xyz;        %Forming the Jacobian matrix
J=det(Jmat);            %Calculate the Jacobian

dNdx=Jmat\dNdnat;

dsB=zeros(3,8);         %Create displacement-strain matrix
dsB(1,1:2:7)=dNdx(1,:);
dsB(2,2:2:8)=dNdx(2,:);
dsB(3,1:2:7)=dNdx(2,:);
dsB(3,2:2:8)=dNdx(1,:);
end
```

B.3.5 zmr.m

```
function [frameNatcoord] = zmr(XYZ,gamma,Frame_Nodes)

if gamma==1
    id1=find(abs(XYZ(:,1)-max(XYZ(:,1)))<10^-10);%Boundary 1 node nr
    id2=find(abs(XYZ(:,1)-min(XYZ(:,1)))<10^-10);%Boundary 2 node nr
    xy1=XYZ(id1,2); xy2=XYZ(id2,2);           %Boundary node y-coord
else
    id1=find(abs(XYZ(:,2)-max(XYZ(:,2)))<10^-10);%Boundary 3 node nr
    id2=find(abs(XYZ(:,2)-min(XYZ(:,2)))<10^-10);%Boundary 4 node nr
    xy1=XYZ(id1,1); xy2=XYZ(id2,1);           %Boundary node x-coord
end

```

```
nb1nodes=length(id1); nb2nodes=length(id2);%Number of boundary nodes
Lb=xy1(nb1nodes)-xy1(1);                    %Length of boundary
lambdaNatcoord1=2*(xy1-min(xy1))./Lb - 1;%Boundary 1 nat node coord
lambdaNatcoord2=2*(xy2-min(xy2))./Lb - 1;%Boundary 2 nat node coord

%If just end nodes
if (nb1nodes==2 && nb2nodes==2) || strcmpi(Frame_Nodes,'end')
    frameNatcoord=[-1,1];
    if gamma==1
        figure(3)
        plot(zeros(length(frameNatcoord),1),frameNatcoord,'ob');
        hold on
        set(gca,'xtick',[])
        plot(ones(nb1nodes,1)*-0.15,lambdaNatcoord1,'sg');hold on
        plot(ones(nb2nodes,1)*0.15,lambdaNatcoord2,'sr')
        xlim([-1 1])
        title('Frame nodes for coupling 1-2')
    else
        figure(4)
        plot(frameNatcoord,zeros(length(frameNatcoord),1),'ob');
        hold on
        set(gca,'ytick',[])
        plot(lambdaNatcoord1,ones(nb1nodes,1)*-0.15,'sg');hold on
        plot(lambdaNatcoord2,ones(nb2nodes,1)*0.15,'sr');
        ylim([-1 1])
        title('Frame nodes for coupling 3-4')
    end
    return
end

[force1 force2] = zmr_force(lambdaNatcoord1,lambdaNatcoord2);

b1=lambdaNatcoord1; b2=lambdaNatcoord2;  %Shortening of name only
b=zeros(nb1nodes+nb2nodes-4,1);   %Internal nodes from both RVEs
force=b;              %Internal node multipliers from both RVEs

%Arranging the internal nodes in the order from low to high
%in b and the multipliers in force
i=2;j=2;
for m=1:length(b)
    if b1(i) < b2(j)
        b(m)=b1(i);
        force(m)=force1(i);
        i=i+1;
    else
        b(m)=b2(j);
        force(m)=force2(j);
        j=j+1;
    end
end
%Starting at 3rd node and calculate the moments at this and
%the former node.
```

```
n=2;
for k=2:length(b)
    xi1=b(k-1);                          %Former node coord
    xi2=b(k);                            %Node coord
    id1=find((xi1-b)>0.0001);    %Find the array number for...
    id2=find((xi2-b)>0.0001);        %...the positive moment arms
    %Moments
    M1=(force1(1)+force2(1))*(xi1+1)+force(id1)'*(xi1-b(id1));
    M2=(force1(1)+force2(1))*(xi2+1)+force(id2)'*(xi2-b(id2));

    if xi1==xi2                  %If matching nodes
        frameNatcoord(n)=xi1;
        n=n+1;
    elseif sign(M1) == sign(M2)
    elseif abs(M1)<10^-10 || abs(M2)<10^-10
    else                         %If different signs on moments
        frameNatcoord(n)=xi1+abs(M1)/(abs(M1)+abs(M2))*(xi2-xi1);
        n=n+1;
    end
end
frameNatcoord(1)=-1;                                %First node
frameNatcoord(length(frameNatcoord)+1)=1;           %Last node

if strcmpi(Frame_Nodes,'internal')
    frameNatcoord=frameNatcoord(2:length(frameNatcoord)-1);
elseif strcmpi(Frame_Nodes,'everyother')
    frameNatcoord_test=frameNatcoord(1:2:length(frameNatcoord))
    if frameNatcoord_test(length(frameNatcoord_test))~=1
        frameNatcoord_test2=frameNatcoord(1:2:length(frameNatcoord)/2);
        frameNatcoord_test2(length(frameNatcoord_test2)+1:length ...
            (frameNatcoord_test2)*2)=fliplr(frameNatcoord(length ...
            (frameNatcoord):-2:length(frameNatcoord)/2+1));
        frameNatcoord=frameNatcoord_test2;
    else
        frameNatcoord=frameNatcoord_test;
    end

end
if gamma==1
    figure(3)
    plot(zeros(length(frameNatcoord),1),frameNatcoord,'ob');hold on
    set(gca,'xtick',[])
    plot(ones(nb1nodes,1)*-0.15,lambdaNatcoord1,'sg');hold on
    plot(ones(nb2nodes,1)*0.15,lambdaNatcoord2,'sr')
    xlim([-1 1])
    title('Frame nodes for coupling 1-2')
else
    figure(4)
    plot(frameNatcoord,zeros(length(frameNatcoord),1),'ob');hold on
    set(gca,'ytick',[])
    plot(lambdaNatcoord1,ones(nb1nodes,1)*-0.15,'sg');hold on
    plot(lambdaNatcoord2,ones(nb2nodes,1)*0.15,'sr');
```

```
117     ylim([-1 1])
118     title('Frame nodes for coupling 3-4')
119 end
120 end
```

zmr_force.m

```
1  function [force1 force2] = zmr_force(lambdaNatcoord1,lambdaNatcoord2)
2
3  b1=lambdaNatcoord1; b2=lambdaNatcoord2;  %Shortening of name only
4  nb1nodes=length(b1); nb2nodes=length(b2);%Number of boundary noes
5
6  force1=zeros(nb1nodes,1);        %Boundary 1 multipliers
7  force2=zeros(nb2nodes,1);        %Voundary 2 multipliers
8
9  for i=2:nb1nodes
10     l=b1(i)-b1(i-1);             %length of boundary element
11     f=1/2*l;           %Force = Area of triangular shape function
12
13     force1(i-1)=force1(i-1) - f;%Negative forces for boundary 1
14     force1(i)=force1(i) - f;
15 end
16 for i=2:nb2nodes
17     l=b2(i)-b2(i-1);
18     f=1/2*l;
19
20     force2(i-1)=force2(i-1) + f;%Positive forces for boundary 2
21     force2(i)=force2(i) + f;
22 end
23 end
```

B.3.6 L_matrix.m

```
1  function [L] = L_matrix(XYZ,frameNatcoord,gamma)
2
3  if gamma==1
4      var=max(XYZ(:,1));
5      id=find(abs(XYZ(:,1)-var)<10^-10);  %Boundary node numbers
6      xy=XYZ(id,2);                       %Boundary node coordinates
7  elseif gamma==2
8      var=min(XYZ(:,1));
9      id=find(abs(XYZ(:,1)-var)<10^-10);
10     xy=XYZ(id,2);
11 elseif gamma==3
12     var=max(XYZ(:,2));
```

```
    id=find(abs(XYZ(:,2)-var)<10^-10);
    xy=XYZ(id,1);
else
    var=min(XYZ(:,2));
    id=find(abs(XYZ(:,2)-var)<10^-10);
    xy=XYZ(id,1);
end

nbnodes=length(id);                          %Number of boundary nodes
nfnodes=length(frameNatcoord);               %Number of frame nodes
Lb=xy(nbnodes)-xy(1);                        %Length of boundary
lambdaNatcoord=2*(xy-min(xy))./Lb - 1;       %Boundary natural node coord

L=zeros(2*nbnodes,2*nfnodes);
for i=1:nbnodes
    for j=1:nfnodes
        xi=lambdaNatcoord(i);

        [Nb]=shapefunc_frame(j,xi,frameNatcoord);

        L(2*i,2*j)=Nb;
        L(2*i-1,2*j-1)=Nb;
    end
end
end
```

shapefunc_frame.m

```
function [Nb] = shapefunc_frame(j,xi,framenatcoord)

a=framenatcoord;    %Shortening of name only
b=length(a);        %Number of frame nodes

if b==1
    error('Must have at least TWO frame nodes')
end

if j==1                     % If bottom frame node
    if xi<=a(2)                 %Upper triangle
        l=a(1)-a(2);            %Length of element
        Nb=xi/l+a(2)/abs(l);    %Value at xi
    else                    % If outside shape function
        Nb=0;
    end
elseif j==b                 % If top frame node
    if xi>=a(b-1)               %Lower triangle
        l=a(b)-a(b-1);
        Nb=xi/l-a(b-1)/abs(l);
    else
```

```
22          Nb=0;
23      end
24  else                        % If middle frame node(s)
25      if xi<=a(j) && xi>=a(j-1)     %Lower triangle
26          l=a(j)-a(j-1);
27          Nb=xi/l-a(j-1)/abs(l);
28      elseif xi>a(j) && xi<=a(j+1) %Upper triangle
29          l=a(j)-a(j+1);
30          Nb=xi/l+a(j+1)/abs(l);
31      else
32          Nb=0;
33      end
34  end
```

B.3.7 B_matrix.m

```
1  function [B] = B_matrix(XYZ,gamma,BCcorner)
2
3  if gamma==1
4      var=max(XYZ(:,1));
5      id=find(abs(XYZ(:,1)-var)<10^-10);  %Boundary node numbers
6  elseif gamma==2
7      var=min(XYZ(:,1));
8      id=find(abs(XYZ(:,1)-var)<10^-10);
9  elseif gamma==3
10     var=max(XYZ(:,2));
11     id=find(abs(XYZ(:,2)-var)<10^-10);
12 else
13     var=min(XYZ(:,2));
14     id=find(abs(XYZ(:,2)-var)<10^-10);
15 end
16 nbnodes=length(id);                    %Number of boundary nodes
17 nnodes=length(XYZ);                    %Number of nodes
18
19 B=zeros(2*nbnodes,2*nnodes);
20
21 for i=1:nbnodes
22     B(2*i,2*id(i))=1;
23     B(2*i-1,2*id(i)-1)=1;
24 end
25 B=B(:,BCcorner);
26 end
```

B.3.8 A_matrix.m

```
function [A] = A_matrix(K,B1,B2,B3,B4,L1,L2,L3,L4)

Ki=size(K,1);
B1i=size(B1,1);
B2i=size(B2,1);
B3i=size(B3,1);
B4i=size(B4,1);
L1j=size(L1,2);
L2j=size(L2,2);
L3j=size(L3,2);
L4j=size(L4,2);

i=Ki+B1i+B2i+B3i+B4i+L1j+L3j;
A=zeros(i,i);

A(1:Ki,1:Ki) = K;
A(Ki+1:Ki+B1i,1:Ki) = B1;
A(Ki+B1i+1:Ki+B1i+B2i,1:Ki) = B2;
A(Ki+B1i+B2i+1:Ki+B1i+B2i+B3i,1:Ki) = B3;
A(Ki+B1i+B2i+B3i+1:Ki+B1i+B2i+B3i+B4i,1:Ki) = B4;

A(1:Ki,Ki+1:Ki+B1i) = B1';
A(1:Ki,Ki+B1i+1:Ki+B1i+B2i) = B2';
A(1:Ki,Ki+B1i+B2i+1:Ki+B1i+B2i+B3i) = B3';
A(1:Ki,Ki+B1i+B2i+B3i+1:Ki+B1i+B2i+B3i+B4i) = B4';

A(i-L3j-L1j+1:i-L3j,Ki+1:Ki+B1i) = -L1';
A(i-L3j-L1j+1:i-L3j,Ki+B1i+1:Ki+B1i+B2i) = -L2';
A(i-L3j+1:i,Ki+B1i+B2i+1:Ki+B1i+B2i+B3i) = -L3';
A(i-L3j+1:i,Ki+B1i+B2i+B3i+1:Ki+B1i+B2i+B3i+B4i) = -L4';

A(Ki+1:Ki+B1i,i-L3j-L1j+1:i-L3j) = -L1;
A(Ki+B1i+1:Ki+B1i+B2i,i-L3j-L1j+1:i-L3j) = -L2;
A(Ki+B1i+B2i+1:Ki+B1i+B2i+B3i,i-L3j+1:i) = -L3;
A(Ki+B1i+B2i+B3i+1:Ki+B1i+B2i+B3i+B4i,i-L3j+1:i) = -L4;

for k=Ki+1:i            %No zeros along the diagonal
   A(k,k)=10^-18;
end
end
```

B.3.9 r_vector.m

```
function [r] = r_vector(XYZ,CON,DOF,D,ElemType,th,eps,BCcorner)

nel=size(CON,1);              %Number of elements
ndof=2*length(XYZ);           %Number of DOFs

r=zeros(ndof,1);
for i=1:nel
    id=DOF(i,:);              %Element DOF numers
    xyz=XYZ(CON(i,:),:);      %Element node coordinates
    De=D(i*3-2:i*3,1:3);      %Constitutive element matrix

    [~, re]=feval(ElemType,xyz,De,th,eps);

    r(id)=r(id) + re;
end
r=-r(BCcorner);
end
```

B.3.10 sigmaHomo_vector.m

```
function [sigmaHomo] = sigmaHomo_vector(XYZ,CON,DOF,D,eps,d)

nel=size(CON,1);              %Number of elements

a=1/sqrt(3);
w=1;
Gauss=[-a a a -a;-a -a a a];%4-point Gauss

sigma=zeros(3,1);             %Stress vector
vol=0;                        %Volume
for i=1:nel
    De=D(i*3-2:i*3,1:3);      %Constitutive element matrix
    id=DOF(i,:);              %Element DOF numers
    xyz=XYZ(CON(i,:),:);      %Element node coordinates
    de=d(id);                 %Element node displacements

    for j = 1:4
        xi=Gauss(1,j);
        eta=Gauss(2,j);

        [dsB J]=dispstrain_B(xyz,xi,eta);

        sigmaGauss=De*(dsB*de+eps);   %Stress in Gauss-point j in ...
            element i

```

```
25          sigma=sigma + sigmaGauss*J*w; %Add the stresses in all ...
               Gauss-points
26
27          vol=vol+J;
28      end
29  end
30  sigmaHomo=sigma./vol; %Divide the summed stresses by total volume
31  end
```

B.3.11 VoigtReuss.m

```
1   %% Calculates Voigt and Reuss bounds
2
3   nel=size(CON,1);              %Number of elements
4   v=zeros(3,3);                 %Voigt matrix
5   r=zeros(3,3);                 %Reuss matrix
6   vol=0;                        %Total volume
7   Gauss=[-1 1 1 -1;-1 -1 1 1]./sqrt(3); %4 point Gauss
8   for i=1:nel
9       De=D(i*3-2:i*3,1:3);      %Constitutive element matrix
10      xyz=XYZ(CON(i,:),:);      %Element node coordinates
11
12      vole=0;                   %Element volume
13      for j = 1:4
14          xi=Gauss(1,j);
15          eta=Gauss(2,j);
16          [~, J]=dispstrain_B(xyz,xi,eta);
17          vole=vole+J;
18      end
19
20      v=v+De*vole;              %Voigt sum
21      r=r+vole./De;             %Reuss sum
22
23      vol=vol+vole;             %Total volume
24  end
25
26  Lv=v./vol;                    %Voigt average matrix
27  Lr=1./r*vol;                  %Reuss average matrix
28
29  [speedsv] = christoffelspeeds(Lv,theta,rho); %Voigt velocities
30  [speedsr] = christoffelspeeds(Lr,theta,rho); %Reuss velocities
31
32  figure(8)             %Polar P-wave plots
33  polar(theta',speedsv(:,1),'k'); hold on;
34  polar(theta',speedsr(:,1),'k');
35
36  figure(9)             %Polar S-wave plots
37  polar(theta',speedsv(:,2),'k'); hold on;
38  polar(theta',speedsr(:,2),'k');
```

```
39
40 [xvp,yvp] = pol2cart(theta,speedsv(:,1)');
41 [xvs,yvs] = pol2cart(theta,speedsv(:,2)');
42 [xrp,yrp] = pol2cart(theta,speedsr(:,1)');
43 [xrs,yrs] = pol2cart(theta,speedsr(:,2)');
44
45 figure(10)            %Plots shaded bounds
46 area(xvp,yvp,'FaceColor',[0.8 0.8 0.9]);hold on
47 area(xrp,yrp,'FaceColor',[1 1 1]); hold on
48 area(xvs,yvs,'FaceColor',[0.9 0.8 0.8]); hold on
49 area(xrs,yrs,'FaceColor',[1 1 1]);hold on
50 axis image
```

christoffelspeeds.m

```
1  function [speeds] = christoffelspeeds(L,theta,rho)
2
3  speeds=zeros(length(theta),2);
4  for i=1:length(theta)
5      [x,y] = pol2cart(theta(i),1);
6
7      D=[x 0 y;
8         0 y x];
9      D=D/norm([x y]);       %Make them unit vectors
10
11     eigvals = eig(D*L*D');
12     eigvals = sort(eigvals,'descend');  %Eigenvalues
13
14     speeds(i,:) = sqrt(eigvals./rho);   %Wave velocities
15 end
16 end
```

B.3.12 macro_K_matrix.m

```
1  function [K] = macro_K_matrix(XYZma,CONma,DOFma,HomoL,th,BCmacro)
2
3  nel=size(CONma,1);         %Number of RVEs
4  ndof=2*length(XYZma);      %Number of DOFs
5
6  K=zeros(ndof,ndof);
7  for i=1:nel
8      id=DOFma(i,:);            %RVE DOF numers
9      xyz=XYZma(CONma(i,:),:); %RVE node coordinates
10
11     a=1/sqrt(3);
```

```
12      w=1;
13      Gauss=[-a a a -a;-a -a a a];        %4-point Gauss
14
15      ke=zeros(8,8);
16      for j = 1:4
17          xi=Gauss(1,j);
18          eta=Gauss(2,j);
19
20          [dsB J]=dispstrain_B(xyz,xi,eta);
21
22          ke=ke+dsB'*HomoL*dsB*J*th*w;
23      end
24
25      K(id,id)=K(id,id) + ke;
26  end
27  K=K(BCmacro,BCmacro);
28  end
```

B.4 Main Script Using MPC

```
1   clear all
2   clc
3   %% INPUT MICRO
4
5   %5.1.1 Homogeneous Material with Matching Grids
6   bot=  [0 2];        %x-nodes of bottom side
7   top=  [0 2];        %x-nodes of top side
8   left= [0 1 2];      %y-nodes of left side
9   right=[0 1 2];      %y-nodes of right side
10  [XYZ CON DOF] = mesh(bot,top,left,right);
11  E=ones(size(CON,1),1)*200000;   %Young's modulus [El1 El2 ...]
12  nu=ones(size(CON,1),1)*0.3;     %Poisson's ratio [El1 El2 ...]
13
14  th=1;               %Thickness
15  ElemType='Q4';      %Element type
16
17  PlaneStrain='yes';  %Plane strain: yes / Plane stress: no
18
19  WavePlots = 'no';   %Want wave velocity plots: yes / no
20  Theta = pi;         %Max propagation angle
21
22  %% INPUT MACRO
23
24  BOT=  [0 2 4];
25  TOP=  [0 2 4];
26  LEFT= [0 2 4];
27  RIGHT=[0 2 4];
28
```

```
29 [XYZma CONma DOFma] = macro_mesh(BOT,TOP,LEFT,RIGHT);
30 %[CONma DOFma] = macro_mesh_givenXYZ(XYZ,nelx,nely);
31
32 BCmacro = ones(2*length(XYZma),1);
33 rmacro = zeros(2*length(XYZma),1);
34
35 %Boundary constraints; give constrained DOFs
36 BCmacro([1 7 8 13])=0;
37
38 %Load; give value and DOFs that the value are applied to
39 rmacro([5 17]) = 5;
40 rmacro(11) = 10;
41
42 BCmacro=find(BCmacro);
43 rmacro=rmacro(BCmacro);
44
45 %% MICRO ANALYSIS
46
47 if strcmpi(PlaneStrain,'yes')
48     [D] = plane_strain(E,nu);
49 else
50     [D] = plane_stress(E,nu);
51 end
52
53 [BCcorner] = BCcorner(XYZ);   %All DOFs except corner DOFs
54
55 [T] = T_matrix(XYZ,BCcorner);
56
57 [K] = K_matrix(XYZ,CON,DOF,D,ElemType,th,BCcorner);
58
59 Km=T'*K*T;                    %Modify K
60
61 ndofs=2*length(XYZ);          %Number of DOFs
62 HomoL=zeros(3,3);
63 for m=1:3
64     eps=zeros(3,1);
65     eps(m,1)=1;
66
67     [r] = r_vector(XYZ,CON,DOF,D,ElemType,th,eps,BCcorner);
68
69     rm=T'*r;                  %Modify r
70
71     dm=Km\rm;
72
73     d=zeros(ndofs,1);
74     dms=T*dm;
75     d(BCcorner)=dms;          %Total displacement vector
76
77     [sigmaHomo] = sigmaHomo_vector(XYZ,CON,DOF,D,eps,d);
78     HomoL(:,m)=sigmaHomo;
79
80     figure(2+m)
```

```
81      plot(XYZ(:,1),XYZ(:,2),'sk'); hold on
82      plot(XYZ(:,1)+d(1:2:length(d)-1),XYZ(:,2)+d(2:2:length(d)),'or')
83      title(['Displacment plot for unit strain state ', num2str(m)])
84  end
85  display('The homogenized constitutive matrix is')
86  HomoLmpc=HomoL
87
88  if strcmpi(WavePlots,'yes')      %If wave velocity plots
89      theta=0:pi/100:Theta;
90      rho=1;
91
92      VoigtReuss       %Runs VoigtReuss script
93
94      [speeds] = christoffelspeeds(HomoLmpc,theta,rho);
95
96      figure(8)
97      polar(theta',speeds(:,1),'b'); hold on;
98      title('P-wave velocity polar plot')
99
100     figure(9)
101     polar(theta',speeds(:,2),'r'); hold on;
102     title('S-wave velocity polar plot')
103
104     [xp,yp] = pol2cart(theta,speeds(:,1)');
105     [xs,ys] = pol2cart(theta,speeds(:,2)');
106     figure(10);
107     plot(xp,yp,'b'); hold on
108     plot(xs,ys,'r'); hold on
109     title('Wave velocity plots')
110 end
111
112 %% MACRO ANALYSIS
113
114 [Kmacro] = macro_K_matrix(XYZma,CONma,DOFma,HomoL,th,BCmacro);
115
116 u=Kmacro\rmacro;
117
118 dmacro=zeros(2*length(XYZma),1);
119 dmacro(BCmacro) = u;     %Total macro displacement vector
120
121 figure(11)
122 plot(XYZma(:,1),XYZma(:,2),'ok');hold on
123 plot(XYZma(:,1)+dmacro(1:2:length(dmacro)-1), ...
124     XYZma(:,2)+dmacro(2:2:length(dmacro)),'or')
125 title('Macro scale displacement plot')
```

B.4.1 T_matrix.m

```
function [T] = T_matrix(XYZ,BCcorner)

nnodes=length(XYZ);                                 %Number of nodes

id1=find(abs(XYZ(:,1)-max(XYZ(:,1)))<10^-10);    %Boundary 1 node nr
id2=find(abs(XYZ(:,1)-min(XYZ(:,1)))<10^-10);    %Boundary 2 node nr
id3=find(abs(XYZ(:,2)-max(XYZ(:,2)))<10^-10);    %Boundary 3 node nr
id4=find(abs(XYZ(:,2)-min(XYZ(:,2)))<10^-10);    %Boundary 4 node nr

ids2=id2(2:length(id2)-1);                  %Boundary 2 slave node nr
ids4=id4(2:length(id4)-1);                  %Boundary 4 slave node nr

ns2nodes=length(ids2);              %Number of boundary 2 slave nodes
ns4nodes=length(ids4);              %Number of boundary 4 slave nodes
nsnodes=ns2nodes+ns4nodes;          %Number of slave nodes

slavedofs=zeros(2*nsnodes,1);               %Slave DOFs
slavedofs(1:2:2*nsnodes-1)=[2*ids4'-1 2*ids2'-1];
slavedofs(2:2:2*nsnodes)=[2*ids4' 2*ids2'];

a=ones(nnodes,1);
a(id2)=0; a(id4)=0;         %Taking away slave nodes+3 corner nodes
a(length(a))=0;             %Taking away upper right corner node
idm=find(a);                %Master nodes(including internal nodes)

nmnodes=length(idm);                        %Number of master nodes
masterdofs=zeros(2*nmnodes,1);              %Master DOFs
masterdofs(1:2:2*nmnodes-1)= 2*idm-1;
masterdofs(2:2:2*nmnodes)= 2*idm;

G = zeros(2*nsnodes,2*nnodes);
for i=1:ns4nodes  %Boundary 4 slave nodes first, lower node numbers

    G(2*i-1,2*ids4(i)-1) = -1;              %Slave x-dof in Gs
    G(2*i,2*ids4(i)) = -1;                  %Slave y-dof in Gs

    gamma=3;                                %Boundary 3 and 4

    [alpha masternode] = mpc_alpha(XYZ,gamma,i);

    G(2*i-1,2*id3(masternode)-1) = alpha;%Upper master x-dof in Gm
    G(2*i-1,2*id3(masternode-1)-1) = 1-alpha;%Lower master x-dof
    G(2*i,2*id3(masternode)) = alpha;    %Upper master y-dof in Gm
    G(2*i,2*id3(masternode-1)) = 1-alpha;    %Lower master y-dof
end
c=ns4nodes;                             %Shortening of name only
for i=1:ns2nodes  %Loop over boundary 2 slave nodes

    G(2*(i+c)-1,2*ids2(i)-1) = -1;
```

```
50      G(2*(i+c),2*ids2(i)) = -1;
51
52      gamma=1;                                    %Boundary 1 and 2
53
54      [alpha masternode] = mpc_alpha(XYZ,gamma,i);
55
56      G(2*(i+c)-1,2*id1(masternode)-1) = alpha;
57      G(2*(i+c)-1,2*id1(masternode-1)-1) = 1-alpha;
58      G(2*(i+c),2*id1(masternode)) = alpha;
59      G(2*(i+c),2*id1(masternode-1)) = 1-alpha;
60  end
61
62  Gs=G(:,slavedofs);                              %Gs-matrix
63  Gm=G(:,masterdofs);                             %Gm-matrix
64
65  T=zeros(2*nnodes,2*nmnodes);
66  T(masterdofs,:) = diag(ones(2*nmnodes,1));
67  T(slavedofs,:) = -inv(Gs)*Gm;
68
69  T=T(BCcorner,:);                            %Taking away corner dofs
70  end
```

mpc_alpha.m

```
1   function [alpha masternode] = mpc_alpha(XYZ,gamma,i)
2
3   if gamma==1
4       id1=find(abs(XYZ(:,1)-max(XYZ(:,1)))<10^-10);%Boundary 1 node nr
5       id2=find(abs(XYZ(:,1)-min(XYZ(:,1)))<10^-10);%Boundary 2 node nr
6       xy1=XYZ(id1,2); xy2=XYZ(id2,2);         %Boundary node y-coord
7   else
8       id1=find(abs(XYZ(:,2)-max(XYZ(:,2)))<10^-10);%Boundary 3 node nr
9       id2=find(abs(XYZ(:,2)-min(XYZ(:,2)))<10^-10);%Boundary 4 node nr
10      xy1=XYZ(id1,1); xy2=XYZ(id2,1);         %Boundary node x-coord
11  end
12
13  nb1nodes=length(id1);                       %Number of boundary 1 nodes
14  Lb=xy1(nb1nodes)-xy1(1);                    %Length of boundary
15  lambdaNatcoord1=2*(xy1-min(xy1))./Lb - 1;%Boundary 1 nat node coord
16  lambdaNatcoord2=2*(xy2-min(xy2))./Lb - 1;%Boundary 2 nat node coord
17
18  b1=lambdaNatcoord1; b2=lambdaNatcoord2;  %Shortening of name only
19
20  xi=b2(i+1);
21  for j=2:nb1nodes
22      if xi<=b1(j)              %If slave node lower than master node
23          alpha = (xi-b1(j-1))/(b1(j)-b1(j-1));
24          masternode=j;         %Upper master node
25          return
```

```
    end
end
```

B.5 Backus2D.m

```
%% Calculates Backus average
clear all
clc

%5.1.2 Layered Materials with Matching Grids
E=[100 1000 10 1 0.01 1000 0.1 10 100 1];
nu=[0.45 0.405 0.36 0.315 0.27 0.225 0.18 0.135 0.09 0.045];

[D] = plane_strain(E,nu);
%[D] = plane_stress(E,nu);

a1=0;a2=0;f1=0;c1=0;l1=0;
for i=1:size(D,1)/3

a=D(i*3-2,1);
f=D(i*3-1,1);
c=D(i*3-1,2);
l=D(i*3,3);

a1=a1+a-f^2/c;
a2=a2+(f/c);
c1=c1+(1/c);
f1=f1+(f/c);
l1=l1+(1/l);
end

A=a1/i+(c1/i)^-1*(a2/i)^2;
F=(c1/i)^-1*(f1/i);
C=(c1/i)^-1;
L=(l1/i)^-1;

display('The Backus average matrix becomes')
HomoLbackus=[A F 0;F C 0;0 0 L]
```