Development about composite homogenization in static and in dynamic - application to UD composite materials

Jianhui Yang

Supervisor: Patrick Rozycki

l'Institut de Recherche en Génie Civil et Mécanique

Ecole Centrale de Nantes

A thesis submitted for the degree of

Erasmus Mundus Master

Yet to be decided

Abstract

This paper deals with two scale modeling of unidirectional Fiber-reinforced composite material, we are going to present a homogenization approach from Microscopic to Macroscopic for prediction of linear and nonlinear mechanical property of UD composite with Abaqus and Python scripts.Demonstration of Validity of the introduction of Periodic Boundary Condition and homogenization procedure will be given. At last, numerical result will be compared with experimental data.

Contents

List of Figures

Introduction

Composite Materials are engineered material made from two or more constituent materials with significantly different physical or chemical properties which remain separate and distinct on a macroscopic level within the finished structure.Composite materials have gained popularity in high-performance products that need to be lightweight, yet strong enough to take harsh loading conditions such as aerospace components (tails, wings, fuselages, propellers), boat and scull hulls, bicycle frames and racing car bodies. Other uses include fishing rods, storage tanks, and baseball bats. The new Boeing 787 structure including the wings and fuselage is composed largely of composites. Composite materials are also becoming more common in the realm of orthopedic surgery.

Composites are made up of individual materials referred to as constituent materials. There are two categories of constituent materials: matrix and reinforcement. The Matrix help to keep the local position of reinforcement constituents and reinforcement, normally are fibers, can give its mechanical property to the composite to enhance the overall mechanical property. In this paper we use fiber as the constituent material for reinforcement, and just talk about unidirectional fiber reinforcement material. In this particular category of composite material, all the fibers has the same axial direction.

To utilize the composite in the industrial, we need to know clearly the mechanical property of the material. Experiment is doing to find out the mechanical behavior, but due to the time costing economic reason, it's necessary to develop numerical method to get mechanical parameter quickly and accurately.This Master thesis is going to explore the numerical method of computing the mechanical

parameters of UD composite material. The aim of this action is to reduce the time of experiment and economic expense. In order to do that, homogenization method will be used in this paper to identify the overall parameter of UD composite. To implement it, some development for the commercial software ABAQUS will be taken place. With this tool, the user could input the 3D geometry of UD composite in ABAQUS, and input the mechanical parameter of matrix and fiber, then run the Python scripts developed from this paper, the user could easily get the overall mechanical property of the composite from the output.

In this paper, we are going to have a brief view of the feature of UD composite, and homogenization method will be introduced. According to the feature of ABAQUS, a algorithm for implementation of homogenization procedure is proposed. Then some demonstration is given to verify the validation of the algorithm. Then a brief user manual of this script will show the steps of utilization. Finally, some numerical example will be given to compare the result with experimental data.

Chapter 1

UD Composite and Homogenization

1.1 Literature Review

Prediction of the mechanical property of a unidirectional fiber reinforced composite has been an active research area.We can obtain some parameter of composite material by doing experiment, but doing that is very time costing and expensive. Thus numerical simulation is necessary to be introduced to prediction the overall property of UD composite.It's a big challenge to develop a accurate and reliable numerical method to predict the mechanical behavior of composite.

A common idea is replacing heterecious composite material to a kind of equivalent homogenized material. Some theoretical prediction method has been developed for the elastic modulus of UD composite. They indicated that evern though resin properties are the predominant factor in transverse modulus and shear modulus, they can be improved by high modulus fibers such as boron.And the existing micromechanical models has been compared. Experimental results on boron fiber reinforced composites indicate reasonable agreement with theory for longitudinal and transverse modulus, whereas very poor agreement with theory is obtained in the case of shear modulus.[4,12,5]

Beside the effective property of composite material, some researchers begin to study the micro-scale mechanical behavoir of composite matherial, to get a deeper understanding of the internal physical information. The study of local field behavior is in the domain Micromechanics, doing that will involve reasonable computing effect. Direct numerical simulation by FEM for micromechanics with periodic microstructures was done in the end of 1960s. Recently, model with complex microstructures was considered.

Significant progress about replacing the heterogenous material with homogeneous material was made in 1970s, with the introduction of mathematical theorem Homogenization.[1] In these research, author focused on linear constituents and periodic microstructures. After that, research were extended to plastic material.[7,10,6]

1.2 Feature of UD composite material and Homogenization approach

1.2.1 UD composite material

A typical UD composite material is showed below,the fibers are aligned in one direction in the matrix. The direction parallel to the fibers is the "axial" direction and the direction perpendicular to the fiber is the transverse direction.The properties of the composite are dependent on the properties of constituents and orientation of the fiber.

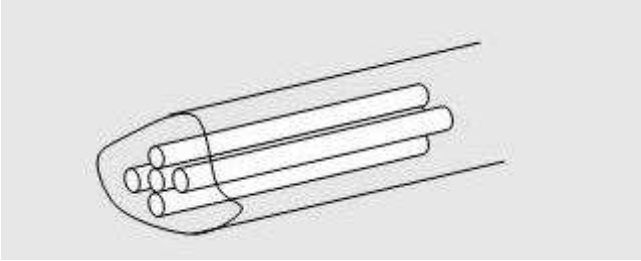


Figure 1.1: Typical UD composite

1.2 Feature of UD composite material and Homogenization approach

The mechanical properties of composite materials are generally not isotropic due to the diofferents behavior of consitutants and due to the fiber direction The relationship between stress and strain for an isotropic material in elastic can be described with three parameters: Young's Modulus, the Shear Modulus and the Poisson's ratio, in a relatively simple form. For the anisotropic material, it requires the mathematics of a second order tensor and up to 21 material property constants.

For example, the figure below illustrates the different mechanical behavior under same loading between isotropic material and anisotropic material.(In this case is the UD composite with the axial direction angle 45 degrees)[3]

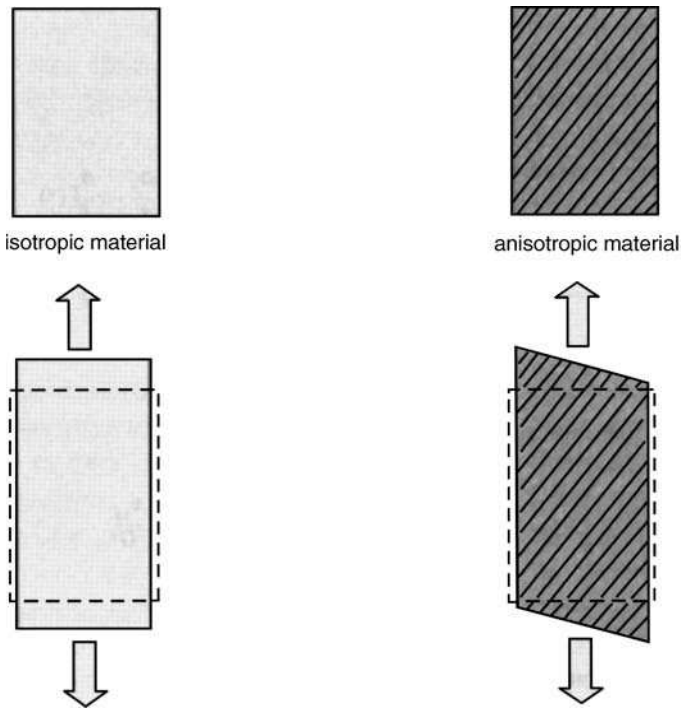


Figure 1.2: Homogeneous Material and UD composite material under same loading

We take an 2D example to illustrate the reason of these differences. For 2D

problem. The relation below is valid for anisotropic and elastic material.[1]

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{pmatrix} = \begin{pmatrix} \frac{1}{E_x} & -\frac{\nu_{yx}}{E_y} & 0 \\ -\frac{\nu_{xy}}{E_x} & \frac{1}{E_y} & 0 \\ 0 & 0 & \frac{1}{G_{xy}} \end{pmatrix} \begin{pmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{pmatrix}$$

From the equation, we can find that, for anisotropic material, to predict the mechanical property with more precision in behavior,we need to more parameter.

1.2.2 Introduction of Concept of RVE

The homogenization procedure usually contain introduction of two scales, Marcro-scale and Micro Scale, Marco-scale is usually refer to a homogenized continuous medium, and micro-scale is usually related to a statistically representative volume element[11] (RVE originally introduced by Hill 1963).Instead of predicting the mechanical property of composite, we use homogenization technique and consideration of RVE to predict the overall property from microscopic level.

The RVE must be selected such that the microstructure should be composed as copies of RVEs, and without overlapping and voids between the boundaries. The choice of RVE is usually not unique, but it should be big enough to present the feature of the material and as small as possible to reduce the computation cost.Also, the RVE should have the same volume friction with the composite.The figure below gives an example of different selection of RVE.[8]

[1] u is the displacement, $\varepsilon_x = \partial u_x/\partial x$, $\varepsilon_y = \partial u_y/\partial y$, $\gamma_{xy} = \partial u_x/\partial y + \partial u_y/\partial x$

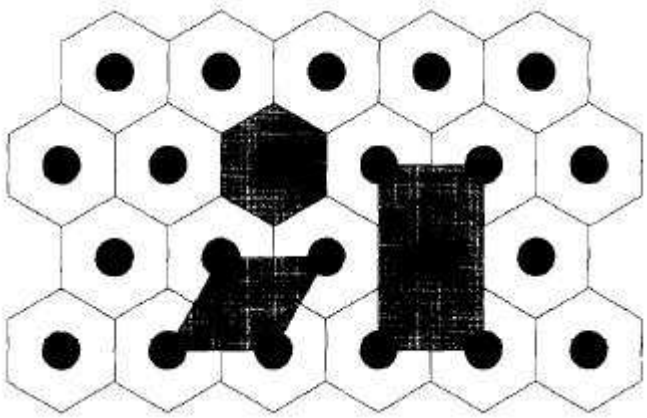


Figure 1.3: Different choice of RVE

1.2.3 Homogenization

We use properties of individual constituents and micromechanical analysis, homogenization method to predict the overall composite behavior. Homogenization approach is the "bridge" between the microscopic and macroscopic. Generally, there are two principle steps to implicit a homogenization procedure

- Compute microscopic problem with RVE to get local stress and strain
- Use homogenization method to link microscopic problem to macroscopic problem, and get the global stress and strain

For UD composite material, most of the homogenization approaches assume that the material has global periodicity on microstructure. Globally the material is consist of spatially repeated RVE.

There are variety of homogenization approach to predict the property of composite material, for example Mean field method. In this paper, we use Mean Field Method to predict the property of composites. The Mean field method is widely used in prediction of composite material because of its relative low computation

cost which enable us to use implicit constitutive laws in large scale simulations.

With the using of homogenization approach and RVE, we need to pay attention to following two points.

- The proper selection of RVE
- Because of periodicity of RVE, proper boundary condition should be selected.

1.2.4 Boundary conditions

Periodic Boundary condition are imposed on RVE, and the choice of this boundary conditions has been justified by a number of researchers.They prove that, by applying periodic boundary condition, we can get more reasonable result compare with other boundary conditions like stress uniform boundary condition(SUBC) or kinematic uniform boundary conditions(KUBC). A typical RVE with Periodicity is showed in below.[11]

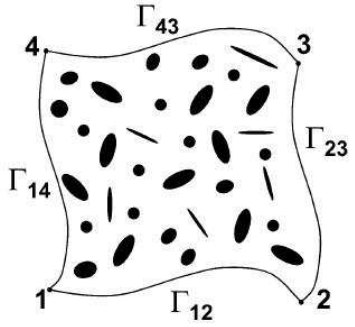


Figure 1.4: A typical RVE with periodic boundary conditions

The periodic boundary conditions imply the following two fact:

- On the opposite surface, the deformation and the orientation should be same

- In order to have stress continuity across the boundaries the stress vectors acting on opposite sides are opposite in direction.

$$u = Ex + u^* \qquad u^* \text{ is a periodic function}$$

Take the RVE of figure 1.4 as an example. We consider the points on the boundaries $\Gamma_{14}, \Gamma 23$.

$$u(x_{\Gamma_{14}}) = Ex_{\Gamma_{14}} + u^*(x_{\Gamma_{14}}) \tag{1.1}$$
$$u(x_{\Gamma_{23}}) = Ex_{\Gamma_{23}} + u^*(x_{\Gamma_{23}}) \tag{1.2}$$

Use 1.2-1.1, we have

$$u(x_{\Gamma_{23}}) - u(x_{\Gamma_{14}}) = E(x_{\Gamma_{23}} - x_{\Gamma_{14}}) \tag{1.3}$$

More generally, we have A and B are two points on the opposite surface of RVE.Then we have periodic boundary conditions as followed

$$u(x_B) = u(x_A) + E(x_B - x_A) \quad or \tag{1.4}$$
$$u(x_B) - u(x_A) = E(x_B - x_A) \quad A, B \quad on\Gamma \tag{1.5}$$
$$\varepsilon(u(x)) = E + \varepsilon(u^*(x)), \quad u^* \quad periodic \tag{1.6}$$

Now we introduce the brackets $\langle . \rangle$ denote the volume average of a field V.

$$\langle f \rangle = \frac{1}{|V|} \int_V f(x)\, dx$$

1.3 Computational Homogenization

1.3.1 Initial boundary condition for Microscopic level

In the microscopic level, the material should satisfy the equations below

$$\nabla \cdot \sigma = 0 \quad in\Omega \quad \text{(equilibrium equations)} \tag{1.7}$$

$$\sigma_{ij} = L_{ijkl}\varepsilon_{kl} \quad in\Omega \quad \text{(constitutive equations)} \tag{1.8}$$

$$\varepsilon_{ij} = u_{i,x_j} \quad in\Omega \quad \text{(Kinematic equations)} \tag{1.9}$$

$$u_i = \overline{u}_i \quad on\Gamma_u \quad \text{(Displacement boundary conditions)} \tag{1.10}$$

$$\sigma_{ij} = \bar{t}_i \quad on\Gamma_i \quad \text{(Traction boundary condition)} \tag{1.11}$$

$$\begin{gathered} u(x_B) - u(x_A) = E(x_B - x_A) \\ A, Bon\Gamma \quad \text{(Periodic boundary conditions)} \end{gathered} \tag{1.12}$$

1.3.2 Coupling of Microscopic and Macroscopic scales

In classical lamination theory the composite lamina is considered as a homogeneous orthotropic material with effective modulus that describe the overall mechanical property of composite. To obtain this macroscopic modulus, a homogenization procedure are introduced during the computing linking the microscopic scale and the macroscopic scale.

One of the mostly used method for homogenization is doing an volume integration over the RVE to averaging the stress and strain tensor:

$$\overline{\sigma}_{ij} = \frac{1}{V}\int_V \sigma_{ij}(x,y,z)\,dV \tag{1.13}$$

$$\overline{\varepsilon}_{ij} = \frac{1}{V}\int_V \varepsilon_{ij}(x,y,z)\,dV \tag{1.14}$$

We need to demonstrate that this procedure can keep the equivalence of strain energy between the heterogeneous composite and the homogeneous composite from averaging the stress and strain.In order to demonstrate the equivalence of the energy, we assume that we have boundary traction t_i and boundary displacement u_i.

The strain energy in a homogeneous material with volume V is:

$$U = \frac{1}{2}\overline{\sigma}_{ij}\overline{\varepsilon}_{ij}V \tag{1.15}$$

The strain energy stored in the heterogeneous material of volume V is:

$$U' = \tfrac{1}{2}\int_V \sigma_{ij}\varepsilon_{ij}\, V \tag{1.16}$$

$$U' = \tfrac{1}{2}\int_V \sigma_{ij}(\varepsilon_{ij} + \overline{\varepsilon}_{ij} - \overline{\varepsilon}_{ij})\, V \tag{1.17}$$

$$= \tfrac{1}{2}\int_V \sigma_{ij}(\varepsilon_{ij} - \overline{\varepsilon}_{ij})\, V + \tfrac{1}{2}\overline{\varepsilon}_{ij}\int_V \sigma_{ij}\, V \tag{1.18}$$

As we have illustrated in Equation 1.13, so we have

$$U' = \tfrac{1}{2}\int_V \sigma_{ij}(\varepsilon_{ij} - \overline{\varepsilon}_{ij})\, V + \tfrac{1}{2}\overline{\sigma}_{ij}\overline{\varepsilon}_{ij} V \tag{1.19}$$

$$= \tfrac{1}{2}\int_V \sigma_{ij}(\varepsilon_{ij} - \overline{\varepsilon}_{ij})\, V + U \tag{1.20}$$

$$U' - U = \tfrac{1}{2}\int_V \sigma_{ij}(\tfrac{\partial u_i}{\partial x_j} - \tfrac{\partial \overline{u}_i}{\partial x_j})\, V \tag{1.21}$$

$$= \tfrac{1}{2}\int_V [\tfrac{\partial \sigma_{ij}}{\partial x_j}(u_i - \overline{u}_i) + \tfrac{\partial(\sigma_{ij}(u_i - \overline{u}_i))}{\partial x_j}]\, V \tag{1.22}$$

Use the equilibrium equation

$$\frac{\partial \sigma_{ij}}{\partial x_j} = 0$$

We can write the energy difference as

$$U' - U = \frac{1}{2}\int_V \frac{\partial}{\partial x_j}[\sigma_{ij}(u_i - \overline{u}_i)]\, V \tag{1.23}$$

By using Gauss theorem we can transform the integration of the volume into the integration of the surface

$$U' - U = \frac{1}{2}\int_S \sigma_{ij}(u_i - \overline{u}_i) n_j\, S \tag{1.24}$$

S is the surface of the RVE, and n denote the unit outward normal vector and On the surface S, we have

$$u_i = \overline{u}_i$$

So we have conclusion that

$$U = U'$$

The derivation above showed that, the homogenization we used ensure the equivalence between the heterogeneous RVE and the homogenized RVE.

Chapter 2

Implement Homogenization in Abaqus

2.1 Introduction of implementation

In order to implement the Periodic Boundary Conditions in Abaqus, we need to use constrain functions in Abaqus, however, this functions is designed for points constrain only. To apply this BC on the all boundaries of the model, we need to creates hundreds of constrain equations for every couple of points on the corresponding surfaces. Considering the scale of the problem, it's not realistic to create the constrain equations manually.

Abaqus Scripting Interface(API) enable us to costmize the Abaqus ourselves. With the API, we can

- Automate repetitive tasks
- Extend functionality
- Enhance the interface

Creating the constrain equations is a kind of repetitive task, so it is possible to do that with API.

Python is the standard programming language for ABAQUS products, it has a very good compatibility with Abaqus.In Abaqus,

- The ABAQUS environment file uses Python statements.
- The parameter definitions on the data lines of the *PARAMETER option in the ABAQUS input file are Python statements.
- The parametric study capability of ABAQUS requires the user to write and to execute a Python scripting (.psf) file.
- ABAQUS/CAE records its commands as a Python script in the replay (.rpy) file.
- You can execute ABAQUS/CAE tasks directly using a Python script.
- You can access the output database (.odb) using a Python script.

And Python is an interpreted language. This means you can type a statement and view the results without having to compile and link your scripts. Experimenting with Python statements in Abaqus is quick and easy.[9]

These advantages make Python the best programming language for creating scripts for Abaqus.

In this chapter, we are going to illustrate how to implement homogenization procedure in Abaqus and give some demonstrations of validity.First we are going to have some preview of Python language, after that, the constrain equation function in Abaqus will be introduced. Then, the general structure of Python scripts will be given, at last, some demonstration will be given to proof the validity of the method.

2.2 Use Python scripts to implement homogenization

2.2.1 Abaqus Scripting Interface and Python

The Abaqus Scripting Interface is an application programming interface (API) to the models and data used by Abaqus. The Abaqus Scripting Interface is an extension of the Python object-oriented programming language; Abaqus Scripting Interface scripts are Python scripts. You can use the Abaqus Scripting Interface to do the following:[9]

- Create and modify the components of an Abaqus model, such as parts, materials, loads, and steps.
- Create, modify, and submit Abaqus analysis jobs.
- Read from and write to an Abaqus output database.
- View the results of an analysis.

In Abaqus, there is no option for applying periodic boundary conditions. Python scripts enable us to implement Homogenization procedure in Abaqus for composite material.In order to do that, we need to create a python scripts which should include following functions (This Python scripts inquire the RVE should be cuboid)

- Read the mesh database and find the boundaries
- Scan the geometry model and find opposite surfaces
- Pick up corresponding coordinates of opposite surfaces from mesh database
- create reference points to implement homogenization approach
- create constrain equations for applying periodic boundary conditions

2.2.2 Constrain equations for periodic boundary conditions

In Interaction module of Abaqus, we can define different kind of interactions between parts of the model or inside the model. In constrain submodule, it provide a possibility to create a linear equation constraint by entering data in the Edit Constraint dialog box. The terms of an equation consist of a coefficient applied to a degree of freedom of every node in a set.

A linear multi-point constraint requires that a linear combination of nodal variables is equal to zero; that is,

$$A_1 u_i^P + A_2 u_j^Q + \cdots + A_N u_k^R = 0$$

Where u_i^P is a nodal variable at node P, degree of freedom i, and the A_n are coefficients that define the relative motion of the node.In general, we need to specify following parameters to define a linear constraint equation

- the number of terms in the equation, N
- the nodes, P, and the degrees of freedom, i, corresopnding to the nodal variables u_i^P
- coefiicients, A_n

For example, to impose the equation

$$u_1^2 - u_2^3 - 2u_5^1 = 0 \quad u_i \text{ is the displacement of point } i$$

We need to define three point set to contain point1,point3 and point5. Use python command

mdb.models[ModelName].rootAssembly.Set('pointset1',nodes=Noeuds[(location in the database)])
Then use command

mdb.models[ModelName].Equation(name=Constraintname,terms=((1.0,'pointset1',2),(-1.0,'pointset2',3),(-2,"pointset3,1)))

to creat the linear constrain illustrate above, we use loops to read the database, and create the constrain equations automatically.

In our model for 3D cuboid RVE, we need to apply periodic boundary condition 1.6 to the computation, We have Macroscopic strain matrix $E = E_{ij} \quad i, j = 1, 2, 3$. In order to include Macroscopic strain component E_{ij}, we can create six reference point RP-1, RP-2 ... RP-6. Their displacement of degree of freedom one stand for $E_{11}, E_{12}, E_{13}, E_{22}, E_{23}, E_{33}$ respectively.[1]

$$\begin{aligned} U^1_{RP-1} &= E_{11} \\ &\vdots \\ U^1_{RP-6} &= E_{33} \end{aligned}$$

By introducing six reference points, we can apply periodic boundary conditions by creating constrain equations using the python command above in Abaqus.

2.2.3 Validly of homogenization procedure

In this chapter,we are going to demonstrate that the introduction of six reference could give correct output for macroscopic strain.

To verify the accuracy of the output of macroscopic strain from the displacement of reference points, we need to demonstrate that the displacement of reference points are exactly the integration of local strain over the RVE. Take the RVE in fig as an example.

[1] u is the macro displacement, $E_{11} = \partial u_x/\partial x$, $E_{22} = \partial u_y/\partial y$, $E_{12} = (\partial u_x/\partial y + \partial u_y/\partial x)/2$

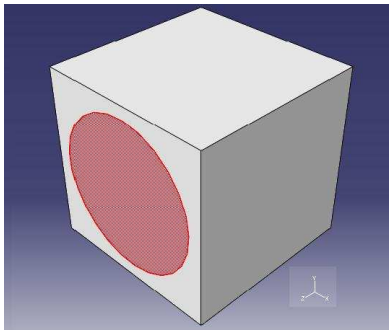


Figure 2.1: Representative Element Volume

Because of the different mechanical properties of matrix and fiber, the strain across the interaction face of matrix and fiber may be not continues, so we divide the integration area into two, the matrix and the fiber. In order to use Gauss theorem during the demonstration, we need to ensure the connective property of integral regions. So we set the boundary of two regions as follow:[2]

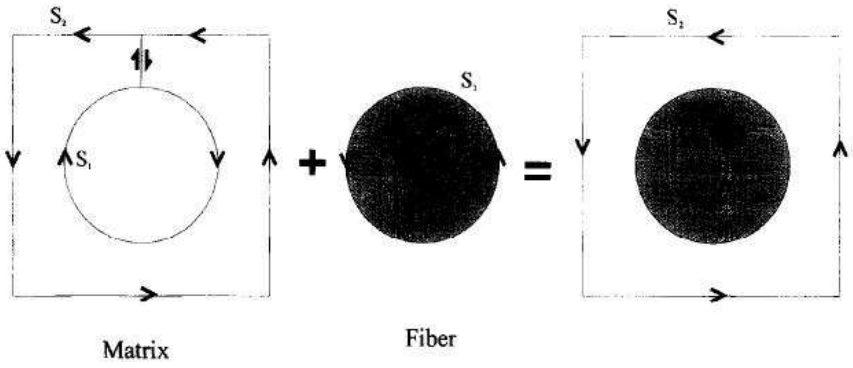


Figure 2.2: Integration contour for square array

Now, we are back to the definition of macroscopic strain. S_1 is interaction surface between fiber and matrix, and S_2 is the outer surface of RVE.

$$\begin{aligned}\bar{\varepsilon}_{ij} =& \frac{1}{V}\left(\int_V \varepsilon_{ij}\, dV\right) \\ =& \frac{1}{V}\left(\int_{V_f} \varepsilon_{ij}\, dV\right) + \frac{1}{V}\left(\int_{V_m} \varepsilon_{ij}\, dV\right)\end{aligned} \tag{2.1}$$

Use Gauss theorem, transform the integration of volume to the surface integration

$$\begin{aligned}\bar{\varepsilon}_{ij} =& \frac{1}{V}(\frac{1}{2}\int_{S_1}(u_i n_j + u_j n_i)\,dS)) \\ &+ \frac{1}{V}(\frac{1}{2}\int_{S_2}(u_i n_j + u_j n_i)\,dS) - \frac{1}{2}\int_{S_1}(u_i n_j + u_j n_i)\,dS))\end{aligned} \tag{2.2}$$

So we can get the volume integration of local strain from the displacement of boundary

$$\bar{\varepsilon}_{ij} = \frac{1}{V}\int_{S_2}(u_i n_j + u_j n_i)\,dS \tag{2.3}$$

We assume we are using triangle element mesh, and first order shape function in the finite element computing. We are going to prove that if E_{ij} satisfy the constrain equation 1.6, E_{ij} is equal to the volume integration of ε_{ij}. We take E_{11} as an example to demonstrate the validity. The RVE model is showed in Fig 2.1 , with lenght, width, and height α, β, γ respectively.

In this Case, we have

$$u_A - u_B = \alpha E_{11} \tag{2.4}$$

for every point on surface A and B.

In finite element analysis, we use linear combination of shape function to describe the displacement function. Within a 3-noded triangular element we can interpolate the displacement as:

$$U = U_1 N_1 + U_2 N_2 + U_3 N_3 \tag{2.5}$$

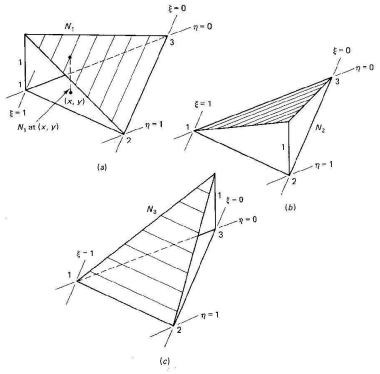


Figure 2.3: Shape functions

Where n_1, N_2, N_3 are shape functions illustrated in Fig 2.3 They are linear unctions of nondimensional coordinates ξ and η

$$N_1 = \xi, \quad N_2 = \eta, \quad N_3 = 1 - \xi - \eta \tag{2.6}$$

$$\int_\Omega N_i \, ds = \tfrac{1}{3} S_\Omega \tag{2.7}$$

We start from one of the element A_e on the surface of A, and with the area S_e,because of the symmetric of mesh, on the opposite side, we have an element B_e which has the same shap with A_e we assume that $\Omega_e = A_e \cup B_e$. In element Ω_e, we have constrain equations from the periodic boundary conditions we applied in the Abaqus.

$$u_{Ai} - u_{Bi} = \alpha E_{11} \quad i = 1, 2, 3 \quad in \Omega_e \tag{2.8}$$

And

$$u_A = \textstyle\sum_{i=1}^{n} u_{Ai} N_i \quad in \quad A_e \tag{2.9}$$

$$u_B = \textstyle\sum_{i=1}^{n} u_{Bi} N_i \quad in \quad B_e \tag{2.10}$$

Equations 2.8 multiply $\frac{1}{3} S_e \quad \Rightarrow$

$$u_{Ai} \frac{1}{3} S_e - u_{Bi} \frac{1}{3} S_e = \frac{1}{3} S_e \alpha E_{11} \quad i = 1, 2, 3 \tag{2.11}$$

Because we have 2.7,

$$N_1 = \xi, \quad N_2 = \eta, \quad N_3 = 1 - \xi - \eta \tag{2.12}$$

$$\int_\Omega N_i \, ds = \tfrac{1}{3} S_\Omega \tag{2.13}$$

substitude formula 2.7 into equation2.11, we have

$$\int_{A_e} u_{Ai} N_i \, ds - \int_{B_e} u_{Bi} N_i \, ds = S_e \alpha E_{11} \quad i = 1, 2, 3 \tag{2.14}$$

Because the normal vector of surface A and B are opposite, $n_A = (1, 0, 0) \quad n_B = (-1, 0, 0)$ So, we can write 2.14 as

$$\int_{\Omega_e} u_{Ai} N_i - u_{Bi} N_i \, ds = S_e \alpha E_{11} \quad i = 1, 2, 3 \tag{2.15}$$

Use relation 2.9,2.10 and sum these three equation, we get

$$\int_{\Omega_e} u_A - u_B \, ds = S_e \alpha E_{11} \quad \Rightarrow \tag{2.16}$$

Sum all the element Ω_e on the surface A and B, we get the equation for the whole surface.

$$\int_\Omega u_1 n_1 \, ds = V E_{11} \quad \Rightarrow \tag{2.17}$$

$$E_{11} = \tfrac{1}{V} \int_{S_2} u_1 n_1 \, ds \quad \Rightarrow \tag{2.18}$$

$$\overline{\varepsilon}_{ij} = E_{11} \tag{2.19}$$

The prove for other component of E_{ij} is similar. This prove enable us to compute the value of E_{ij} in the constrain equation instead of computing the volume integration in homogenization procedure.

2.2.4 Equation elimination

To apply the periodic boundary conditions, we need to convert the boundary conditions on the surface into point constrain equations on mesh points. However, during this procedure, we may have some extra equations which could be derived by other equations. Solving the equation systems without eliminating these extra equations could cause singular matrix calculation. Although in some

software, they can deal with this kind of cases automatically, but in Abaqus, entering equation systems with extra formulas will cause error. So we need to avoid such situation and it is necessary to explore our constrain equation systems, and eliminate extra equations.

We start with 2D single fiber RVE model. And we can see why we need to eliminate the extra equations. Suppose we have a RVE, length and width are α and β respectively. Now, we consider equations of four coner points:

$$u_{B1} - u_{A1} = \alpha E_{11} \tag{2.20}$$
$$u_{B2} - u_{A2} = \alpha E_{12} \tag{2.21}$$
$$u_{C1} - u_{D1} = \alpha E_{11} \tag{2.22}$$
$$u_{C2} - u_{D2} = \alpha E_{12} \tag{2.23}$$
$$u_{A1} - u_{D1} = \beta E_{12} \tag{2.24}$$
$$u_{A2} - u_{D2} = \beta E_{22} \tag{2.25}$$
$$u_{B1} - u_{C1} = \beta E_{12} \tag{2.26}$$
$$u_{B2} - u_{C2} = \beta E_{22} \tag{2.27}$$

Degree freedom 1 and degree freedom 2 are independent to each other.We take the solving of DOF 1 as an example to show the necessary of elimination.

$$u_{B1} - u_{A1} = \alpha E_{11} \tag{2.28}$$
$$u_{C1} - u_{D1} = \alpha E_{11} \tag{2.29}$$
$$u_{A1} - u_{D1} = \beta E_{12} \tag{2.30}$$
$$u_{B1} - u_{C1} = \beta E_{12} \tag{2.31}$$

Write it into matrix form:

$$\begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} u_{A1} \\ u_{B1} \\ u_{C1} \\ u_{D1} \end{pmatrix} = \begin{pmatrix} \alpha E_{11} \\ \alpha E_{11} \\ \beta E_{12} \\ \beta E_{12} \end{pmatrix} \tag{2.32}$$

We can find that the matrix on the left is a singular matrix. So it's not possible to do the matrix calculation to get the value of u.

Using elimination to avoid the singular matrix.

$$\begin{aligned} u_{B1} - u_{C1} &= u_{B1} - u_{A1} + u_{A1} - u_{D1} + u_{D1} - u_{C1} \\ &= \alpha E_{11} + \beta E_{12} - \beta E_{12} \\ &= \alpha E_{11} \end{aligned} \tag{2.33}$$

Equation2.28,Equation2.30,Equation2.31⇒ Equation2.29

$$\begin{aligned} u_{B2} - u_{C2} &= u_{B2} - u_{A2} + u_{A2} - u_{D2} + u_{D2} - U_{C2} \\ &= \alpha E_{12} + \beta E_{22} - \beta E_{22} \\ &= \alpha E_{12} \end{aligned} \tag{2.34}$$

Equation2.21,equation2.25,equation2.27 ⇒ equation2.23

From the derivation above, we can eliminate two extra equations from the system Now, we extend our solution to 3D case. Consider a RVE single fiber model in Fig2.1.

We categorise the mesh points into three sorts:

- points on the surface but not on the edges and corner
- points on the edges but not on corner
- points on the corner

For the points on the surface, the equation system is unique, there are no extra equations in the system. For the points on the edges, we can consider it as 2D problem, take the Fig as an example, Points A, B, C, D could be looked as point on four corners of a square, then we can use the solution above to eliminate two extra equations.

The elimination for corner points of a cuboid will be more complex. In this case, we have eight points to be considered: a, b, c, d, k, l, m, n. And the length ,

width and height are α, β, γ respectively. To simplify the demonstration, we take degree of freedom 1 as an example.

Equations system for corner points of degree of freedom 1

$$u_{a1} - u_{b1} = \alpha E_{11} \tag{2.35}$$
$$u_{n1} - u_{m1} = \alpha E_{11} \tag{2.36}$$
$$u_{k1} - u_{l1} = \alpha E_{11} \tag{2.37}$$
$$u_{d1} - u_{c1} = \alpha E_{11} \tag{2.38}$$
$$u_{a1} - u_{k1} = \beta E_{12} \tag{2.39}$$
$$u_{b1} - u_{l1} = \beta E_{12} \tag{2.40}$$
$$u_{d1} - u_{n1} = \beta E_{12} \tag{2.41}$$
$$u_{c1} - u_{m1} = \beta E_{12} \tag{2.42}$$
$$u_{k1} - u_{n1} = \gamma E_{23} \tag{2.43}$$
$$u_{l1} - u_{m1} = \gamma E_{23} \tag{2.44}$$
$$u_{a1} - u_{d1} = \gamma E_{23} \tag{2.45}$$
$$u_{b1} - u_{c1} = \gamma E_{23} \tag{2.46}$$

Now, we are going to demonstrate that equation system consisted of 7 equations is enough to derive the other 5 equations.

$$\begin{aligned} u_{a1} - u_{b1} &= u_{d1} - u_{c1} + u_{c1} - u_{b1} + u_{a1} - u_{d1} \\ &= \alpha E_{11} + \gamma E_{23} - \gamma E_{23} \\ &= \gamma E_{23} \end{aligned} \tag{2.47}$$

Equation2.38,equation2.45,equation2.46 $\Rightarrow$ equation2.35

$$\begin{aligned} u_{a1} - u_{k1} &= u_{d1} - u_{n1} + u_{a1} - u_{d1} + u_{n1} - u_{k1} \\ &= \beta E_{12} + \gamma E_{23} - \gamma E_{23} \\ &= \beta E_{12} \end{aligned} \tag{2.48}$$

Equation2.41,equation2.45,equation2.43 ⇒ equation2.39

$$\begin{aligned} u_{c1} - u_{m1} &= u_{d1} - u_{n1} + u_{n1} - u_{m1} + u_{c1} - u_{d1} \\ &= \beta E_{12} + \alpha E_{11} - \alpha E_{11} \\ &= \beta E_{12} \end{aligned} \tag{2.49}$$

Equation2.41,equation2.38,equation2.36 ⇒ equation2.42

$$\begin{aligned} u_{k1} - u_{l1} &= u_{n1} - u_{m1} + u_{k1} - u_{n1} + u_{m1} - u_{l1} \\ &= \alpha E_{11} + \gamma E_{23} - \gamma E_{23} \\ &= \gamma E_{23} \end{aligned} \tag{2.50}$$

Equation2.36,equation2.43,equation2.44 ⇒ equation2.37

$$\begin{aligned} u_{b1} - u_{l1} &= u_{c1} - u_{m1} + u_{b1} - u_{c1} + u_{m1} - u_{l1} \\ &= \beta E_{12} + \gamma E_{23} - \gamma E_{23} \\ &= \beta E_{12} \end{aligned} \tag{2.51}$$

Equation2.38,equation2.42,equation2.44 ⇒ equation2.40

From the derivation above, we eliminate equation 2.35,2.37,2.39,2.40,2.42, from the equation system. And finally, we get the equation system of corner points of DOF 1 for Abaqus

$$\begin{aligned} u_{n1} - u_{m1} &= \alpha E_{11} \\ u_{d1} - u_{c1} &= \alpha E_{11} \\ u_{d1} - u_{n1} &= \beta E_{12} \\ u_{k1} - u_{n1} &= \gamma E_{23} \\ u_{l1} - u_{m1} &= \gamma E_{23} \\ u_{a1} - u_{d1} &= \gamma E_{23} \\ u_{b1} - u_{c1} &= \gamma E_{23} \end{aligned}$$

For degree of freedom 2 and 3, the elimination is exactly the same. Now, we have considered all the mesh points on the surface of RVE. The equation system now is ready to be input in Abaqus by Python scripts.

2.2.5 Brief instruction of the use of Python scripts

With the preparation above, here gives the general structure of the use of Python Scripts for applying periodic boundary conditions and homogenization procedure.

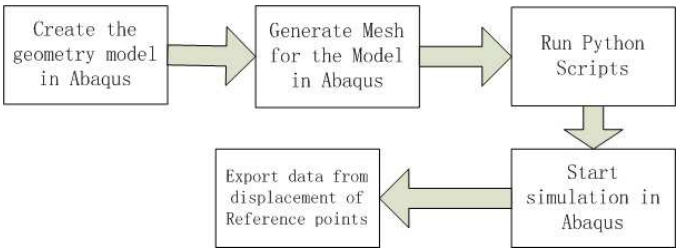


Figure 2.4: Instruction of use of Python Scripts

The complete code with comments could be found at the appendix of this paper.

To use this python scripts, you need to create a .CAE file in Abaqus, with geometry, material property of fiber and matrix, symmetric mesh and macroscopic loading. Then run Python scripts in Abaqus, the scripts will add the reference points, create the constrain equations automatically. When the scripts is done, run the simulation in Abaqus, then you can get the result from Visualization. Microscopic strain and stress could be export from Abaqus database, macroscopic stress is the initial condition used input, and macroscopic strain are the displacement of DOF 1 of six reference points. Now, a homogenization simulation for composite material is complete.

Chapter 3

Numerical simulation and experment

3.1 Experiment for UD composite

Some experiment has been done for single layer UD composite laminar. The component of the composite is epoxy (Matrix) and E-glass (fiber). The friction volume is $57\% \pm 3\%$ The mechanical property of constituent[3]

Constituent	Young's modulus E	Poisson ratio ν
Epoxy 977-20	3GPa	0.4
E Glass	74GPa	0.26

Table 3.1: Elastic constant

And harding law for Matrix:

GNUPLOT

We are going to do three experiment to the single composite laminar. They are showed as below:

- Fiber along with x direction
- Fiber along with y direction

- Fiber has a angle of 45 ° along with x direction

And we make Uniaxial Tension Test on these three laminar to get the macroscopic constant of E_{ij}

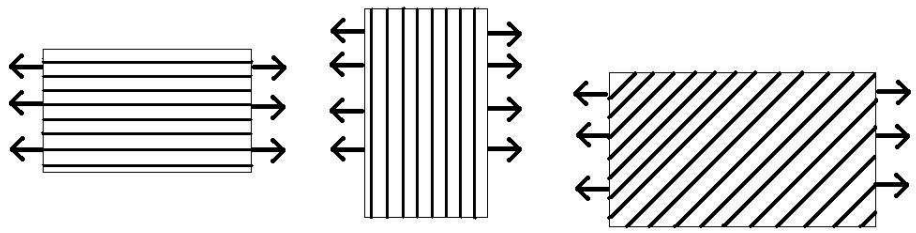


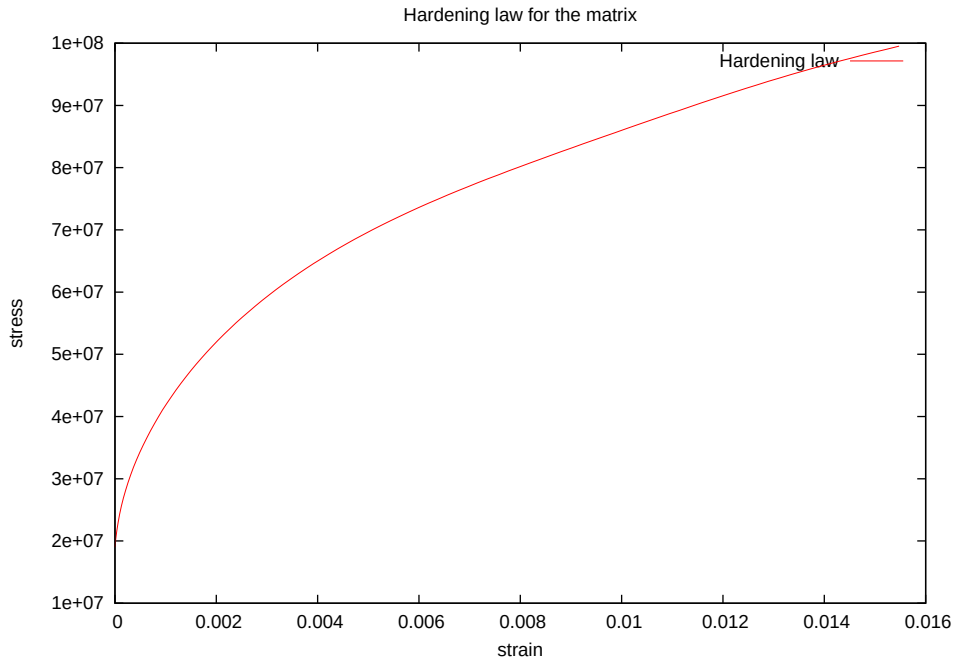
Figure 3.1: Uniaxial Tension Test

3.2 Numerical result

In this section, we are going to use Python scripts to implement the homogenization for the composite laminar illustrate above. And compare with the experimental data.

3.2.1 Fiber direction 0 °

We chose the RVE showed in Fig 2.1. Use Three-dimensional continuum elements, a 6-node linear triangular prism. For plasticity, we use Hill yield surfaces with associated plastic flow, which allow for anisotropic yield. The rupture is ignored in this case. And harding law for Matrix:



And after executive the Python scripts, the periodic boundary conditions and reference points creation is done.

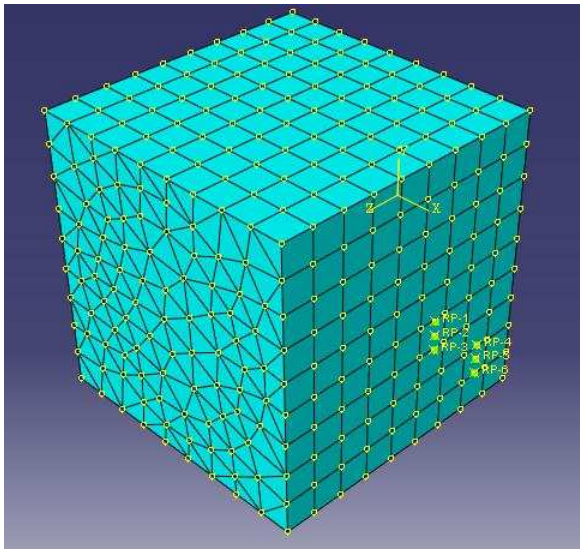


Figure 3.2: RVE with Periodic Boundary condition and mesh

In Abaqus load section, we apply pressure on the surface along with x direc-

tion.

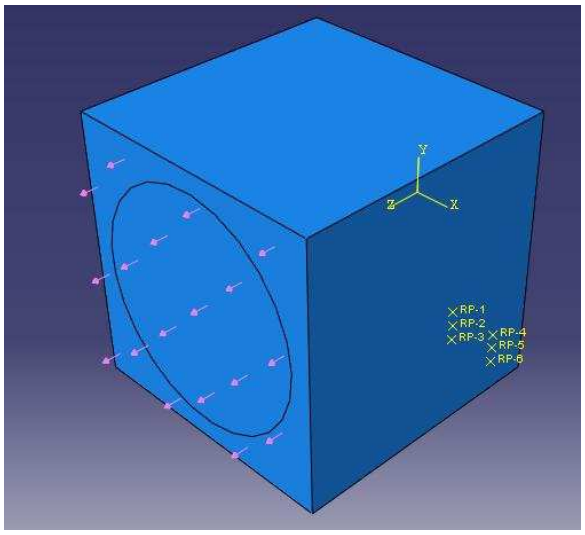


Figure 3.3: Uniaxial Tension Test

Submit the simulation, export the result as distribution of Von Mises stress.

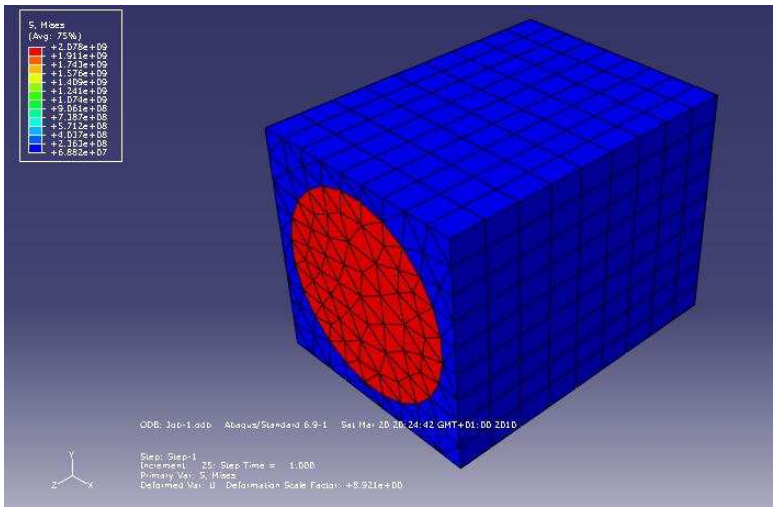


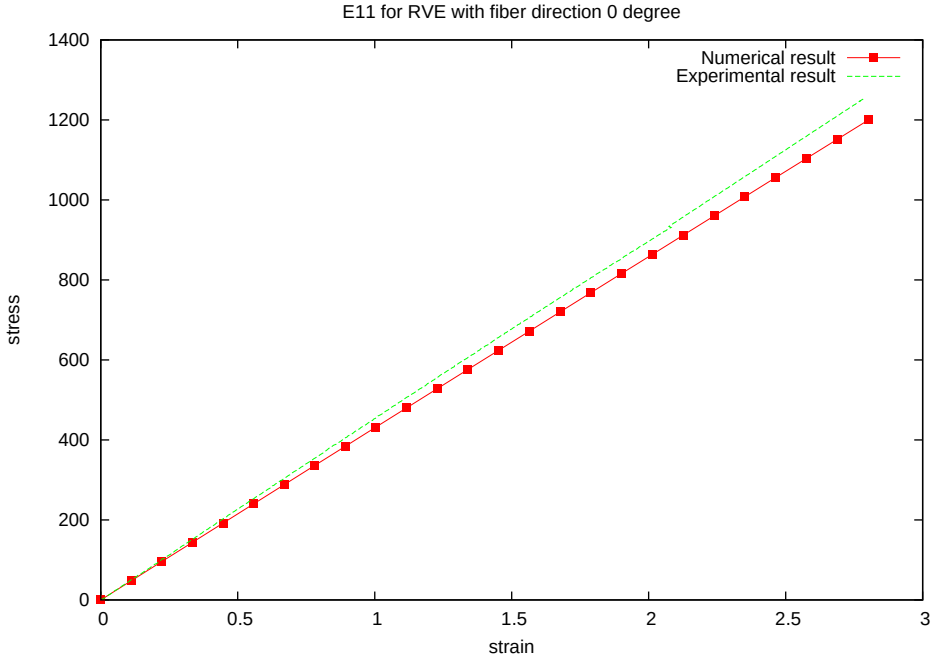
Figure 3.4: Deformation with stress distribution of the tensile test

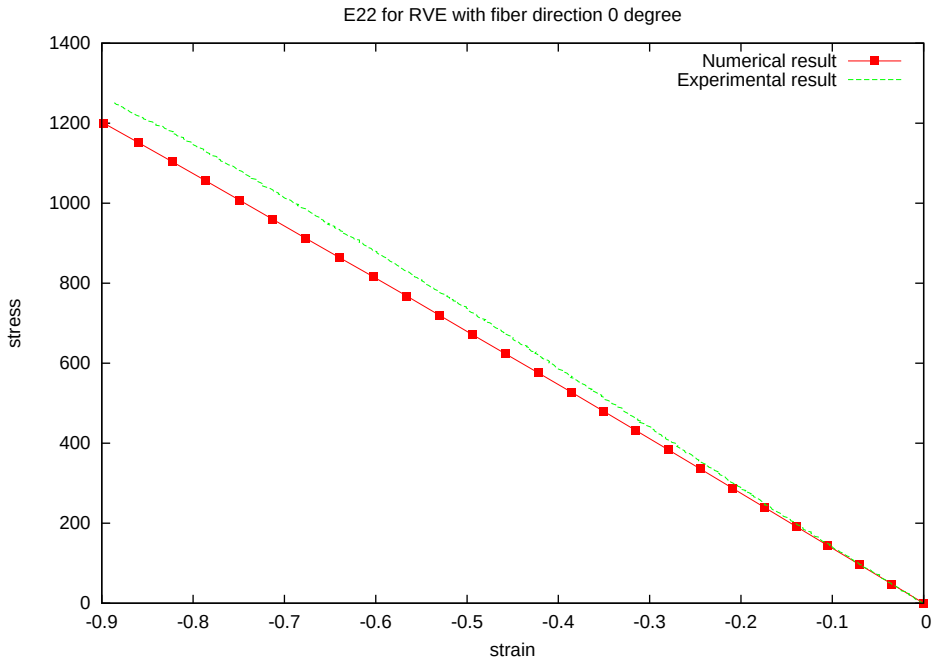
Export the macroscopic $\varepsilon_{11}, \varepsilon_{22}$ from correspondent reference points, and compare with the experimental result:

	σ_{11}(MPa)	ε_{11}(%)	ε_{22}(%)	E_{11}(MPa)	ν_{12}
numerical result	1200	2.802	-0.898	43034.8	-0.3132
Experimental data	1252	2.78	-0.89	45043	0.311

Table 3.2: Result comparison of longitudinal Young's modulus

From the figure of strain stress curve, we can find that, the numerical result match the experimental data very well at the beginning, and the error occurs when the strain increase. This error comes from the lack of consideration of rupture in the simulation. As the strain and stress increase, some crack will appear in the matrix as well as in the section between matrix and fiber.And from the plot, we can also discover that, the behavior is linear, the fiber reinforce the material efficiently.



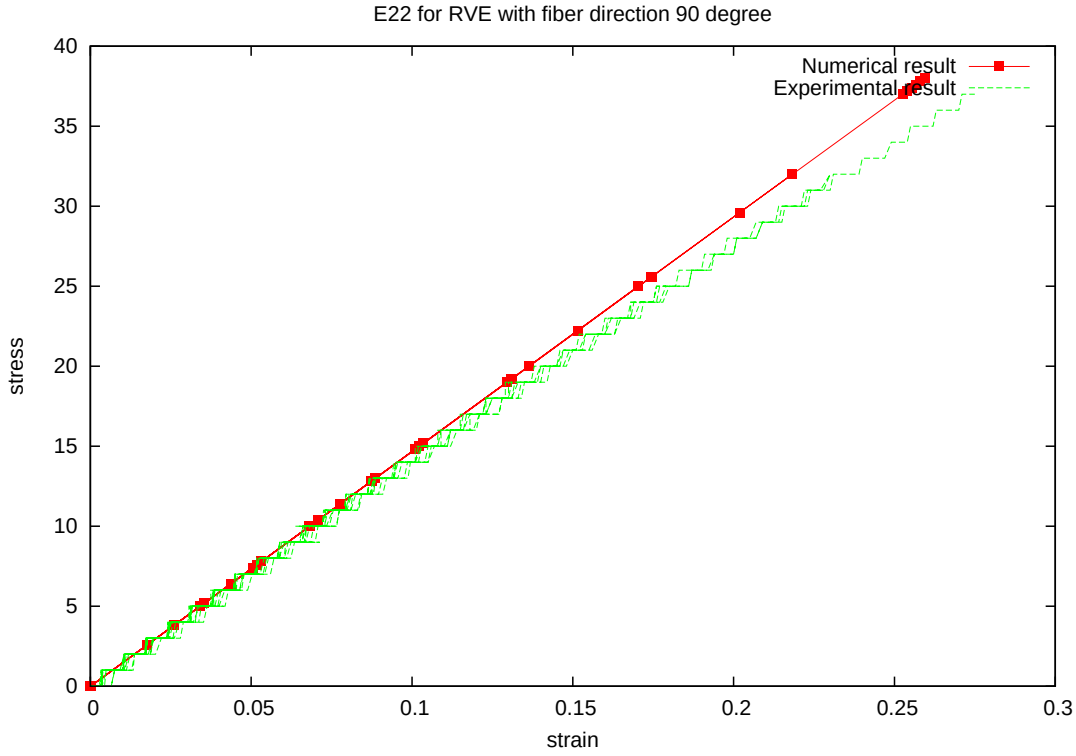


3.2.2 Fiber direction 90°

Use the same RVE as Fig3.4,but apply the macroforce in y direction. E_{22} will be obtained by this test.

	σ_{22}(MPa)	ε_{22}(%)	E_{22}(MPa)
numerical result	38	0.2595	14676.2
Experimental data	37.132	0.275	143479

Table 3.3: Result comparison of transverse Young's Modulus



Transverse tensile test shows similar phenomenon as longitudinal test. The prediction agrees the experimental data well at small strain area, then error grows as the strain increase. And the behavior is still near linear.

3.2.3 Fiber direction 45°

Now we have got the transverse and longitudinal modulus, we are going to design a test for finding out shear modulus. We test composite laminar with fiber direction 45°, as the composite is anisotropic, when we apply transverse load, it will also give shear strain. We can use coordinate transformation to calculate the shear modulus.

The RVE 2.1 used for previous test does not appropriate now, we use RVE below:

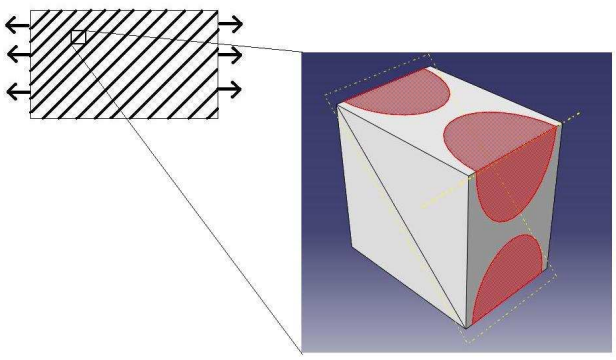


Figure 3.5: RVE for composite laminar with fiber direction 45 °

To get G_{12}, we need to do axes transform for strain and stress.Suppost we have known the stress and strain on xy planes, but we want to stresses acting on planes oriented at θ, We will propose a means to transform the stresses to these new $x'y'$ planes.

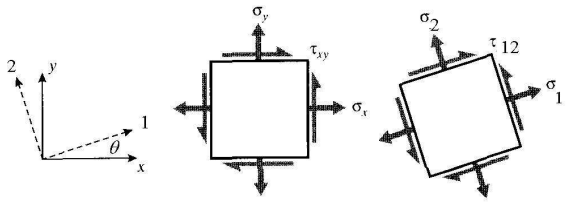


Figure 3.6: Axes transformation for strain and stress

In our case, for the composite laminar as Fig 3.7 below, we know the σ ε in xy direction, but we want to use transformation of axes to get σ ε in fiber direction, which has an angle θ with the original coordinates.

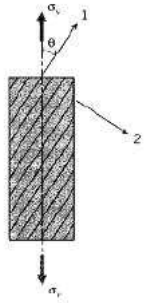


Figure 3.7: Axes transformation for UD composite laminar

We have solution for the problem:

$$\begin{Bmatrix} \sigma_{x'} \\ \sigma_{y'} \\ \tau_{x'y'} \end{Bmatrix} = \begin{pmatrix} c^2 & s^2 & 2sc \\ s^2 & c^2 & -2sc \\ -sc & sc & c^2 - s^2 \end{pmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} \tag{3.1}$$

Where $s = sin\theta, c = cos\theta$, and we set matrix A as the transformation matrix above, we can write transformation as

$$\sigma' = A\sigma \tag{3.2}$$

and for strain transformation

$$\begin{Bmatrix} \varepsilon_{x'} \\ \varepsilon_{y'} \\ \frac{1}{2}\gamma_{x'y'} \end{Bmatrix} = A \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \frac{1}{2}\gamma_{xy} \end{Bmatrix} \tag{3.3}$$

Use the conclusion of axes transformation above, we can calculate the corresponding pressure value from the experiment and apply it on our RVE.When the computing is done, we use transformation to get $\varepsilon_{x'}, \varepsilon_{y'} and \varepsilon_{x'y'}$.

Computational result

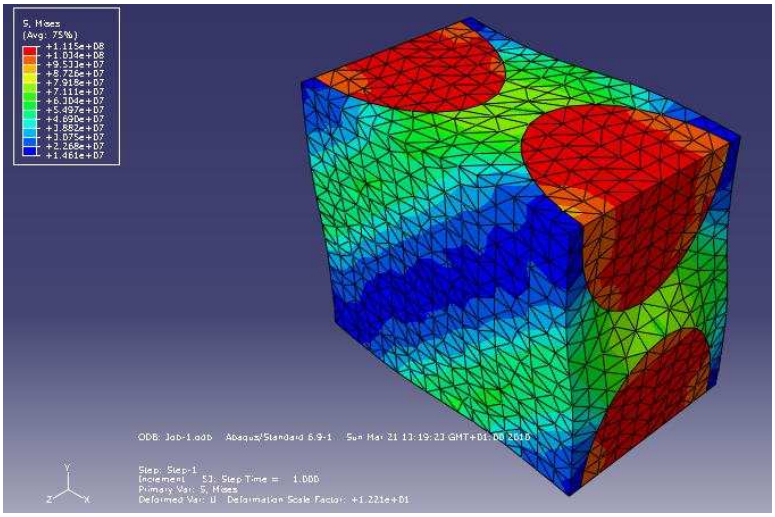


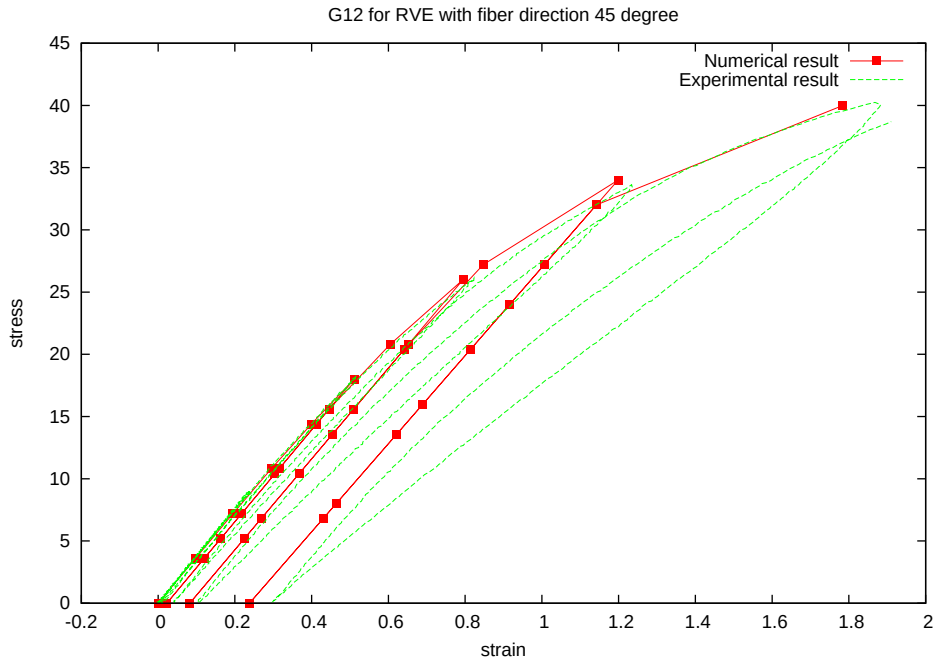
Figure 3.8: Computational result for laminar with 45 °

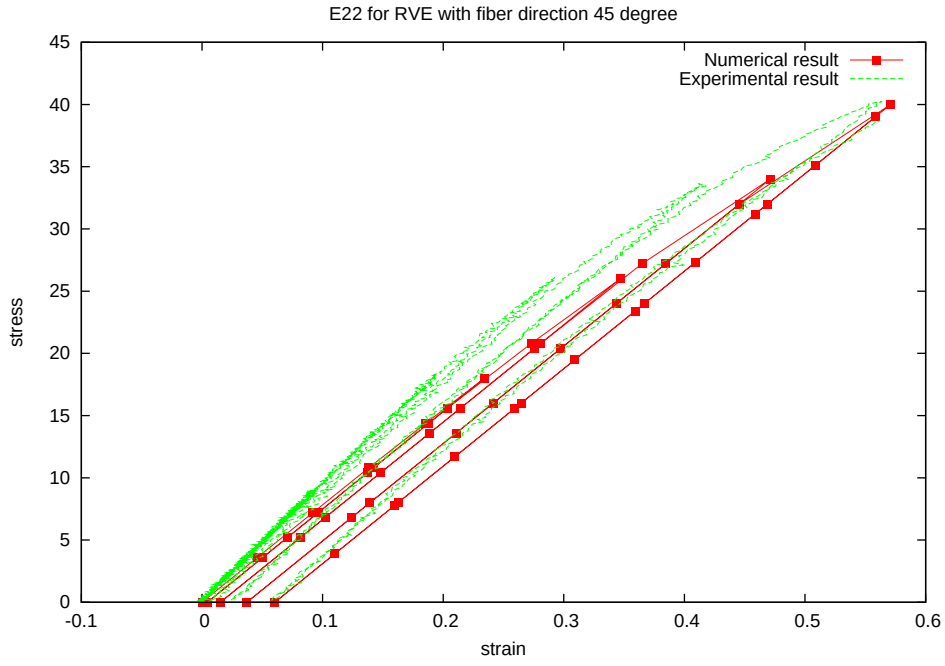
Elastic modulus

	G_{12}(MPa)	E_{22}(MPa)
numerical result	3696.58	7834.466
Experimental data	3755	9618

Table 3.4: Result comparison of shear modulus

For the plasticity, we can compare the result by strain stress curve.





In the result we can find, for shear strain stress curve, the model can match the experiment curve very well at elastic strain and small plastic strain area. but

the error becomes big after that for the model without damage introduction. In this case, damage is a significant parameter to affect the evolution of the material, however, in Abaqus, only linear damage parameter law or exponential damage parameter law is available for the model. In this two damage laws, both require the damage parameter start from 0 and end at 1. And both of them is not for this case.[1]. This problem could be solved by introducing User Subroutine which enable the user to introduce the damage law themselves.

[1]The experiment shows that, for this kind of UD composite, the damage parameter law is linear, but not start from 0 and also not end with 1

Chapter 4

Conclusion

The aim of this paper is to introduce Homogenization method to widely used commercial software Abaqus, to enable it to estimate the overall mechanical behavior of UD composite material.

We first shortly introduce the history of composite material, the feature of composite material and concept of UD composite. Due to the outstanding mechanical performance of composite, it's important to develop a reliable and economic numerical method to estimate the overall mechanical parameters of composite from the component material. A brief review of method of estimation of parameter of composite are given. In this paper, we are going to use homogenization method to evaluate the mechanical property of composite.The key difficults of predicting mechanical parameter of composite with Abaqus are appying periodic boundary conditions and homogenization procedure from microscopic level to macroscopic level.

After that, concepts RVE, periodic boundary conditions which are related with homogenization method are introduced. And some detailed derivation is also presented. In order to valid the homogenization method in FEM, concepts of Scripts interface, features of commercial FEM software Abaqus and Python object oriented programming language are presented. The task of the scripts is to enable Abaqus to apply periodic boundary conditions to the model and execute the homogenization procedure. The algorithm and structure of implementation

using Python in Abaqus are showed in the second chapter. At last part of the article, some numerical example are done by Abaqus with Python scripts. And the numerical result are compared with experimental data to verify the validation of the code.

According to the comparison of numerical result and experimental data. We can conclude that, predicting the mechanical parameter of UD composite materiel by homogenization method is possible in Abaqus. It could be done by using Python scripts in Abaqus. Some proof has been done to verify the correctness theoretically. And numerical test showed the reliability of this scripts as well as the validity. So we can conclude that, using Python scripts, it's possible to predict the mechanical parameter of elasticity and plasticity of UD composite material safely. But the error increase as the strain grows because of the lack of consideration of interaction between fiber and matrix as well as appropriate damage law during the computing.And the existing scripts is only valid for simple geometry model. In the future research, dealing with complex geometry model and including the interaction between fiber and matrix could be considered. And using User Subroutine to introduce appropriate damage law could also be done to improve the model.

Appendix A

Python Scripts

```
from abaqus import* from abaqusConstants import *

ModelName='Model-1' # Input data InstanceName='Part-1-1'
Noeuds=mdb.models[ModelName].rootAssembly.
instances[InstanceName].nodes

# Get the dimensions of the square model # note : next step => try
to find the type of VER and automatise the process
Nb-noeuds=len(Noeuds) Max-x=-1000 Max-y=-1000 Min-x=1000 Min-y=1000
Max-z=-1000 Min-z=1000 radius=100 cen=0

for i in range(Nb-noeuds):
    Cx= Noeuds[i].coordinates[0]
    Cy= Noeuds[i].coordinates[1]
    Cz= Noeuds[i].coordinates[2]
    if Max-x<=Cx:
        Max-x=Cx
    if Cx <=Min-x:
        Min-x=Cx
    if Max-y<=Cy:
        Max-y=Cy
    if Cy <=Min-y:
        Min-y=Cy
```

```
    if Max-z<=Cz:
        Max-z=Cz
    if Cz <=Min-z:
        Min-z=Cz

lx=Max-x-Min-x  ly=Max-y-Min-y
lz=Max-z-Min-z  lxc=(Max-x+Min-x)/2
lyc=(Max-y+Min-y)/2  lzc=(Max-z+Min-z)/2

for i in range(Nb-noeuds):
    Cx= Noeuds[i].coordinates[0]
    Cy= Noeuds[i].coordinates[1]
    Cz= Noeuds[i].coordinates[2]
    if (Cx-lxc)*(Cx-lxc)+(Cy-lyc)*(Cy-lyc)+(Cz-lzc)*(Cz-lzc)
    <=radius :
    cen=i
    radius=(Cx-lxc)*(Cx-lxc)+(Cy-lyc)*(Cy-lyc)+(Cz-lzc)*(Cz-lzc)



# Storing of nodes belonging to the external Faces # (Used for the
treatment of constraints equation ; important note : this is not
node label stored but array index of Noeuds) Set-xDroite=[]
Set-xGauche=[] Set-yHaut=[] Set-yBas=[] Set-zDroite=[]
Set-zGauche=[]

lSet-xDroite=[] lSet-xGauche=[] lSet-yHaut=[] lSet-yBas=[]
lSet-zDroite=[] lSet-zGauche=[]

for i in range(Nb-noeuds):
    Cx= Noeuds[i].coordinates[0]
    Cy= Noeuds[i].coordinates[1]
    Cz= Noeuds[i].coordinates[2]
# Only strore the nodes includes in the constrains equations
    if (Max-x==Cx) and (Cy!=Max-y) and (not ((Cy==Min-y)
    and (Cz==Max-z))):
        Set-xDroite.append(i)
        lSet-xDroite.append(i+1)
    if (Cx==Min-x) and (Cy!=Max-y) and (not ((Cy==Min-y)
    and (Cz==Max-z))):
```

```
        Set-xGauche.append(i)
        lSet-xGauche.append(i+1)
    if (Max-y==Cy) and ((Cz!=Max-z) or ((Cz==Max-z) and
    (Cx==Min-x)) or ((Cz==Max-z) and (Cx==Max-x)))
        Set-yHaut.append(i)
        lSet-yHaut.append(i+1)
    if (Cy==Min-y) and ((Cz!=Max-z) or ((Cz==Max-z) and
    (Cx==Min-x)) or ((Cz==Max-z) and (Cx==Max-x))):
        Set-yBas.append(i)
        lSet-yBas.append(i+1)
    if (Cz==Max-z) and ((Cx!=Max-x) or ((Cx==Max-x) and
    (Cy==Min-y))) and (not ((Cx==Min-x) and (Cy==Max-y))):
        Set-zDroite.append(i)
        lSet-zDroite.append(i+1)
    if (Cz==Min-z) and ((Cx!=Max-x) or ((Cx==Max-x) and
    (Cy==Min-y))) and (not ((Cx==Min-x) and (Cy==Max-y))):
        Set-zGauche.append(i)
        lSet-zGauche.append(i+1)
    if Cx*Cx+Cy*Cy+(Cz-1)*(Cz-1)<=radius :
    cen=i
    radius=(Cx-lxc)*(Cx-lxc)+(Cy-lyc)*(Cy-lyc)+(Cz-lzc)*(Cz-lzc)



# Creating the three reference points (macro nodes)
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.2,0.1,1.0))
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.2,0.0,1.0))
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.2,-0.1,1.0))
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.5,0.1,1.0))
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.5,0.0,1.0))
mdb.models[ModelName].rootAssembly.ReferencePoint
(point=(1.5,-0.1,1.0))



# Built the Sets of every faces. It's convenience for checking
```

```
mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setxd',nodeLabels=((InstanceName, (2,3,5,7)), (InstanceName,
lSet-xDroite)))

mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setxg',nodeLabels=((InstanceName, (2,3,5,7)),(InstanceName,
lSet-xGauche)))

mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setyh',nodeLabels=((InstanceName, (2,3,5,7)),(InstanceName,
lSet-yHaut)))

mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setyb',nodeLabels=((InstanceName, (2,3,5,7)),(InstanceName,
lSet-yBas)))

mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setzd',nodeLabels=((InstanceName, (2,3,5,7)),(InstanceName,
lSet-zDroite)))

mdb.models[ModelName].rootAssembly.SetFromNodeLabels(name=
'setzg',nodeLabels=((InstanceName, (2,3,5,7)),(InstanceName,
lSet-zGauche)))
```

```
# Creating unique set for the previous identified nodes
Set-name='Set-N'


for i in range(len(Set-xDroite)):
        mdb.models[ModelName].rootAssembly.Set(name=
        Set-name+'xD'+str(i),nodes=
        Noeuds[Set-xDroite[i]:Set-xDroite[i]+1])



for i in range(len(Set-xGauche)):
    mdb.models[ModelName].rootAssembly.Set(name=
    Set-name+'xG'+str(i),nodes=
    Noeuds[Set-xGauche[i]:Set-xGauche[i]+1])


for i in range(len(Set-yHaut)):
    mdb.models[ModelName].rootAssembly.Set(name=
    Set-name+'yH'+str(i),nodes=
    Noeuds[Set-yHaut[i]:Set-yHaut[i]+1])

for i in range(len(Set-yBas)):
    mdb.models[ModelName].rootAssembly.Set(name=
    Set-name+'yB'+str(i),nodes=
    Noeuds[Set-yBas[i]:Set-yBas[i]+1])


for i in range(len(Set-zDroite)):
        mdb.models[ModelName].rootAssembly.Set(name=
        Set-name+'zD'+str(i),nodes=
        Noeuds[Set-zDroite[i]:Set-zDroite[i]+1])


for i in range(len(Set-zGauche)):
    mdb.models[ModelName].rootAssembly.Set(name=
    Set-name+'zG'+str(i),nodes=
    Noeuds[Set-zGauche[i]:Set-zGauche[i]+1])
```

```
kf=mdb.models['Model-1'].rootAssembly.referencePoints.items()
ks=len(kf) k=kf[ks-1][0] r1 =
mdb.models[ModelName].rootAssembly.referencePoints
refPoints1=(r1[k], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='E11')


refPoints1=(r1[k+1], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='E22')


refPoints1=(r1[k+2], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='GAMMA12')


refPoints1=(r1[k+3], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='E33')


refPoints1=(r1[k+4], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='GAMMA13')


refPoints1=(r1[k+5], )
mdb.models[ModelName].rootAssembly.Set(referencePoints=refPoints1,
name='GAMMA23')


mdb.models[ModelName].rootAssembly.editNode(nodes=Noeuds[cen],
coordinate1=lxc,coordinate2=lyc,
coordinate3=lzc,projectToGeometry=ON)
mdb.models[ModelName].rootAssembly.Set(nodes=Noeuds[cen:cen+1],
name='Centre')


torl=0.01
```

```
# Creating the constraint equations Constraint-name='Eq-x'

s=0 for i in range(len(Set-xDroite)):
    coord-yd=Noeuds[Set-xDroite[i]].coordinates[1]
    coord-zd=Noeuds[Set-xDroite[i]].coordinates[2]
    for j in range(len(Set-xGauche)):
        coord-yg= Noeuds[Set-xGauche[j]].coordinates[1]
        coord-zg= Noeuds[Set-xGauche[j]].coordinates[2]
        if (abs(coord-yd-coord-yg)<torl) and
         (abs(coord-zd-coord-zg)<torl):
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s),terms=((-1.0,
            Set-name+'xD'+str(i),1),
            (1.0,Set-name+'xG'+str(j),1),(lx,'E11',1)))
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s+1),terms=((1.0,
            Set-name+'xD'+str(i),2),(-1.0,
            Set-name+'xG'+str(j),2),(-lx/2,'GAMMA12',1)))
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s+2),terms=((-1.0,
             Set-name+'xD'+str(i),3),
            (1.0,Set-name+'xG'+str(j),3),(lx/2,'GAMMA13',1)))
            s=s+3


Constraint-name='Eq-y' s=0 for i in range(len(Set-yHaut)):
    coord-xh=Noeuds[Set-yHaut[i]].coordinates[0]
    coord-zh=Noeuds[Set-yHaut[i]].coordinates[2]
    for j in range(len(Set-yBas)):
        coord-xb= Noeuds[Set-yBas[j]].coordinates[0]
        coord-zb= Noeuds[Set-yBas[j]].coordinates[2]
        if (abs(coord-xh-coord-xb)<torl) and
        (abs(coord-zh- coord-zb)<torl):
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s),terms=((1.0,
            Set-name+'yH'+str(i),2),(-1.0,
            Set-name+'yB'+str(j),2),(-ly,'E22',1)))
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s+1),terms=((1.0,
            Set-name+'yH'+str(i),1),(-1.0,
```

```
                Set-name+'yB'+str(j),1),(-ly/2,'GAMMA12',1)))
                mdb.models[ModelName].Equation(name=
                Constraint-name+str(s+2),terms=((1.0,
                Set-name+'yH'+str(i),3),(-1.0,
                Set-name+'yB'+str(j),3),(-ly/2,'GAMMA23',1)))
                s=s+3


Constraint-name='Eq-z' s=0 for i in range(len(Set-zDroite)):
    coord-xd=Noeuds[Set-zDroite[i]].coordinates[0]
    coord-yd=Noeuds[Set-zDroite[i]].coordinates[1]
    for j in range(len(Set-zGauche)):
        coord-xg= Noeuds[Set-zGauche[j]].coordinates[0]
        coord-yg= Noeuds[Set-zGauche[j]].coordinates[1]
        if (abs(coord-xd-coord-xg)<torl) and
        (abs(coord-yd-coord-yg)<torl):
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s),terms=((-1.0,
            Set-name+'zD'+str(i),1),(1.0,Set-name+'zG'+str(j),1),
            (lz/2,'GAMMA13',1)))
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s+1),terms=((1.0,
            Set-name+'zD'+str(i),2),(-1.0,Set-name+'zG'+str(j),2),
            (-lz/2,'GAMMA23',1)))
            mdb.models[ModelName].Equation(name=
            Constraint-name+str(s+2),terms=((-1.0,
            Set-name+'zD'+str(i),3),(1.0, Set-name+'zG'+str(j),3),
            (lz,'E33',1)))
            s=s+3
```

```
r1 = mdb.models[ModelName].rootAssembly.referencePoints
refPoints1=(r1[k], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-E11',
```

```
createStepName='Step-1',region=region, u1=0.0,  fixed=OFF,
fieldName='', localCsys=None)



refPoints1=(r1[k+1], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-E22',
createStepName='Step-1',     region=region, u1=0.0,
fixed=OFF,
fieldName='', localCsys=None)



refPoints1=(r1[k+2], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-GAMMA12',
createStepName='Step-1',     region=region, u1=1.0, fixed=OFF,
fieldName='', localCsys=None)


refPoints1=(r1[k+3], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-E33',
createStepName='Step-1',     region=region, u1=0.0,
fixed=OFF,
fieldName='', localCsys=None)



refPoints1=(r1[k+4], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-GAMMA13',
createStepName='Step-1',     region=region, u1=0.0, fixed=OFF,
fieldName='', localCsys=None)



refPoints1=(r1[k+5], ) region = refPoints1

mdb.models[ModelName].DisplacementBC(name='BC-GAMMA23',
```

```
createStepName='Step-1',      region=region, u1=0.0,
fixed=OFF,
fieldName='', localCsys=None)


region=mdb.models[ModelName].rootAssembly.sets['Centre']
mdb.models[ModelName].DisplacementBC(name='BC-centre',
createStepName='Step-1',      region=region, u1=0.0,
u2=0.0, u3=0.0,
fixed=OFF,     fieldName='', localCsys=None)
```

References

[1] Papanicoulau G Benssousan A, Lions JL. *Asymptotic analysis for periodic structures*. North-Holland, 1978.

[2] C.T.Sun and R.S.Vaidya. Prediction of composite properties from a representative volume element. *Composites Science and Technology*, 56:171–179, 1996.

[3] Daniel Gay. *Composite materials-Design and Application*. CRC Press LLC, 2003.

[4] B.W Hashin, Z Rosen. The elastic moduli of fiber-reinforced materials. *ASME J. Appl. Mech*, 31:223–32, 1964.

[5] Z. Hashin. Analysis of composite materials-a survey. *J.Appl. Mech*, 50:481–505, 1983.

[6] M.Pandheeradi J.Fish, K.Shek and M.S.Shephard. Computational plasticity for composite structures based on mathematical homogenization: Theory and practice. *Comp. Meth. Appl. Mech*, 148:53–73, 1997.

[7] J.M.Guedes and N.Kikuchi. Preprocessing and postprocessing for materials based on the homogenization method with adaptive finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 83:143–198, 1990.

[8] J.C.Michel H.Moulinec P.Suquet. Effective properties of composite materials with periodic microstructure: a computational approach. *Comput.Methods Appl Mech Engrg*, 172:109–143, 1999.

[9] Simulia. *Abaqus Scripting User's Manual*. Simulia, 2008.

[10] K Terada and N Kikuchi. Nonlinear homogenization method for practical applications. *Computational Mechanics in Micromechanics, ASME, New York*, AMD-212/MD-62:1–16, 1995.

[11] F.P.T.Baaijens V.Kouznetsova, W.A.M.Brekelmans. An approach to micro-macro modeling of heterogeneous materials. *Computational Mechanics*, 27:37–48, 2001.

[12] M.B Whitney, J.M. Riley. Elastic elastic properties of fiber reinforced composite materials. *AIAA J.*, 4:1537–42, 1966.

Acknowledgements

I would like to extend my sincere gratitude to my supervisor, Mr. Patrick ROZYCKI, for his outstanding instructive advice and remarkable suggestions on my thesis. I am deeply grateful of his help in the completion of this thesis.

I am also deeply indebted to Mr.Laurent GORNET who also has put considerable time and effort into my thesis work.