Cronfa
Swansea University's Research Repository

Swansea University
Prifysgol Abertawe

Multi-Scale Analysis of Reinforced Composites

MALEBOGO TSHEKO

Swansea University

Submitted to Swansea University in fulfillment of the requirements for the Degree of

Ph.D

September 2024

Copyright: The author, Malebogo Tsheko, 2024

Summary

The thesis centres on multi-scale modelling of heterogeneous solids, where the macroscopic behaviour is intricately linked to the microscopic structure. To manage the substantial memory and computational power demands of multi-scale modelling, a discrete Representative Volume Element (RVE) boundary approach based on a finite element model of the microstructure is employed. RVE refers to a small unit of a material that represents the whole material in terms of structure and properties. A new homogenisation method is explored in this research, and subsequently, the results obtained from this approach are rigorously compared to analytical methods using multiple verifying examples. This comparative analysis provides insights into the accuracy and reliability of the multi-scale modelling technique in predicting material behaviours across different scales.

Homogenous, isotropic and transversely isotropic 3D solids are investigated. Then, a homogenisation tool employing the least squares method is used to determine the effective elastic properties of the composite. A more advanced tool, a Neural Network, utilises data from the least squares method to predict the elastic properties of the composite without the necessity of re-modelling the RVE and conducting additional tests.

The trained neural network, which utilises data derived from least squared method plays a crucial role in predicting elastic properties of composites with very high accuracy. By integrating the neural network's predictions with an optimisation algorithm, they can effectively tailor materials to meet specific criteria such as cost-efficiency and lightweight construction. This integrated approach not only enhances design precision but also reduces the need for extensive re-modelling of the RVE and repetitive testing. Ultimately, it fosters the development of innovative materials that strike an optimal balance between performance and resource utilisation in various engineering applications.

Declarations

1. This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

   Signed ....[redacted]...... (candidate)

   Date ....13/10/2025...........................

2. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

   Signed ....[redacted]... (candidate)

   Date......................13/10/2025...........................

3. I hereby give consent for my thesis, if accepted, to be available for electronic sharing

   Signed ....[redacted]......... (candidate)

   Date ....13/10/2025...........................

4. The University's ethical procedures have been followed and, where appropriate, that ethical approval has been granted.

   Signed ....[redacted]...... (candidate)

   Date ....13/10/2025...........................

Table of Contents

Acknowledgments

This research was made possible through the generous funding provided by the Government of Botswana, for which I am sincerely grateful. I would like to express my deepest appreciation to my supervisors, **Prof. W. Dettmer** and **Prof. D. Perić**, whose guidance, help and support have been valuable throughout the course of this project. Their insightful feedback and encouragement were crucial in shaping this work.

I would also like to thank **Prof. W. Dettmer**, for sharing his code ,his friendship ,assistance and support during difficult times. His contributions and efforts were instrumental in the completion of this thesis.

I would also like to thank E. Alhayki and E. M. Zvala for their help and for providing valuable insights when necessary.

Chapter 1

Introduction

Reinforced composite material is defined as a material made from two or more components with different physical or mechanical properties. Usually the behaviour or the properties of the composite is improved as compared to the individual components. The constituents combine to produce a material with better characteristics such as superior strength, enhanced durability or lighter weight. This has emerged as a significant innovation in material science.

According to Ngo [34], Mar-Bal [29], Johnson [24], CompositesLab [6] reinforced composites date back to ancient times as back as before 25 B.C where people used lime and mortars for construction. Mar-Bal [29] indicated that Egypytians and Mesopotamians builders used straws to reinforce mud and bricks, pottery and boats. The introduction of straws to mud and bricks improved the mechanical properties of the mixture compared to the individual components. This helped them create strong and durable buildings. Another evidence of composites existence was the use of glued wood at different angles to create plywood. Since ancient times builders and engineers continued to improve composites for more complicated usage.

CompositesLab [6] and Mar-Bal [29] indicated that in the $1200A.D$ Mongols developed their composite bows from bones, horns, bamboo, cattle tendon, wrapped with silk and sealed it with pine resin. The bows were powerful, accurate and could hit far targets during military operations. From these bows we learn that more materials properties such as overall weight, without compromising the strength, were considered during design.

In the $1800's$ chemical revolutions changed composite material. Resins were transformed from liquid to solid state through a process of polymerisation which lead to plastic developments in $1900's$. Plastics such as vinyl, polystyrene and polyesters were better than single resins derived from nature. Plastic on its own does not have the strength to be used for other structural purposes as a results reinforcement was required for strength which led to Owens Corning introducing glass fibres in 1935 which were strong and light in weight. This resulted in the birth of the Fibre Reinforced Polymer (FRP) industry. The industry continued to grow with pultrusion, vacuum bag moulding and filament winding. Morden day carbon fibres were also developed in $1950s$ which revolutionised aerospace, automotive, sporting goods and consumer goods.

Composite material continued to grow with developments in ultra-high molecular weight polyethylene and other advances in aerospace components, structures, motor assemblies, electrical braks and other applications. This lead to more cost effective replacements of traditional materials like metal. Advanced manufacturing techniques, where two or more manufacturing process are combined such as computer numerical control and additive processes, may provide solutions where high accuracy and machine speed are required Patel and Kilic [37]. There are also other developments in nano-structured composites where nano-materials are incorporated into composites to enhance mechanical properties Dong [10], Ghamsari [15]. Smart multifunctional composites such as material with sensors embedded or self-healing capabilities offer wide range of functions and find applications in aerospace, automotive and structural engineering as explained by Dulal et al. [11], Therriault and Farahani [48]. Other recent studies involve bio-based and sustainable composites which are renewable and biodegradable composites. The field of composite materials continue to grow.

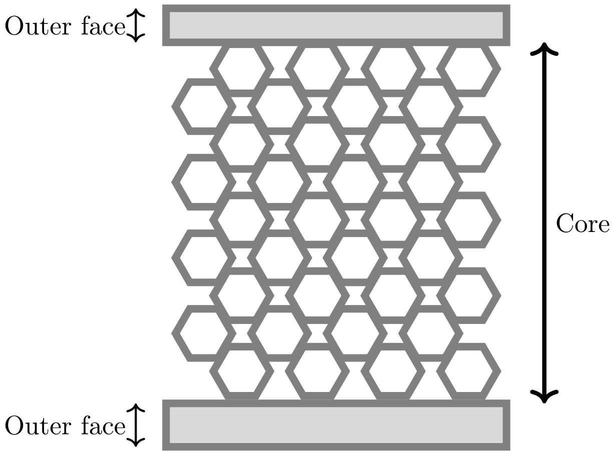


Figure 1.1: Honeycomb core for Sandwich

Sandwich panels are structural materials used in various industries where strength to weight ratio , stiffness and durability are important. This material is made of the outer face sheets and the light core structure normally with a honeycomb pattern as shown by Figure 1.1. The face/outer material can be an aluminium alloy, reinforced plastic, heat resistant material. Materials and geometry of the core may vary with the honeycomb as the most familiar type of the core Petrovic and Lazarevic [39]. The core material and cell size can be varied to tailor the panel's properties to specific needs. The core mainly separates and keeps external faces at a given distance from each other in order to resist buckling. This gives flexibility to use other material as sandwich cores for heat resistance, insulation and many more.

There are different sandwich core designs in sandwich panels, these cores may include but are not limited to wood core, foam cores, honeycomb cores and corrugated cores Ramly et al. [41]. These different cores make them different in terms of their overall mechanical behaviour. Sandwich composites are advantageous as they are excellent in abosorbing acoustic sound making it useful to isolate sound for passengers and cabin comfort Ramly et al. [41]. These sandwich composite panels are less susceptible to corrosion and fatigue compared to metallic structures, resulting in reduced maintenance requirements. Some of the composite material designed in this research can be used for sound proofing as inspired by the honeycomb structures' ability to control composite material properties from the honey comb parameters Leone et al. [25], Petrovic and Lazarevic [39] and Essaadaoui et al. [12] .

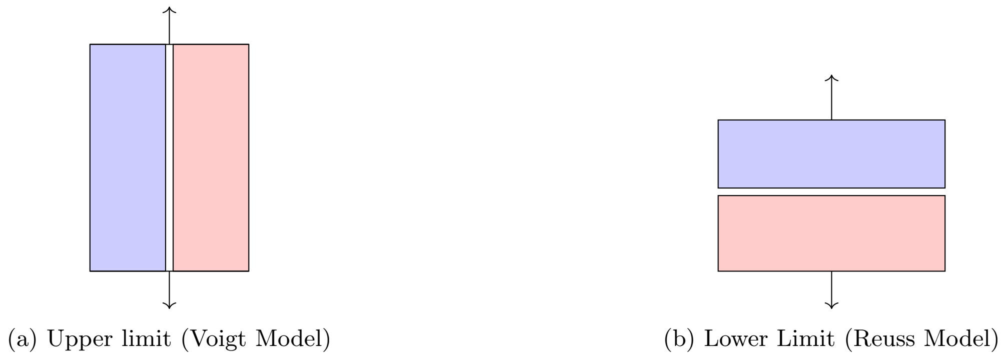
The impact of composite material is profound across industries such us aerospace where lightweight materials are used to reduce aircraft weights and improve fuel efficiency. For marine and construction industries composites are used for corrosion resistance and higher strength to weight ratio of composites. From the economical view point optimisation of fibre reinforced composites reduces manufacturing costs and the enhance sustainability through improved efficiency and reduced carbon footprint compared to unoptimised composites.

1.1 Motivation

This research is driven by motivation to advance composite material technology by reducing experimental iterations and minimising scraps, thereby lowering environmental impact. Optimisation techniques lead to developments of superior composites which are beneficial in enhancing fuel efficiency, sustainable construction and reducing environmental impacts.

This study is motivated by the principle of the rule of mixtures, which indicates that when two materials, one possessing superior properties, are combined, the resulting composite's properties will fall within the bounds of the two constituent materials, as further detailed by Figure 1.2.

When the composite micro-structure's constituents (matrix and inclusions) mechanical properties are closely matched, the range for potential improvements in the composite material becomes narrower. This reduced scope for development is illustrated in Figure 1.3.


(a) Upper limit (Voigt Model)

(b) Lower Limit (Reuss Model)

Figure 1.2: Parallel and series model for the elastic property limits of a composites material

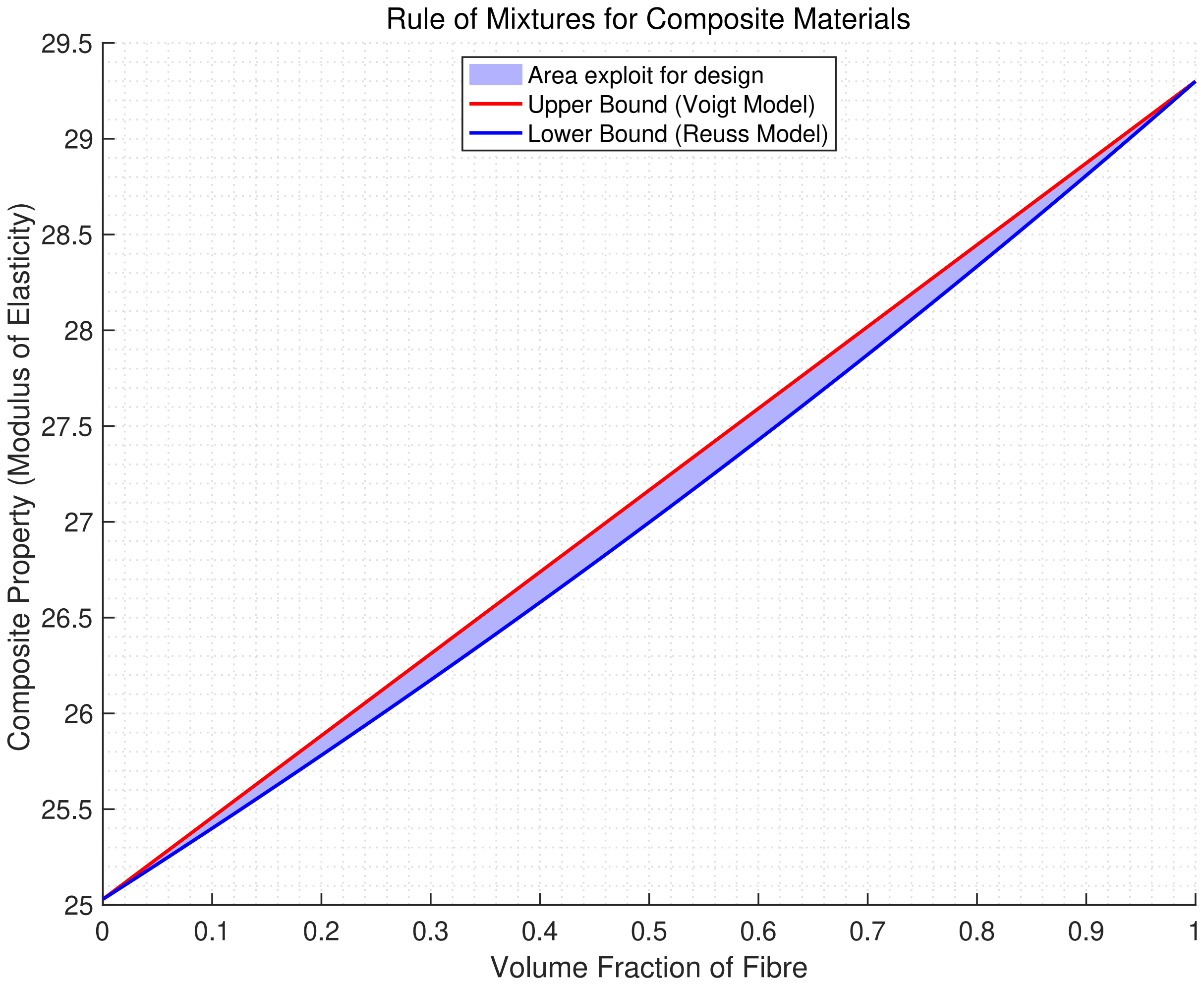


Figure 1.3: Fibre and matrix with Young's Modulus $29GPa$ and $25GPa$ respectively

A significant difference in mechanical properties between the matrix and fibres expands the scope for development, as demonstrated by the Figure 1.4. This disparity drives the study, as the fibres under consideration possess substantially better mechanical properties compared to the matrix. Introducing voids and varying the fibre volume ratio, aspect ratio, and fibre spacing at the microscopic level can result in the development of superior composites that optimise resource utilisation. This gives scope for optimisation of the micro-structure to get design parameters such as fibre volume ratios and aspect ratios.

The work is supported by a grant from the government of Botswana which is a developing country and looking for greener and sustainable industrial development. Botswana takes pride in its environment as

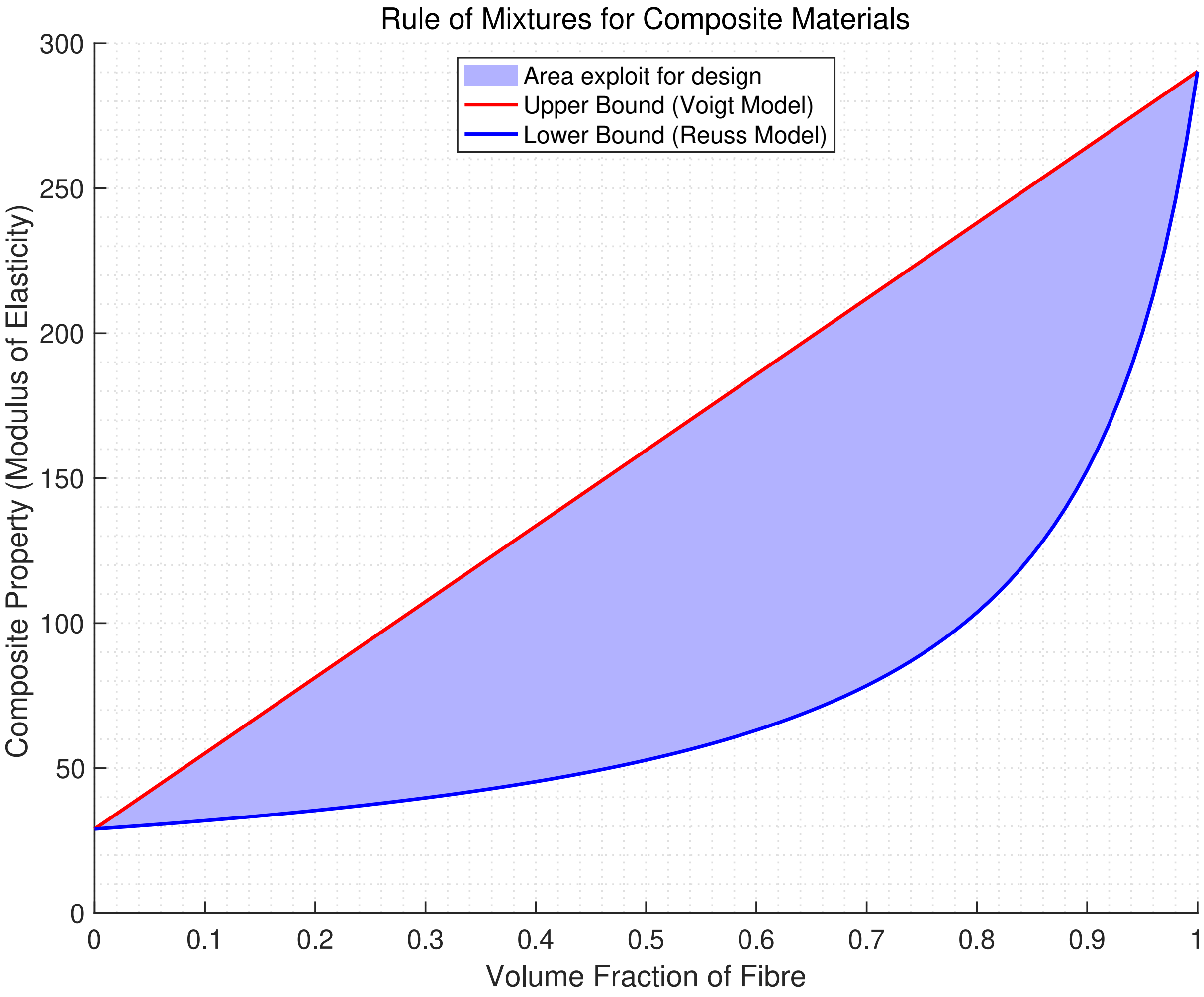


Figure 1.4: Fibre and matrix with Young's Modulus $290GPa$ and $29GPa$ respectively

it is dependent on tourism and wildlife hence the interest in sustainable development. As new infrastructures are built better techniques have to be used for sustainable infrastructure development. The primary focus of this research is computational modelling and analysis to predict and optimise micro-structure in line with development of lightweight, cost effective and yet strong composites for different applications. The applications include but not limited to sandwich cores, insulators, partition walls, sound proofing and more. This study involves setting objectives and employing advanced optimisation techniques to the micro-structure to achieve better materials. This research is computer based and is conducted in Swansea University facility equipped with the tools for computational modelling. Working from this facility with colleagues and supervisors fosters innovation and facilitate the exchange of ideas.

The study faces several technical challenges, including modelling complex meshes, handling extensive computational workloads and converting natural problem into a numerical objective function for optimisation. Composite material with desired properties and yet lightweight or cheaper costs will contribute to the composite material industry by providing practical solutions. The long term goal is to further explore sustainable material and enhance the performance of composites to drive continuous innovation in the field.

1.2 Literature Review

In manufacturing the most studied voids are defects often from processing of fibre reinforced composite. Voids formation, causes and control differ for all manufacturing techniques due to thermodynamic and rheological phenomena happening in the processes of composites manufacturing. Mehdikhani et al. [32] shows that voids may be due to mechanical air entrapment during resin flow, gases from chemical reactions and nucleation of dissolved gases. Voids can be formed at macro or micro levels where micro voids are between fibres and macro voids are observable to the naked eye. Macro void refers to a zone that has not been inmpregnated by macroscopic resin yet, this may be due to irregular flow patterns or irregular permeability of the preform or presence of insects or ribs in the mold Mehdikhani et al. [32]. Voids can be characterised by density characterisation that is void volume fraction, microscopy by means of optical and microscopic techniques, ultrasonic testing where reflecting/scattering of ultrasonic pulse depends on the void and fibre volume fraction and X-ray micro-CT. Furthermore voids can be characterised by shape, size, location ,void content and distributions. Void content is most commonly used and can be measured using any of the methods above. The void content in composite materials can be controlled through means of controlled temperature, pressure and vacuum levels Mehdikhani et al. [32].

Discontinuous fibres Wang et al. [51] have superior mechanical properties compared to continuous fibre hence promoting their use in engineering applications. Composites are also prone to process - induced defects hence affecting their mechanical properties. Other factors that influences composite properties are the fibre length, fibre shape and fibre orientation. Mechanical properties can be tailored by controlling fibre direction. Al-Maharma et al. [1] carried out a research on the effects of porosity on the mechanical properties of additively manufactured components and how undesired effects of voids can be removed. For engineered porosity that is where voids are introduced intentionally and are controlled for a particular function, Al-Maharma et al. [1] indicated that this porosity's physical properties including size, dimension, shape can be controlled through fabrication processes. Al-Maharma et al. [1] also explained that porosity in lattice structures and scaffolds can be tailored to certain mechanical properties and biodegradation behaviour required in biological field for carrying nutrients delivery which decreases overall weight without affecting the loading capabilities.

Micro-porosity can be spherical from powder containing entrapped gas or keyholes from wrong processing parameters as explained by Al-Maharma et al. [1] due to incomplete melting induced porosity, lack of fusion with unmelted particles inside large irregular pores and cracks Sola and Nouri [45]. There are many other process induced pores/defects as described in details with their solution [45]. Changing the structure, bulk and cellular materials, the mechanical properties can be can be achieved through graded pores and units cells. For larger porosity-graded light weight lattices finite element is used for design and analysis Terriault and Brailovski [47].

Carrera et al. [5] evaluated the influence of voids on a 3D representative volume element (RVE) of fibre-reinforced polymer composites and concluded that void arrangement influences the stress fields. That is, a cluster of voids results in high stress mean values. Carrera et al. [5] also found out that although void distribution influence mechanical properties void content is the fundamental property to consider independent of void arrangement. Jiang et al. [23] investigated the effects of voids on carbon fibre fabric reinforcement and found out that flexural strength and modulus decreases by $23MPa$ and $2.17GPa$ for each 1% increase in void fraction respectively. That is flexural properties reduces significantly with small amount of void content increment.

Other research have been also carried out to develop longer discontinuous carbon fibres for higher performance Yang et al. [52]. Pressure and temperature can be controlled and to remove unwanted porosity Li et al. [26].

Some research have been done on cheaper and environmentally friendly composites made from natural fibres. Venkatarajan et al. [49] investigated mechanical properties of natural cellulose fibre reinforced polymer composite and concluded that these fibres may give comparable property levels and they are freely available at lower costs saving industries such as plastic ,packaging and automotive. Although they may have lower mechanical properties they are encouraged by cost reduction and environmental conditions. This type of composites may also have other attributes such as chemical resistance. Jawaid et al. [22] studied chemical resistance of, void content and tensile properties of oil palm/jute fibre reinforced polymer hybrid composites. From these experiments it was observed that matrix and all the composites showed better chemical resistance and better adhesion to the matrix.

Praveena et al. [40] studied experimentally density and volume fraction of voids and mechanical characteristics of Areca nut leaf sheath fibre reinforcement and found out that efficiency of composites are considerably affected by these voids. High void content leads to lower fatigue resistance and higher chance of penetration and weathering when subjected to moisture content. Depending on the matrix replaced by the fibres, composites are found to substitute traditional metals for their limited densities. This Areca composite can be used as a construction material for automotive, marine, etc where high stress with less weight is required.

Academic advancement in composite materials involves multi-scale and hierarchical modelling, which couples micro-scale fibre–matrix interactions with meso-scale damage evolution, enabling more accurate life performance predictions and estimations Sayam et al. [43]. Data-driven machine learning techniques are increasingly being used to accelerate the determination of material parameters. Beyond modelling and monitoring, academic research has also made significant progress in failure analysis and digital integration. Considerable work has been done on natural fibre composites to promote sustainable alternatives, although the long-term durability of these materials remains in question. Additionally, ongoing studies are providing unprecedented insights into failure patterns through in-situ imaging techniques Ghadarah [14]. With the integration of digital tools and machine learning frameworks, academic approaches are increasingly aligning with industrial needs.

Industrial adoption of composites has primarily focused on improving weight-to-performance ratios. This trend is especially evident in industries such as aerospace and wind turbine manufacturing, where product quality, certification, and durability are critical. The automotive sector also utilises composites, particularly where impact resistance and tolerance are essential. Fuel efficiency and processing costs can be optimised through careful selection of composite materials. Case studies conducted Hamzat et al. [18] show that the choice of composites in various industries also depends on factors such as supply chain considerations, maintenance requirements, and long-term durability.

Industries are increasingly focusing on the sustainability of composite materials. There has been some use of bio-based resins, which have shown promising results in reducing the overall carbon footprint. However, translating these lab-scale recyclable processes into industrial-scale applications presents challenges, particularly in terms of economic feasibility and alignment with industry standards Maiti et al. [28].

Even though extensive research is ongoing, a significant gap remains between academic studies and industrial application. Key challenges include the validation of multiscale models, full-scale component testing under real-world conditions, addressing manufacturing variabilities, and integrating structural health monitoring systems into certified maintenance protocols. Bridging this gap requires controlled experiments and pilot-scale industrial trials Hassani et al. [19].

This thesis contributes to the feasibility assessment of multi-scale predictive framework for fibre-reinforced composites, supporting both material design and practical structural applications. In particular, it evaluates how variations in fibre and matrix properties influence the overall mechanical behaviour, thereby providing guidance for experimental validation and industrial design. This approach enables the development of optimised materials and structural designs, improving manufacturability and reducing reliance on costly physical prototyping.

To strengthen industrial relevance, future work should incorporate life cycle analysis and recyclability when proposing new materials, and quantify uncertainties arising from process variability to provide robust design guidance. These steps are essential for bridging the gap between academic research and industrial practice.

1.3 Methodology

This research study is a computer-based modelling and analysis of reinforced composites to optimise the micro-structure. The study creates RVE meshes using software known as `Gmsh`, which is a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. The quality of these meshes is controlled and tailored to achieve the desired accuracy in the results. This approach allows for more precise simulations as a contribution towards the development of advanced composite materials. A block of a homogenous material, a cubic block with spherical inclusions, cubic block with continuous fibres and one with discontinuous fibres are all created to represent the micro-structure of the composite using Gmsh. Figure 1.5 show the steps undertaken for this study.

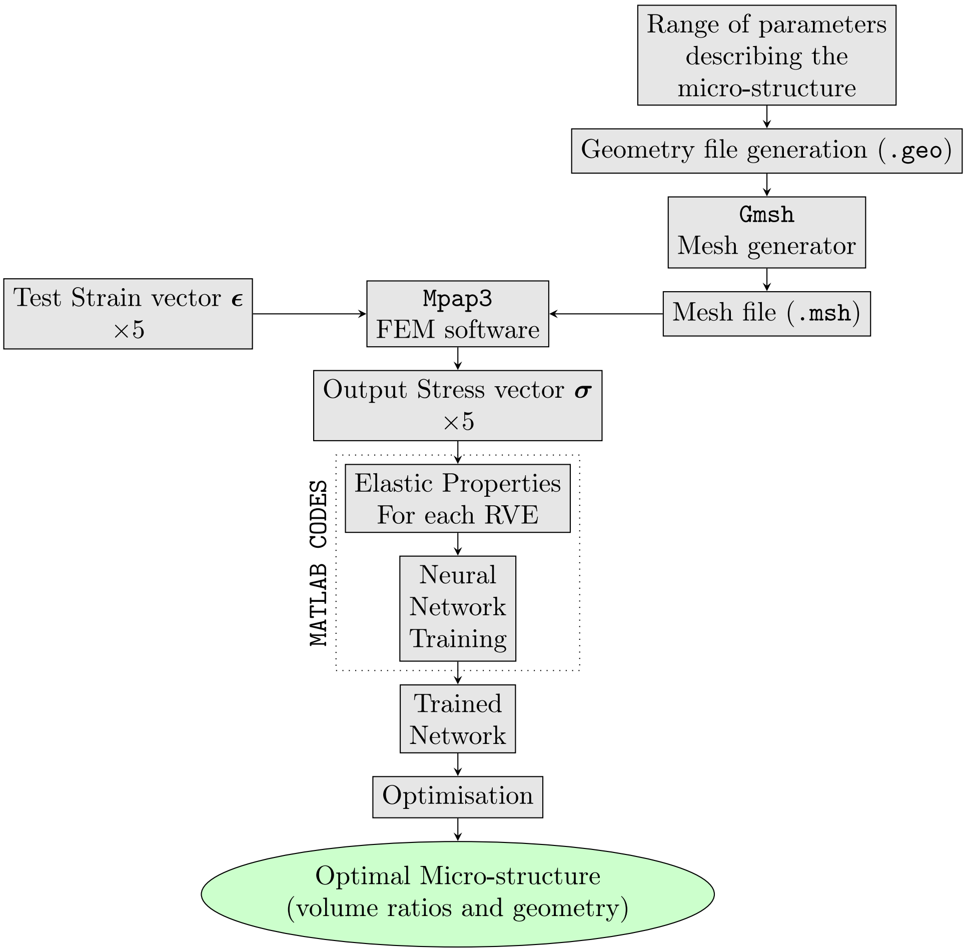

Range of parameters describing the micro-structure
Geometry file generation (.geo)
Gmsh Mesh generator
Mesh file (.msh)
Test Strain vector ϵ ×5
Mpap3 FEM software
Output Stress vector σ ×5
MATLAB CODES
Elastic Properties For each RVE
Neural Network Training
Trained Network
Optimisation
Optimal Micro-structure (volume ratios and geometry)

Figure 1.5: Procedural Methodology Utilized

Dettmer's Mpap3 (Multi Physics Analysis Program) is utilised to handle an arbitrary number of 3D RVEs which uses the mesh generated by `Gmsh`. Each RVE is subjected to five different test strain vectors as a means of improving the accuracy of the results. Mpap3 uses the finite element method to generates the output stresses which can be used to predict the effective composite material's elastic properties.

Using a built-in Mathworks function `lsqr`, a least square method is employed to generate the homogenised elastic properties of the composite. As part of this study, a MATLAB code was specifically developed to carry out the homogenisation of both isotropic and transversely isotropic materials.. This is an iterative method which works by reducing the normal residuals. By comparing this method to Halpin Tsai and Hashin Rosen, this approach ensures a more detailed and accurate analysis, contributing to the development of superior composite materials.

The results of the least square homogenisation are used to train a neural network, which in turn, can predict the elastic properties of the composites without the need to re-model its RVEs. This trained network produces accurate results, proving to be a reliable tool. The trained network, together with MATLAB built-in function `gamult`, which is an optimisation tool, are used to optimise the composite's micro-structure based on the desired objectives. Combination of this tools enhances the efficiency and precision of developing advanced composite materials.

1.4 Layout of Thesis

- **Chapter 1**
  Covers mechanics of solid materials: The stress state of a point in a continuum is defined by 81 elastic parameters. Using material symmetry and geometry these parameters are reduced. Orthotropic material is then defined with 21 constants, transverse isotropy with 9 constants and isotropic material with 2 constants. Hooke's law equations are then utilised for the continuum. The mathematical constants are then related to engineering constants which translates to Young's modulus, shear modulus, bulk modulus and Poisson's ratio which are used to define a material and consequently they can also be used to compute other elastic constants. This chapter also covers a brief review of the finite element method.

- **Chapter 2**
  Multi-scale Modelling: This chapter focuses on creating Representative Volume Element of a macro-continuum. The RVE is modelled in micro scale and different boundary conditions are discussed here. The effective use of this RVEs is looked into as well to get better results from tthem. The chapter also shows how the meshes were generated.

- **Chapter 3**
  Homogenisation based on the least square method is developed here. The transversely isotropic and isotropic equations developed in Chapter 1 are solved using the least square method for the effective composites' elastic constants using a MATLAB function that is already a built-in algorithm by MathWorks. Results from this method are compared to Halpin Tsai and Hashin Rosen approaches referred to as analytical methods in this chapter.

- **Chapter 4**
  Machine Learning Enhanced Homogenisation, this chapter focuses on training a neural network using the elastic properties obtained from Chapter 3 as training data. Since experiments carried out in chapter 3 are finite, this chapter trains a network such that elastic properties of the macro-structure can be estimated without the need to re-modelled the RVE.

- **Chapter 5**
  Optimisation: A trained network from Chapter 4 is subjected to objectives functions and the optimisation tool is used to optimise the micro-structure such that a balance between performance and resource utilisation is achieved. Objectives are developed as functions of material property (shear modulus and bulk modulus) and resource utilisation such as cost or weight. This chapter converts natural problems that is resource utilisation into mathematical problems. A multi-objective algorithm is used here to solve for multiple objectives. The purpose of this chapter is to identify this trade-offs and provide optimised values of the parameters which describe the micro-structure namely aspect ratio, void volume ratio and fibre volume ratio.

Chapter 2

Mechanics of Solid Materials

Mechanics of solid material is the study of the response of solid bodies under loading. The stresses and strains within the body are studied in order to enable the design of elements that will not fail within their service life. This chapter uses equations developed by Daniel and Ishai [7] to describe mathematical constants which are in turn related to Engineering constants/physical quantities. Engineering constants include the Young's modulus, Shear modulus, Poisson's ratio etc. The solids considered here are assumed to have continuous properties over a region of interest. This chapter will cover elasticity of an isotropic material and a transversely isotropic material. Mechanics of solids is a very important part of design in various fields such as civil, mechanical and aerospace hence understanding solid mechanics is crucial for safety and performance.

2.1 Linear Elasticity

In general, the stress state of a point in a continuum can be represented by Figure 2.1.

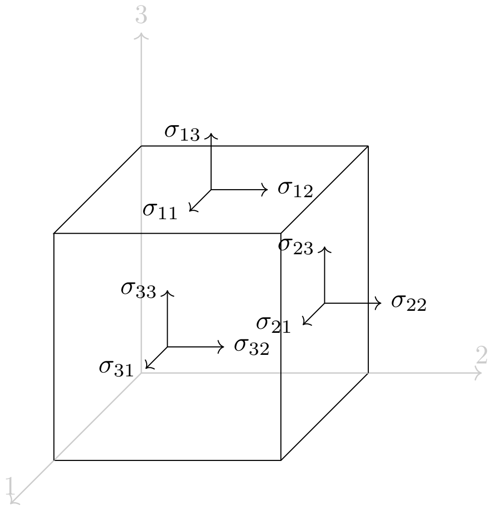


Figure 2.1: State of stress at a point in a continuum

Using a small cube of material, each of the three planes has a stress σ_{ij}, where $(j, i = 1, 2, 3)$, applied in the 1, 2, and 3 directions as shown above. The strain state at this point can also be obtained in a similar manner. Using the generalised Hooke's law, stresses and strains can be related using Equation 2.1 as Daniel and Ishai [7] showed.

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{31}\\ \sigma_{12}\\ \sigma_{32}\\ \sigma_{13}\\ \sigma_{21}\end{bmatrix} = \begin{bmatrix} c_{1111} & c_{1122} & c_{1133} & c_{1123} & c_{1131} & c_{1112} & c_{1132} & c_{1113} & c_{1121}\\ c_{2211} & c_{2222} & c_{2233} & c_{2223} & c_{2231} & c_{2212} & c_{2232} & c_{2213} & c_{2221}\\ c_{3311} & c_{3322} & c_{3333} & c_{3323} & c_{3331} & c_{3312} & c_{3332} & c_{3313} & c_{3321}\\ c_{2311} & c_{2322} & c_{2333} & c_{2323} & c_{2331} & c_{2312} & c_{2332} & c_{2313} & c_{2321}\\ c_{3111} & c_{3122} & c_{3133} & c_{3123} & c_{3131} & c_{3112} & c_{3132} & c_{3113} & c_{3121}\\ c_{1211} & c_{1222} & c_{1233} & c_{1223} & c_{1231} & c_{1212} & c_{1232} & c_{1213} & c_{1221}\\ c_{3211} & c_{3222} & c_{3233} & c_{3223} & c_{3231} & c_{3212} & c_{3232} & c_{3213} & c_{3221}\\ c_{1311} & c_{1322} & c_{1333} & c_{1323} & c_{1331} & c_{1312} & c_{1332} & c_{1313} & c_{1321}\\ c_{2111} & c_{2122} & c_{2133} & c_{2123} & c_{2131} & c_{2112} & c_{2132} & c_{2113} & c_{2121}\end{bmatrix} \times \begin{bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \varepsilon_{23}\\ \varepsilon_{31}\\ \varepsilon_{12}\\ \varepsilon_{32}\\ \varepsilon_{13}\\ \varepsilon_{21}\end{bmatrix} \tag{2.1}$$

In index notation the above equations can be represented as

$$\boldsymbol{\sigma_{ij} = C_{ijkl}\varepsilon_{kl}} \tag{2.2}$$

with $i, j, k, l = 1, 2, 3$ and $\boldsymbol{C_{ijkl}}$ as the stiffness matrix. It is therefore deduced that 81 elastic constants are required to fully describe the material. From the symmetry of the stress and strain tensor this 81 constants required can be reduced to 36 constants Daniel and Ishai [7]. That is

$$\boldsymbol{\sigma_{ij} = \sigma_{ji}} \tag{2.3}$$
$$\boldsymbol{\varepsilon_{ij} = \varepsilon_{ji}} \tag{2.4}$$

therefore from Equation (2.3) and (2.4) it follows that

$$\begin{gathered}\sigma_{11} = \sigma_1\\ \sigma_{22} = \sigma_2\\ \sigma_{33} = \sigma_3\\ \sigma_{23} = \sigma_4 = \tau_{23} = \tau_4\\ \sigma_{31} = \sigma_5 = \tau_{31} = \tau_5\\ \varepsilon_{11} = \varepsilon_1\\ \varepsilon_{22} = \varepsilon_2\\ \varepsilon_{33} = \varepsilon_3\\ 2\varepsilon_{23} = \varepsilon_4 = \gamma_{23} = \gamma_4\\ 2\varepsilon_{31} = \varepsilon_5 = \gamma_{31} = \gamma_5\\ 2\varepsilon_{12} = \varepsilon_6 = \gamma_{12} = \gamma_6.\end{gathered} \tag{2.5}$$

For the stiffness matrix repeated subscipts can be put together and the C-matrix/Stiffness matrix is reduced as follows

$$\boldsymbol{C_{iijj} = C_{ij}}. \tag{2.6}$$

Therefore substituting Equations (2.3), (2.4) and (2.6) into Equation (2.1) the following reduced equation which represents the stress-strain relationship for anisotropic material is obtained.

$$\begin{bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\\ \tau_4\\ \tau_5\\ \tau_6\end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16}\\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26}\\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36}\\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46}\\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56}\\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66}\end{bmatrix} \times \begin{bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_3\\ \gamma_4\\ \gamma_5\\ \gamma_6\end{bmatrix} \tag{2.7}$$

In index notation

$$\boldsymbol{\sigma_i} = \boldsymbol{C_{ij}\varepsilon_j} \tag{2.8}$$

where $i, j = 1, 2, ...6$. Anisotropy of a material refers to its property to change or assume different properties in different directions as opposed to isotropy. These properties involve Youngs modulus, shear modulus etc. Examples of anisotropic materials include wood, carbon fibres, graphite etc. Using the energy considerations, strain energy density (per unit volume) can be represented by the area under the stress-strain curve as shown in Figure 2.2.

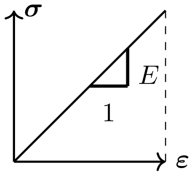


Figure 2.2: Stress-strain curve for a linear elastic material subjected to uniaxial loading.

Therefore the work done per unit volume can be presented as

$$W = \frac{1}{2}\boldsymbol{E}\varepsilon^2. \tag{2.9}$$

Since the Young's modulus, $E = \frac{\sigma}{\varepsilon}$ and expressing $\boldsymbol{\sigma}$ in a similar manner as in Equation (2.8). The work done can be expressed as

$$W = \frac{1}{2}\boldsymbol{C_{ij}\varepsilon_i\varepsilon_j}. \tag{2.10}$$

With the property

$$\boldsymbol{\sigma_i} = \frac{\partial W}{\partial \boldsymbol{\epsilon_i}} \tag{2.11}$$

It can be deduced that the strain energy density W is a quadratic function of strain $\boldsymbol{\varepsilon}$. Materials with the above properties are called elastic materials. Similarly W can also be expressed as

$$W = \frac{1}{2}\boldsymbol{C_{ji}\varepsilon_j\varepsilon_i} \tag{2.12}$$

therefore Equation (2.10) minus Equation (2.12) should be zero,

$$0 = (\boldsymbol{C_{ij}} - \boldsymbol{C_{ji}})\boldsymbol{\varepsilon_j\varepsilon_i}. \tag{2.13}$$

From Equation (2.13) there is a property $\boldsymbol{C_{ij} = C_{ji}}$ which implies that this two matrices should be symmetric. That is the stiffness matrix can be expressed as

$$\boldsymbol{C_{ij}} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & & & & C_{55} & C_{56} \\ & & & & & C_{66} \end{bmatrix} \tag{2.14}$$

so that the number of constants required to fully describe the stress state of an anisotropic material at a point in continuum is further reduced to 21 constants.

2.2 Orthotropy

An orthotropic material with three planes of symmetry has a stiffness matrix fully described by a 6x6 matrix in Equation (2.14). However, the elastic constants are reduced to 9 constants due to the orthotropic nature of the material and the similarities in the stiffness matrix as explained Daniel and Ishai [7]. Therefore, Equation (2.7) is reduced to

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} \tag{2.15}$$

and

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} & 0 & 0 & 0 \\ S_{21} & S_{22} & S_{23} & 0 & 0 & 0 \\ S_{31} & S_{32} & S_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{66} \end{bmatrix} \times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} \tag{2.16}$$

where $\boldsymbol{S}$ is the inverse matrix of the stiffness matrix. Orthotropic material under plane stresses, laminae are used with a condition that stress components in direction 3 are all zero. Daniel and Ishai [7] reduced Equation (2.15) reduced to

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ 0 \\ 0 \\ 0 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} \tag{2.17}$$

and expanding Equation (2.17) gives

$$\begin{gathered} \sigma_1 = (C_{11} - \frac{C_{13}C_{13}}{C_{33}})\varepsilon_1 + (C_{12} - \frac{C_{13}C_{23}}{C_{33}})\varepsilon_2 = Q_{11}\varepsilon_1 + Q_{12}\varepsilon_2 \\ \sigma_2 = (C_{12} - \frac{C_{23}C_{13}}{C_{33}})\varepsilon_1 + (C_{22} - \frac{C_{23}C_{23}}{C_{33}})\varepsilon_2 = Q_{12}\varepsilon_1 + Q_{22}\varepsilon_2 \\ \tau_6 = C_{66}\gamma_6 = Q_{66}\gamma_6 \end{gathered} \tag{2.18}$$

which can also be expressed as

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{12} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \gamma_6 \end{bmatrix} \tag{2.19}$$

where

$$Q_{ij} = C_{ij} - \frac{C_{i3}C_{j3}}{C_{33}} \tag{2.20}$$

and $i, j = 1, 2..6$. For in-plane stress, the constants are further reduced to 4. However, the out-of-plane strain ϵ_3 is related to σ_1 and σ_2 through C_{13} and C_{23}. Therefore, two more constants are required to describe the in-plane stress-strain relation.

2.3 Transverse Isotropy

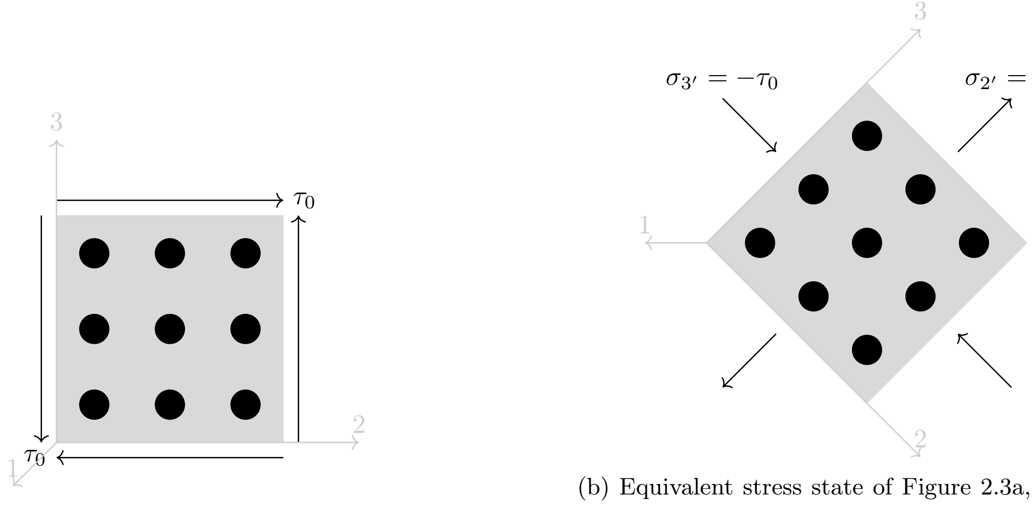
This is a material property characterised by anisotropy, in which elastic properties exhibit symmetry about an axis perpendicular to the primary axes. For transversely isotropic materials, properties may vary with direction but remain unchanged along the plane perpendicular to the primary axes. These planes are called planes of isotropy. Therefore it can be deduced that

$$\begin{aligned} C_{12} &= C_{13} \\ C_{22} &= C_{33} \end{aligned} \tag{2.21}$$

while also subscripts 5 and 6 are interchangeable so that

$$\begin{aligned} C_{55} &= C_{66} \\ S_{55} &= S_{66}. \end{aligned} \tag{2.22}$$

Using the stress transformation on the plane of isotropy of a transversely isotropic material as illustrated by Figure 2.3, it can be shown that C_{44} is not independent.



(a) Transversely isotropic material under pure shear.

(b) Equivalent stress state of Figure 2.3a, rotated 45°, subjected to axial tensile and compressive loading.

Figure 2.3: Stress transformation of a transversely isotropic material.

Therefore under pure shear $\tau_0 = \tau_{23}$ and $\varepsilon_0 = \varepsilon_{23}$ and from Equation (2.15)

$$\begin{aligned} \tau_4 = \tau_{23} = C_{44}\gamma_{23} = C_{44}\gamma_4 &= \tau_0 \\ \sigma_{2'} &= \tau_0 \\ \sigma_{3'} &= -\tau_0 \end{aligned} \tag{2.23}$$

resulting in

$$\begin{aligned} \varepsilon_{2'} = -\varepsilon_{3'} &= \frac{\gamma_{23}}{2} \\ \varepsilon_1 &= 0 \end{aligned} \tag{2.24}$$

hence from Equation (2.15)

$$\begin{aligned} \sigma_{2'} &= C_{2'2'}\varepsilon_{2'} + C_{2'3'}\varepsilon_{3'} = C_{22}\varepsilon_{2'} - C_{23}\varepsilon_{2'} \\ \sigma_{2'} &= \varepsilon_{2'}(C_{22} - C_{23}) = \frac{\gamma_{23}}{2}(C_{22} - C_{23}) \end{aligned} \tag{2.25}$$

Since $C_{2'2'} = C_{22}$ and $C_{2'3,} = C_{23}$ by combining Equations (2.23) and (2.25), we obtain $C_{44} = \frac{(C_{22}-C_{23})}{2}$. Therefore, the stress-strain relations of a transversely isotropic material can be described as

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{12} & C_{23} & C_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(C_{22}-C_{23})}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{55} \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} \tag{2.26}$$

hence 5 elastic constants are required to to fully describe stress state of a transversely isotropic material.

2.4 Isotropy

An isotropic material is fully described by 2 elastic constants which are independent. All directions are equivalent therefore

$$\begin{aligned} C_{11} = C_{22}, C_{22} = C_{33}, C_{33} = C_{11} \\ C_{13} = C_{23}, C_{13} = C_{21}, C_{23} = C_{31} \\ C_{44} = C_{55}, C_{55} = C_{66}, C_{66} = C_{44} \end{aligned} \tag{2.27}$$

which means

$$\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{C_{11}-C_{12}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{C_{11}-C_{12}}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{C_{11}-C_{12}}{2} \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} \tag{2.28}$$

2.5 The relationship between Engineering constants and mathematical constants

For a material subjected to uniaxial loading see Figure 2.4 , where σ_1 is nonzero, and σ_2 and σ_3 are zero , from constitutive relations in Equation (2.16), we obtain

$$\begin{aligned} \varepsilon_1 &= S_{11}\sigma_1 \\ \varepsilon_2 &= S_{12}\sigma_1 \\ \varepsilon_3 &= S_{13}\sigma_1 \\ \gamma_4 = \gamma_5 &= \gamma_6 = 0. \end{aligned} \tag{2.29}$$

With relation to Engineering constants the Youngs modulus in direction 1 is defined by

$$\varepsilon_1 = \frac{\sigma_1}{E_1}. \tag{2.30}$$

The Poisson's ratio ν_{12} and ν_{13} are given by

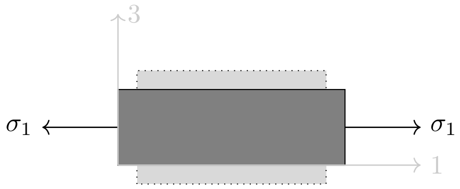


Figure 2.4: Material under simple tension in direction 1

$$\begin{aligned} \varepsilon_2 &= -\frac{\nu_{12}}{E_1}\sigma_1 \\ \varepsilon_3 &= -\frac{\nu_{13}}{E_1}\sigma_1 \\ \gamma_4 &= \gamma_5 = \gamma_6 = 0. \end{aligned} \tag{2.31}$$

From the constitutive relations by Daniel and Ishai [7], we can establish a relationship that

$$\begin{aligned} S_{11} &= \frac{1}{E_1} \\ S_{12} &= -\frac{\nu_{12}}{E_1} \\ S_{13} &= -\frac{\nu_{13}}{E_1} \end{aligned} \tag{2.32}$$

The same procedure can be applied to a material subjected to uniaxial loading in the other two directions and the following relations are obtained

$$\begin{aligned} \varepsilon_1 &= S_{12}\sigma_2 = -\frac{\nu_{21}}{E_2}\sigma_2, & \varepsilon_1 &= S_{13}\sigma_3 = -\frac{\nu_{23}}{E_3}\sigma_3 \\ \varepsilon_2 &= S_{22}\sigma_2 = \frac{\sigma_2}{E_2}, & \varepsilon_2 &= S_{23}\sigma_3 = -\frac{\nu_{32}}{E_3}\sigma_3 \\ \varepsilon_3 &= S_{23}\sigma_2 = -\frac{\nu_{23}}{E_2}\sigma_2, & \varepsilon_3 &= S_{33}\sigma_3 = \frac{\sigma_3}{E_3} \\ \gamma_4 &= \gamma_5 = \gamma_6 = 0 \end{aligned} \tag{2.33}$$

from which

$$\begin{aligned} S_{12} &= -\frac{\nu_{21}}{E_2}, & S_{13} &= -\frac{\nu_{31}}{E_3} \\ S_{22} &= \frac{1}{E_2}, & S_{23} &= -\frac{\nu_{32}}{E_3} \\ S_{23} &= -\frac{\nu_{21}}{E_2}, & S_{33} &= \frac{1}{E_3} \end{aligned} \tag{2.34}$$

For a material subjected to pure shearing as shown in Figure 2.5, when γ_4, γ_5, σ_1, σ_2 and σ_3 are all zero so that

$$\begin{aligned} \varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \gamma_4 = \gamma_5 = 0 \\ \gamma_6 = S_{66}\tau_6 = \frac{\tau_6}{G_{12}} \end{aligned} \tag{2.35}$$

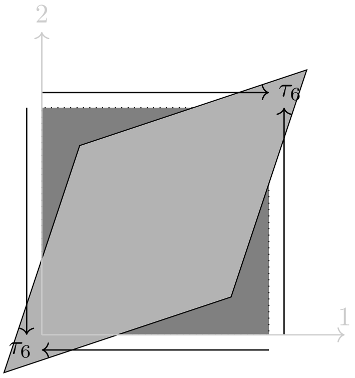


Figure 2.5: Material under simple pure shear

S_{66} is given by

$$S_{66} = \frac{1}{G_{12}} \tag{2.36}$$

Applying similar procedure to the out of plane shear τ_4 and τ_5 on 2-3 and 1-3 plane respectively S_{55} and S_{44} can be give as

$$S_{44} = \frac{1}{G_{23}}, \quad S_{55} = \frac{1}{G_{13}} \tag{2.37}$$

Equation (2.16) in Engineering constants can be expressed as

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{v_{21}}{E_2} & -\frac{v_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{v_{12}}{E_1} & \frac{1}{E_2} & -\frac{v_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{v_{13}}{E_1} & -\frac{v_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{13}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} \tag{2.38}$$

As a results of the symmetry of the $\boldsymbol{S_{ij}}$ matrix, we obtain the following relationship between Poisson's ratios and Young's modulus

$$\frac{\nu_{ij}}{E_i} = \frac{\nu_{ji}}{E_j}. \tag{2.39}$$

For transversely isotropic material $E_2 = E_3, G_{12} = G_{13}$ and $v_{12} = v_{13}$. In case of an isotropic material the stiffness matrix in Equation 2.28 where $C_{44} = \mu, C_{12} = \lambda$ and $C_{11} = \lambda + 2\mu$ can be expressed as

$$\begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{2.40}$$

and by taking the inverse of the above matrix to get the constants of a compliance matrix we get

$$\begin{aligned}
\det &= (\lambda+2\mu)^3 + 2\lambda^3 - 3\lambda^2(\lambda+2\mu) = 4\mu^2(3\lambda+2\mu) \\
S_{11} &= \frac{(\lambda+2\mu)^2+\lambda^2}{4\mu^2(3\lambda+2\mu)} = \frac{\lambda+\mu}{\mu(3\lambda+2\mu)} = \frac{1}{E} \\
S_{12} &= \frac{\lambda(\lambda+2\mu)^2+\lambda^2}{4\mu^2(3\lambda+2\mu)} = -\frac{\lambda}{2\mu(3\lambda+2\mu)} = -\frac{\nu}{E} \\
S_{44} &= \frac{1}{\mu} = 2\frac{1+\nu}{E}
\end{aligned} \tag{2.41}$$

Where λ is termed elastic moduli, μ is lame coefficient, E is the Young's modulus and ν is the Poisson's ratio we have the relation

$$\begin{aligned}
\lambda &= \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = G = \frac{E}{2(1+\nu)} \\
E &= \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}, \quad \nu = \frac{\lambda}{\lambda+\mu}
\end{aligned} \tag{2.42}$$

therefore 2 elastic constants either λ and μ or E and ν are required to describe an isotropic material fully.

2.6 The Finite Element Method

3D Finite element commonly consist of 4-noded tetrahedron and 8-noded hexahedron however this elements can contain more nodes. For this study, a simple finite element with 4 or 10 nodes see Figure 2.6 and 2.7, often referred to as a tetrahedral element, is considered.

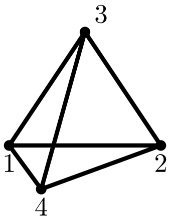


Figure 2.6: A 4-node tetrahedron element

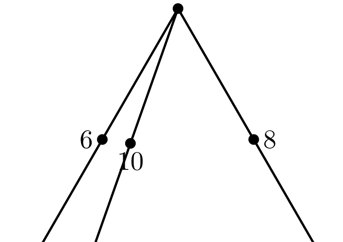


Figure 2.7: A 10-node tetrahedron element

For a 4-noded tetrahedron the approximatiom of linear nodal displacement takes a form

$$\begin{aligned} u(x,y,z) &= \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z \\ v(x,y,z) &= \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 z \\ w(x,y,z) &= \alpha_9 + \alpha_{10} x + \alpha_{11} y + \alpha_{12} z \end{aligned} \tag{2.43}$$

The notations and equations from Zienkiewicz and Taylor [53] were adopted, do let the displacement at any point within the element be a vector with

$$\boldsymbol{u} = \boldsymbol{\hat{u}} = \sum_{\boldsymbol{a}} \boldsymbol{N_a} \boldsymbol{\tilde{u}_a^e} = [\boldsymbol{N_1}, \boldsymbol{N_2}, ...] \begin{Bmatrix} \boldsymbol{\tilde{u}_1} \\ \boldsymbol{\tilde{u}_2} \\ . \\ . \\ . \end{Bmatrix}^{\boldsymbol{e}} = \boldsymbol{N\tilde{u}^e} \tag{2.44}$$

$\boldsymbol{u}$ from Equation (2.44) is the movements of a typical node a, within an element and $\boldsymbol{N_a}$ is the nodal shape function. For a 3D case elastic stress element with k nodes, with cartesian nodal displacement u_a, v_a, w_a associated with node a displacements are expressed as

$$\boldsymbol{u} = \begin{Bmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{Bmatrix} \tag{2.45}$$

$$\begin{aligned} u(x,y,z) &= \sum_{a=1}^{k} N_a(x,y,z) u_a \\ v(x,y,z) &= \sum_{a=1}^{k} N_a(x,y,z) v_a \\ w(x,y,z) &= \sum_{a=1}^{k} N_a(x,y,z) w_a. \end{aligned} \tag{2.46}$$

We now express the internal nodal displacements in terms of shape function. Consider a point p with coordinates (x, y, z) within an element which create 4 volumes (V_1, V_2, V_3, V_4), and V will represent the volume of the element made by node $1, 2, 3, 4$ then a new set of coordinates system L_k can be defined where

$$\begin{aligned} V &= V_1 + V_2 + V_3 + V_4 \\ L_k &= \frac{V_k}{V} \quad \text{for} \quad k = 1, 2, 3, 4. \end{aligned} \tag{2.47}$$

If point p correspond to node 1 we find that $V_1 = V$ and $V_2 = V_3 = V_4 = 0$ such that $L_1 = 1$ and $L_2 = L_3 = L_4 = 0$ at node 1. As p moves away from node 1 V_1 decreases linearly and L_1 is constant on any plane parallel to the base triangle of nodes $2, 3, 4$. L_k exhibits characteristics similar to those of a nodal function, it has a value of 1 at node k and zero on other nodes. Therefore nodal shape functions N at node k within an element can be written as

$$\begin{aligned} N_k &= L_k \\ N_k &= \frac{a_k + b_k x + c_k y + d_k z}{6V} \quad \text{for} \quad k = 1, 2, 3, 4 \end{aligned} \tag{2.48}$$

where

$$6V = \det \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \quad a_1 = \det \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{bmatrix} \tag{2.49}$$

$$b_1 = \det \begin{bmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{bmatrix} \quad c_1 = \det \begin{bmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{bmatrix} \quad d_1 = \det \begin{bmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1. \\ x_4 & y_4 & 1 \end{bmatrix}.$$

More detail on high order tetrahedron elements is explained in details by Zienkiewicz and Taylor [53] and Equation (2.50) can be to used to integrate any polynomial of these shape functions.

$$\iiint\limits_{vol} L_1^a L_2^b L_3^c L_4^d \, dx \, dy \, dz = \frac{a!b!c!d!}{(a+b+c+d+3)!} 6V \tag{2.50}$$

Where a, b, c represent degrees of basis functions in each coordinate direction and V is the total volume of the element. Hence the displacements are interpolated as

$$u = \sum_{k=1}^{4} N_k u_k \quad v = \sum_{k=1}^{4} N_k v_k \quad w = \sum_{k=1}^{4} N_k w_k \tag{2.51}$$

In matrix form this can be expressed as

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = [\boldsymbol{N}] \{\boldsymbol{u}\} \tag{2.52}$$

where $[\boldsymbol{N}]$ is diagonal matrix of snape function and u is the nodal displacement such that

$$\{\boldsymbol{u}\} = [u_1 \; v_1 \; w_1 \; u_2 \; v_2 \; w_2 \; \ldots \; \ldots \; \ldots \; u_4 \; v_4 \; w_4]^T$$

$$\boldsymbol{N} = \begin{bmatrix} [N] & 0 & 0 \\ 0 & [N] & 0 \\ 0 & 0 & [N] \end{bmatrix}. \tag{2.53}$$

It should be noted that each $[N]$ is a 1×4 row matrix of interpolation functions and lots of zero values. For an 8 noded prism the product of linear lagrange polynomials in x,y,z are used, a detailed derivation can be found in Hutton [21]. From the displacement in Equation (2.44) the strains can be obtained as

$$\boldsymbol{\epsilon} = \boldsymbol{S}\boldsymbol{u} \tag{2.54}$$

where $\boldsymbol{S}$ is the linear differential operator the Equation (2.54) can be approximated as

$$\begin{aligned} \boldsymbol{\epsilon} = \hat{\boldsymbol{\epsilon}} &= \boldsymbol{B}\tilde{\boldsymbol{u}}^e \\ \boldsymbol{B} &= \boldsymbol{S}\boldsymbol{N}. \end{aligned} \tag{2.55}$$

The strains can be related to displacements (v, u, w) in the x, y, z directions, which will also defines the operator $\boldsymbol{S}$ where

$$\{\epsilon\} = \left\{\begin{array}{c} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{array}\right\} = \left\{\begin{array}{c} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial w}{\partial z} \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \\ \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \\ \frac{\partial w}{\partial z} + \frac{\partial u}{\partial x} \end{array}\right\} = \left\{\begin{array}{ccc} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \end{array}\right\} \left\{\begin{array}{c} u \\ v \\ w \end{array}\right\}$$

With the shape function known from Neto et al. [33] and Hutton [21] for 4 noded tetrahedron element, internal strains can be deduced as

$$\{\boldsymbol{\epsilon}\} = \frac{1}{6V} \left\{\begin{array}{cccccccccc} \frac{\partial N_1}{\partial x} & 0 & 0 & \dots & \dots & \dots & \frac{\partial N_4}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \dots & \dots & \dots & 0 & \frac{\partial N_4}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_1}{\partial z} & \dots & \dots & \dots & 0 & 0 & \frac{\partial N_4}{\partial z} \\ \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} & 0 & \dots & \dots & \dots & \frac{\partial N_4}{\partial x} & \frac{\partial N_4}{\partial y} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial z} & \dots & \dots & \dots & 0 & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial z} \\ \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_1}{\partial z} & \dots & \dots & \dots & \frac{\partial N_4}{\partial x} & 0 & \frac{\partial N_4}{\partial z} \end{array}\right\} \left\{\begin{array}{c} u_1 \\ v_1 \\ w_1 \\ \dots \\ \dots \\ \dots \\ u_4 \\ v_4 \\ w_4 \end{array}\right\} \tag{2.56}$$

hence

$$\{\boldsymbol{\epsilon}\} = [\mathbf{B}]\{\mathbf{u}\} \tag{2.57}$$

If an element is subjected to initial strains, stresses will be induced within the element. Assuming linear elastic behaviour, these stresses can be added to the deformation, and the stress-strain relationship will take the form

$$\{\boldsymbol{\sigma}\} = \mathbf{D}(\{\boldsymbol{\epsilon}\} - \boldsymbol{\epsilon}_\mathbf{0}) + \boldsymbol{\sigma}_\mathbf{0} \tag{2.58}$$

$$\boldsymbol{\sigma} = \left\{\begin{array}{c} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{array}\right\}$$

where **D** is an elastic matrix of the material properties. **D** for an isotropic material is

$$\mathbf{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{2.59}$$

whereas elasticity matrix **D** for a transversely isotropic material is taken as

$$\mathbf{D} = \begin{bmatrix} \frac{E_1}{1-\nu_{12}^2} & \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{23}} & \frac{\nu_{13}E_2}{1-\nu_{12}\nu_{23}} & 0 & 0 & 0 \\ \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{23}} & \frac{E_1}{1-\nu_{12}^2} & \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{23}} & 0 & 0 & 0 \\ \frac{\nu_{13}E_2}{1-\nu_{12}\nu_{23}} & \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{23}} & \frac{E_2}{1-\nu_{12}^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & G_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & G_{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & G_{23} \end{bmatrix} \tag{2.60}$$

Consequently, the nodal forces equivalent to the boundary stresses and distributed body forces on an element can be defined as

$$\text{Nodal forces} \quad \mathbf{q_k^e} = \begin{Bmatrix} u_k^e \\ v_k^e \\ w_k^e \end{Bmatrix} \quad \text{body forces} \quad \mathbf{b} = \begin{Bmatrix} b_x \\ b_y \\ b_z \end{Bmatrix} \tag{2.61}$$

where u, v, w represents the directions x, y, z respectively and while b_x, b_y, b_z are the body forces per unit volume. Following this definition, the total potential energy of an element, as described by Hutton [21] subjected to direct nodal forces can be expressed as

$$\begin{aligned} \Pi = U_e - W &= \frac{1}{2}\iiint\limits_v \{\boldsymbol{\epsilon}\}^{\boldsymbol{T}}[\boldsymbol{D}]\{\boldsymbol{\epsilon}\}\text{dv} - \{\boldsymbol{u}\}^{\boldsymbol{T}}\{\mathbf{q}\} \\ &= \frac{1}{2}\iiint\limits_v [\mathbf{B}]^{\mathbf{T}}\{\mathbf{u}\}^{\mathbf{T}}[\mathbf{D}][\mathbf{B}]\{\mathbf{u}\}\text{dv} - \{\boldsymbol{u}\}^{\boldsymbol{T}}\{\mathbf{q}\} \end{aligned} \tag{2.62}$$

Where U_e is the total strain energy and W is the work done by applied nodal forces. As the element is a portion of a large structure total energy of an element need to be minimal. Hence with the relations that

$$\{\mathbf{f}\} = \{u\}^T\{q\} \tag{2.63}$$

Equation (2.63) takes the form

$$\{\mathbf{K}\}\{\mathbf{u}\} = \{\mathbf{f}\} \tag{2.64}$$

Where $\{\mathbf{K}\} = [\mathbf{B}]^{\mathbf{T}}[\mathbf{D}][\mathbf{B}]V$ and is the stiffness matrix. Equation (2.64) forms the foundation of the finite element method, and a large system of linear equations is solved to determine the nodal displacements. For an element with distributed edge loadings $\{\mathbf{f^{(d)}}\}$ and body forces$\{\mathbf{f^{(b)}}\}$ and direct nodal forces $\{\mathbf{f}\}$ a general Equation (2.65) below can be used

$$\{\mathbf{K}\}\{\mathbf{u}\} = \{\mathbf{f^{(d)}}\} + \{\mathbf{f^{(b)}}\} + \{\mathbf{f}\} \tag{2.65}$$

where

$$\{\mathbf{f^{(b)}}\} = \int_v [N]^T \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \text{dv} \quad \text{and} \quad \{\mathbf{f^{(d)}}\} = \int_A [N]^T \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \text{dA}. \tag{2.66}$$

For a general 3D continuum as explained by Zienkiewicz and Taylor [53], the equilibrium equations of an elementary volume in terms of cartesian stress tensor can be described as

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = 0 \tag{2.67}$$

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x & \sigma_x & \sigma_x & \tau_{xy} & \tau_{yz} & \tau_{zx} \end{bmatrix}^T.$$

Let

$$\{\mathbf{A}\} = -\begin{Bmatrix} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + b_x \\ \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + b_y \\ \\ \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + b_z \end{Bmatrix} = \mathbf{0} \tag{2.68}$$

therefore using virtual work, we look for a function $\boldsymbol{u}$ such that it satisfy a set of differential equation $A(u) = 0$ in a domain Ω with boundary conditions $B(u)$ on boundaries Γ. Since strain is a function of displacements

$$\boldsymbol{u} = \begin{bmatrix} u & v & w \end{bmatrix}^T, \quad \text{weight function} \quad \boldsymbol{v} = \delta\boldsymbol{u} = \begin{bmatrix} \delta u & \delta v & \delta w \end{bmatrix}^T, \quad \boldsymbol{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}^T \tag{2.69}$$

after introducing the weight function a weak form can be obtained and the integral over the domain can be written as

$$\int_\Omega \delta\boldsymbol{u^T}\boldsymbol{A(u)}\ \mathrm{d}\Omega = 0 \tag{2.70}$$

$$\int_\Omega \left[\tfrac{\partial\delta u}{\partial x}\sigma_x + (\tfrac{\partial\delta u}{\partial y} + \tfrac{\partial\delta v}{\partial x})\tau_{xy} + ... - \delta u b_x - \delta v b_y - \delta w b_z\right] \mathrm{d}\Omega - \int_\Gamma \left[\delta u t_x + \delta v t_y + \delta w t_z\right] \mathrm{d}\Gamma = 0$$

hence

$$\int_\Omega \delta\epsilon^T\boldsymbol{\sigma}\mathrm{d}\Omega - \int_\Omega \delta\boldsymbol{u^T}\boldsymbol{b}\mathrm{d}\Omega - \int_\Gamma \delta\boldsymbol{u^T}\boldsymbol{t}\mathrm{d}\Gamma = 0 \tag{2.71}$$

where $\delta\boldsymbol{\epsilon} = \boldsymbol{S}\delta\boldsymbol{u}$ and $\boldsymbol{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ is the tractions acting per unit area of external boundary surface. For linear elastic behaviour we substitute in Equation (2.6) into Equation (2.71) with condition that displacements are known at each point on the boundary as $u = \hat{u}$ on Γ according to Zienkiewicz and Taylor [53]. Equation (2.71) is written as

$$\sum_e \int_{\Omega^e} \delta\boldsymbol{\epsilon^T}\left[\mathbf{D}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon_0}) + \boldsymbol{\sigma_0}\right]\mathrm{d}\Omega - \sum_e \int_{\Omega^e} \delta\boldsymbol{u^T}\boldsymbol{b}\mathrm{d}\Omega - \sum_e \int_{\Gamma_t^e} \delta\boldsymbol{u^T}\hat{\boldsymbol{t}}\mathrm{d}\Gamma = 0 \tag{2.72}$$

$$\sum_e \delta\hat{\boldsymbol{u}}_a^T \left[\int_{\Omega^e} \boldsymbol{B}_a^T\left(\mathbf{D}(\boldsymbol{B}_b\hat{\boldsymbol{u}}_b - \boldsymbol{\epsilon_0}) + \boldsymbol{\sigma_0}\right)\mathrm{d}\Omega - \int_{\Omega^e} N_a\boldsymbol{b}\,\mathrm{d}\Omega - \int_{\Gamma_t^e} N_a\hat{\boldsymbol{t}}\,\mathrm{d}\Gamma\right] = 0$$

where Ω^e and Γ_e^e represent element domain and tractions respectively. The above equation can be written in the form $\mathbf{K}_{ab}\hat{\mathbf{u}}_b + \mathbf{f}_a = 0$ where the stiffness matrix and load matrix are

$$\mathbf{K}_{ab} = \sum_e \int_{\Omega^e} \mathbf{B}_a^T\mathbf{D}\mathbf{B}_b\mathrm{d}\Omega \tag{2.73}$$

$$\mathbf{f_a} = \sum_{\mathbf{e}} \int_{\boldsymbol{\Omega^e}} \left[\mathbf{B}_a^T(\boldsymbol{\sigma_0} - \mathbf{D}\boldsymbol{\epsilon_0}) - N_a\mathbf{b}\right]\mathrm{d}\Omega - \sum_e \int_{\Gamma_t^e} N_a\hat{\mathbf{t}}\mathrm{d}\Gamma$$

Chapter 3

Multi-Scale Modelling

Multi-scale modelling of composite materials entails the integration of models across different scales to predict the behaviour and properties of composites. This involves the micro-scale, where the composite's microstructure is analysed, such as matrix and fibre interactions. At this level, the mechanical and physical behaviour of individual components, such as the matrix and fibres, is analysed to determine the overall behaviour of the composite. Clusters of fibres and some parts of the composite may be studied as well. At the macro-scale, the micro-scale information is used to predict the overall performance of the composite under loading. This comprehensive approach allows for a more accurate understanding of composite behaviour. Ultimately, multi-scale modelling enhances the design and optimisation of composite materials for various applications.

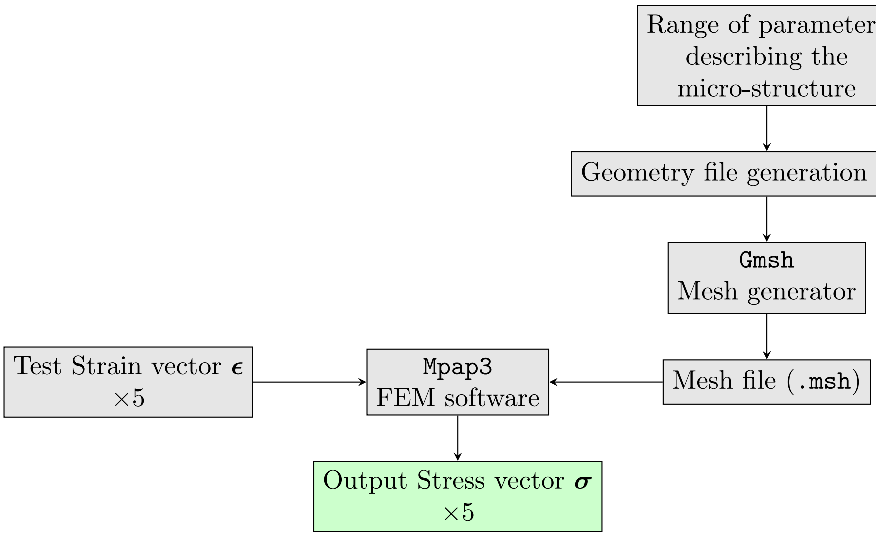

Range of parameters describing the micro-structure
Geometry file generation (.geo)
Gmsh
Mesh generator
Test Strain vector $\boldsymbol{\epsilon}$
×5
Mpap3
FEM software
Mesh file (.msh)
Output Stress vector $\boldsymbol{\sigma}$
×5

Figure 3.1: Procedural Methodology Utilized

3.1 Representative Volume Element

Multi-scale modelling process discussed in this chapter uses concepts and format borrowed from Peric et al. [38] and de Souza Neto and Feijóo [8] which uses the assumptions that material at point x of the macro-scopic continuum is associated to a local RVE with domain Ω_μ which is made of solid part Ω_μ^s and void part Ω_μ^v with boundary $\partial\Omega_\mu$. It is also assumed that the characteristic length of the RVE l_μ is much smaller than the length of the l of the macro-continuum. Therefore, a Representative Volume Element (RVE) see Figure 3.2 is a small local sample of a material. By analysing the RVE, we can understand the material's macroscopic behaviour based on its microscopic details.

$$\Omega_\mu = \Omega_\mu^s \cup \Omega_\mu^v \tag{3.1}$$

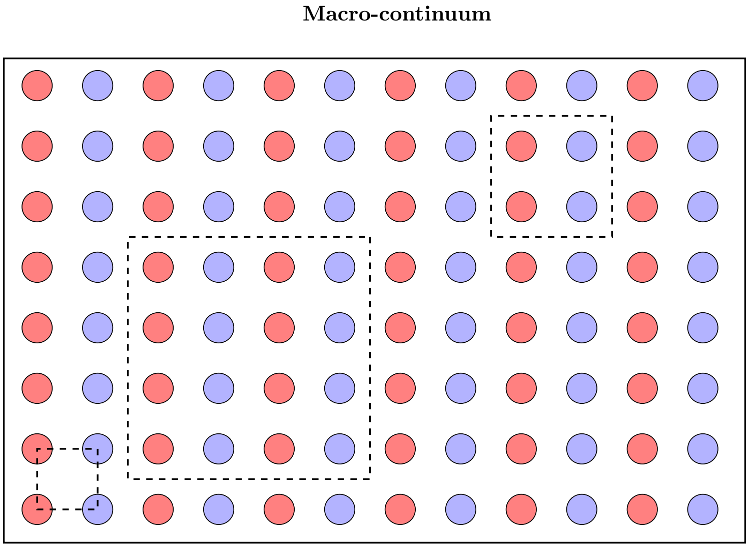


Figure 3.2: Macro-continuum containing micro continuum RVEs. Where red and blue represents different inclusion within an RVE

Assuming strain and stresses at arbitrary point x_{m} in the macro continuum is equivalent to the volume average of the microscopic strain and stress tensor field respectively over the local microscopic RVE domain associated with x_{m}. The strain tensor $\boldsymbol{\epsilon_{\mathrm{m}}}$ at point x_{m} of the macro continuum is expressed as

$$\boldsymbol{\epsilon_{\mathrm{m}}} = \frac{1}{V}\int_{\Omega_\mu} \boldsymbol{\epsilon_\mu}\mathrm{dV} \quad = \frac{1}{V}\int_{\Omega_\mu} \boldsymbol{\nabla^s u_\mu}\mathrm{dV} \tag{3.2}$$

where $\nabla^s u_\mu$ is the gradient of the microscopic displacement field u_μ of the RVE. The process in which a microscopic quantity distribution is translated into a macroscopic quantity over the Representative Volume Element (RVE) is referred to as homogenisation. Equation (3.2) places constraints on the possible displacement fields of the RVE and therefore only displacements fields that satisfy Equation (3.2) are kinematically admissible. This constraints can be expressed as $\boldsymbol{u_\mu} \in \mathscr{K}_\mu^*$, *minimally constrained set of kinematically admissible microscopic displacements* where

$$\mathscr{K}_\mu^* \equiv \left\{ \boldsymbol{v}, \quad \text{sufficiently regular} \mid \quad \int_{\Omega_\mu} \boldsymbol{\nabla^s v}\mathrm{dV} = V_\mu \boldsymbol{\epsilon_{\mathrm{m}}} \right\} \tag{3.3}$$

which can also be expressed in terms of unit normals $\mathbf{n}$ to the RVE boundary as

$$\mathscr{K}_\mu^* \equiv \left\{ \boldsymbol{v}, \quad \text{sufficiently regular} \mid \quad \int_{\partial\Omega_\mu} \boldsymbol{v} \otimes_{\boldsymbol{s}} \boldsymbol{n}\mathrm{dA} = V_\mu \boldsymbol{\epsilon_{\mathrm{m}}} \right\}. \tag{3.4}$$

The microscopic displacement field $\boldsymbol{u_\mu}$ can be divided into linear displacement in $\boldsymbol{y}$ as $\boldsymbol{\epsilon}_{\mathrm{m}}\boldsymbol{y}$ and microscopic displacement fluctuations $\boldsymbol{\tilde{u}_\mu}$ as

$$\boldsymbol{u_\mu}(\boldsymbol{y},t) = \boldsymbol{\epsilon_{\mathrm{m}}}(\boldsymbol{x},t)\,y + \boldsymbol{\tilde{u}}_\mu(\boldsymbol{y},t)\,. \tag{3.5}$$

Similarly the decomposed microscopic strain field constant in $\boldsymbol{y}$ concurrent with the macroscopic strain and a field representing strain fluctuations, can be expressed as

$$\boldsymbol{\epsilon_\mu}(\boldsymbol{y},t) = \boldsymbol{\epsilon_{\mathrm{m}}}(\boldsymbol{x},t) + \boldsymbol{\tilde{\epsilon}_{\mathrm{m}}}(\boldsymbol{y},t) \tag{3.6}$$

where

$$\boldsymbol{\tilde{\epsilon}_{\mathrm{m}}} = \boldsymbol{\nabla^s \tilde{u}^*_\mu} \tag{3.7}$$

that typically changes as a function of $\boldsymbol{y}$. Taking into account the condition that $\boldsymbol{u}_\mu \in \mathscr{K}_\mu^*$ and constraints in Equation (3.4), the final constraint set can be given as $\boldsymbol{\tilde{u}_\mu} \in \tilde{\mathscr{K}}_\mu^*$ such that

$$\tilde{\mathscr{K}}_\mu^* \equiv \left\{ \boldsymbol{v}, \quad \text{sufficiently regular} \mid \quad \int_{\partial\Omega_\mu} \boldsymbol{v} \otimes_{\boldsymbol{s}} \boldsymbol{n}\mathrm{dA} = 0 \right\}. \tag{3.8}$$

The virtual work variational framework imposed here requires a set of kinematically admissible RVE displacement $\tilde{\mathscr{K}}_\mu \subset \tilde{\mathscr{K}}_\mu^*$ with the virtual space as $\mathscr{V}_\mu = \tilde{\mathscr{K}}_\mu \subset \tilde{\mathscr{K}}_\mu^*$. Some examples of such constraints which satisfy Equation (3.8) include periodic boundary condition where the macrostructure should be replicated by stacking together replicas of the RVE. For this case the boundary consists of the sum of mirror image sets subsets $\partial\Omega^+$ and $\partial\Omega^-$ and unit normals n^+ and n^- such $\{n^+ = -n^- | \forall\, y \in \partial\Omega\}$ as illustrated by Figure 3.3 and Figure 3.4. However It should be emphasised, as noted by Guedes and Kikuchi [16], the entirety of the macroscopic continuum does not necessarily entail the repetition of a single Representative Volume Element (RVE). The constraint periodic space is

$$\tilde{\mathscr{V}}_\mu^{pr} \equiv \{\tilde{\boldsymbol{u}}_{\boldsymbol{\mu}} \in \tilde{\mathscr{K}}_\mu^* \quad | \quad \tilde{\boldsymbol{u}}_{\boldsymbol{\mu}}(\boldsymbol{y}^+) = \tilde{\boldsymbol{u}}_{\boldsymbol{\mu}}(\boldsymbol{y}^-) \quad \forall \{\boldsymbol{y}^+, \boldsymbol{y}^-\} \in \partial\Omega_\mu\} \tag{3.9}$$

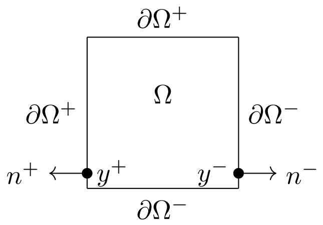
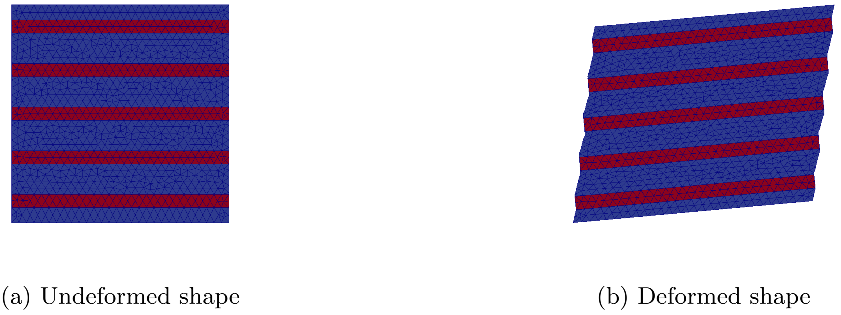


Figure 3.3: RVE for periodic boundary condition



(a) Undeformed shape

(b) Deformed shape

Figure 3.4: Periodic boundary condition

The linear boundary deformation model assumes displacement fluctuations are negligible over the boundary see Figure 3.5 so that the linear virtual space is defined as

$$\mathscr{V}_\mu^{lin} \equiv \{\tilde{\boldsymbol{u}}_{\boldsymbol{\mu}} \in \tilde{\mathscr{K}}_\mu^* \quad | \quad \tilde{\boldsymbol{u}}_{\boldsymbol{\mu}}(\boldsymbol{y}) = 0 \quad \forall\, \boldsymbol{y} \in \partial\Omega\} \tag{3.10}$$

This means the displacements in in y direction are linear meaning $u(y) = \epsilon(y)y$ on $\partial\Omega_\mu$. Figure 3.5 shows 2D linear deformation of an RVE with long continuous inclusions. In the case of uniform

(a) Undeformed shape

(b) Deformed shape

Figure 3.5: Linear boundary condition

boundary traction the model is defined with the assumption of minimum kinematical constrain space

$$\mathscr{V}_\mu^{uni} \equiv \tilde{\mathscr{K}}_\mu^*. \tag{3.11}$$

The state of equilibrium within the Representative Volume Element (RVE) subjected to body forces $\mathbf{b}$ per unit volume and external traction field $\mathbf{t}$ per unit area on its external boundary $\partial\Omega_\mu$ at each instant at point x_μ of the RVE is said to be in equilibrium if

$$\int_{\Omega_\mu^s} \boldsymbol{\sigma_\mu} : \boldsymbol{\nabla^s \eta}\mathrm{dV} - \int_{\Omega_\mu^s} \mathbf{b}.\boldsymbol{\eta}\mathrm{dV} - \int_{\partial\Omega_\mu} \mathbf{t}.\boldsymbol{\eta}\mathrm{dA} = 0 \qquad (3.12)$$
$$\forall\, \eta \in \mathscr{V}_\mu$$

where the (μ) donates the microscopic continuum, the variational equations are valid at time t, where $\mathscr{V}_\mu$ represents the space of virtual displacements of the Representative Volume Element (RVE) with constraints above. In a manner similar to macroscopic strains, the macroscopic stress tensor $\boldsymbol{\sigma_m}$ at point x_m of the macro continuum is described as the homogenisation of the microscopic tensor field $\boldsymbol{\sigma_\mu}$ of the RVE associated with that point as

$$\boldsymbol{\sigma}_{\mathbf{m}} = \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu}\mathrm{dV}. \qquad (3.13)$$

Equation 3.13 can be expressed in terms of of body forces $\mathbf{b}$ and boundary traction $\mathbf{t}$ using a tensor relations as

$$\begin{aligned}\frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu}\mathrm{dV} &= \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu}\mathbf{I}\mathrm{dV} \\ &= \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu}(\boldsymbol{\nabla y})^{\boldsymbol{T}}\mathrm{dV} \\ &= \frac{1}{V_\mu}\int_{\partial\Omega_\mu} (\boldsymbol{\sigma_\mu}\mathbf{n})\otimes\mathbf{y}\mathrm{dA} - \frac{1}{V}\int_{\Omega_\mu^s} (\mathrm{div}\boldsymbol{\sigma_\mu})\otimes\mathbf{y}\mathrm{dV} \\ &+ \frac{1}{V_\mu}\int_{\partial\Omega_\mu^v} (\boldsymbol{\sigma_\mu}\mathbf{n})\otimes\mathbf{y}\mathrm{dA} - \frac{1}{V}\int_{\Omega_\mu^v} (\mathrm{div}\boldsymbol{\sigma_\mu})\otimes\mathbf{y}\mathrm{dV}.\end{aligned} \qquad (3.14)$$

Combining Equation (3.14) and the equilibrium equation with the consideration that the body force acting on Ω^v for all values of $\boldsymbol{y}$ on this void domain is zero the stress tensor can be expressed as

$$\boldsymbol{\sigma}_{\mathbf{m}} = \frac{1}{V_\mu}\left[\int_{\partial\Omega_\mu} \mathbf{t}\otimes_s \mathbf{y}\mathrm{dA} - \int_{\Omega_\mu^s} \mathbf{b}\otimes_s \mathbf{y}dV\right]. \qquad (3.15)$$

The Hill Mandel Principle principle states that the macroscopic stress power is equal to the volume average of the microscopic stress power for any kinematically admissible motion of the RVE. This can be given as

$$\boldsymbol{\sigma}_{\mathbf{m}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{m}} = \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu} : \dot{\boldsymbol{\epsilon}}_{\boldsymbol{\mu}}dV \qquad (3.16)$$

for kinematically admissible microscopic strain rate $\dot{\epsilon}_\mu$, where $\mathscr{V}_\mu$ is the kinematically admissible fluctuation velocity space such that

$$\dot{\boldsymbol{\epsilon}}_{\boldsymbol{\mu}} = \boldsymbol{\nabla^s}\dot{\boldsymbol{u}}_{\boldsymbol{\mu}} = \dot{\boldsymbol{\epsilon}}_{\mathbf{m}} + \boldsymbol{\nabla^s}\dot{\tilde{\boldsymbol{u}}}_{\boldsymbol{\mu}} \qquad \dot{\tilde{\boldsymbol{u}}}_{\boldsymbol{\mu}} \in \mathscr{V}_\mu. \qquad (3.17)$$

Therefore Equation (3.16) can be reduced to

$$\begin{aligned}\frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu} : \dot{\boldsymbol{\epsilon}}_{\boldsymbol{\mu}}\mathrm{dV} &= \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu} : (\dot{\boldsymbol{\epsilon}}_{\mathbf{m}} + \boldsymbol{\nabla^s}\dot{\tilde{\boldsymbol{u}}})\mathrm{dV} \\ &= \boldsymbol{\sigma}_{\mathbf{m}} : \dot{\boldsymbol{\epsilon}}_{\mathbf{m}} + \frac{1}{V_\mu}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu} : \boldsymbol{\nabla^s}\dot{\tilde{\boldsymbol{u}}}_{\boldsymbol{\mu}}\mathrm{dV}\end{aligned} \qquad (3.18)$$

hence

$$\frac{1}{V_u}\int_{\Omega_\mu} \boldsymbol{\sigma_\mu} : \boldsymbol{\nabla^s}\dot{\tilde{\boldsymbol{u}}}_{\boldsymbol{\mu}} \mathrm{dV} = 0 \tag{3.19}$$

for Equation (3.16) to stand. When considering strong form of equilibrium equation and assuming that body forces acting on the void domain are neglected, Equation 3.19 can also be expressed as

$$\int_{\partial\Omega_\mu} \mathbf{t}\cdot\dot{\tilde{\mathbf{u}}}_\mu \mathrm{dA} - \int_{\Omega^s_\mu} \mathbf{b}\cdot\tilde{\mathbf{u}}_\mu \mathrm{dV} = 0 \qquad \forall\, \tilde{\boldsymbol{u}}_{\boldsymbol{\mu}} \in \mathscr{V}_\mu \tag{3.20}$$

Since $\mathscr{V}_\mu$ is a vector space the variational Equation (3.20) only holds when

$$\int_{\Omega^s_\mu} \mathbf{b}.\eta \mathrm{dV} = 0 \quad \text{and} \quad \int_{\partial\Omega_\mu} \mathbf{t}.\eta \mathrm{dA} = 0 \qquad \forall\, \eta \in \mathscr{V}_\mu. \tag{3.21}$$

For the uniform boundary traction constraint $\mathscr{V}^{uni}$ imposed by the boundary of the RVE, body forces are relative to the constraints and are zero over the domain. In other words

$$\begin{gathered}\mathbf{b}(\mathbf{y}) = 0 \qquad \forall\, \boldsymbol{y} \in \Omega_\mu \\ \mathbf{t}(\mathbf{y}) = \boldsymbol{\sigma(y)}\mathbf{n}(\mathbf{y}) = \bar{\sigma}\mathbf{n}(\mathbf{y}) \qquad \forall\, \boldsymbol{y} \in \partial\Omega_\mu.\end{gathered} \tag{3.22}$$

Similarly the periodic boundary condition $\mathscr{V}^{per}$does not have body forces over the domain of the RVE and from Equation (3.21) anti-periodic traction $\mathbf{t}$ is obtained on the boundary as

$$\begin{gathered}\mathbf{b}(\mathbf{y}) = 0 \qquad \forall\, y \in \Omega_\mu \\ \mathbf{t}(\mathbf{y}) = -(\mathbf{y}^-) \qquad \forall\,\text{pairs}\,\{\boldsymbol{y}^+, \boldsymbol{y}^-\} \in \partial\Omega_\mu.\end{gathered} \tag{3.23}$$

For the case of linear boundary condition $\mathscr{V}^{lin}$ the boundary traction $\mathbf{t}$ are a response to $\mathscr{V}^{lin}$ and described earlier and the body forces over the domain is also zero. The Hill- Madel principle discussed in Equation (3.16) reduces Equation (3.19) to

$$\begin{gathered}\int_{\Omega^s_\mu} \boldsymbol{\sigma_\mu} : \boldsymbol{\nabla^s \eta} \mathrm{dV} = 0 \\ \forall\, \eta \in \mathscr{V}_\mu.\end{gathered} \tag{3.24}$$

Assuming that at any time t the stress at point y of the RVE is given by generic constitutive function $\mathscr{G}$ of strain history $\boldsymbol{\epsilon^t_\mu(y)}$ along with the equilibrium equation above, for a given macroscopic strain $\boldsymbol{\epsilon_m(t)}$ RVE equilibrium problem is defined such that a displacement fluctuation function $\tilde{\boldsymbol{u}}_{\boldsymbol{\mu}} \in \mathscr{V}_\mu$ hence

$$\begin{gathered}\boldsymbol{\sigma(y,t)} = \mathscr{G}(\boldsymbol{\epsilon^t_\mu(y)}) \\ \int_{\Omega^s_\mu} \mathscr{G}(\boldsymbol{\epsilon_m(t)} + \boldsymbol{\nabla^s \tilde{u}_\mu(y)})\boldsymbol{t} : \boldsymbol{\nabla^s \eta} \mathrm{dV} = 0 \qquad \forall\, \eta \in \mathscr{V}_\mu.\end{gathered} \tag{3.25}$$

The finite element approximation of periodic boundary conditions involves finding a discrete solution version consisting of finding a vector $\tilde{\boldsymbol{u}}_{\boldsymbol{\mu}} \in \mathscr{V}^h_{\mu'}$ of global nodal displacement fluctuations so that

$$\boldsymbol{G^h(\tilde{u}_\mu)} \equiv \left[\textstyle\int_{\Omega^{s,h}_\mu} \mathbf{B}^T\hat{\sigma}_y(\epsilon + \mathbf{B}\tilde{u}_\mu)\mathrm{dV}\right].\eta = 0 \quad \forall\, \eta \in \mathscr{V}^h_\mu. \tag{3.26}$$

where $\mathbf{B}$ is the global discrete gradient matrix $\boldsymbol{\epsilon}$ is the microscopic strain components and η is global nodal virtual displacement vector of the RVE and $\mathscr{V}^h_\mu$ donate finite dimensional space of virtual nodal displacement vectors of mesh h and microscopic domain domain Ω_μ^s. The arbitrary space of fluctuations and virtual displacement with periodicity on the boundary are defined as follows by Peric et al. [38]

$$\mathscr{V}_{\mu'}^h = \left\{ \boldsymbol{v} = \begin{bmatrix} v_i \\ v_+ \\ v_- \end{bmatrix} | v_+ = v_- \right\}, \tag{3.27}$$

Where v_i, v_+ and v_- represent vectors containing degrees of freedom of RVE interior and the portions Ω_+ and Ω_- of RVE boundary respectively. Uniform RVE boundary traction arbitrary space of fluctuations and virtual displacement are defined as follows by Peric et al. [38]

$$\mathscr{V}_{\mu'}^h \equiv \left\{ \boldsymbol{v} = \begin{bmatrix} v_i \\ v_b \end{bmatrix} | \int_{\Omega_\mu^h} \boldsymbol{N_b v_b} \otimes_{\boldsymbol{s}} \boldsymbol{n} \mathrm{dA} = 0 \right\} \tag{3.28}$$

Where $\boldsymbol{v_b}$ is boundary degrees of freedom $\mathbf{N}_b$ is the global interpolation matrix associated with the boundary nodes of the discretised RVE. The integral constraints $\boldsymbol{v_b}$ can also be expressed in matrix form as

$$\boldsymbol{C v_b = 0} \tag{3.29}$$

In which $\mathbf{C}$ is the constraint matrix. Where C is an elementary matrix which in 2D, for an element with m nodes on the intersection of the RVE boundary and the element boundary is given as

$$C^e = \begin{bmatrix} \int_{\Omega^e} \mathbf{N}_1^e n_1 dA & 0 & ... & \int_{\Omega^e} \mathbf{N}_m^e n_1 dA & 0 \\ 0 & \int_{\Omega^e} \mathbf{N}_1^e n_2 dA & ... & 0 & \int_{\Omega^e} \mathbf{N}_m^e n_2 dA \\ \int_{\Omega^e} \mathbf{N}_1^e n_2 dA & \int_{\Omega^e} \mathbf{N}_1^e n_1 dA & ... & \int_{\Omega^e} \mathbf{N}_m^e n_2 dA & \int_{\Omega^e} \mathbf{N}_m^e n_1 dA \end{bmatrix} \tag{3.30}$$

It is therefore prominent that $\boldsymbol{v_b}$ should be split in order to handle Equation (3.29) as

$$\boldsymbol{v_b} = \begin{bmatrix} \boldsymbol{v_f} \\ \boldsymbol{v_d} \\ \boldsymbol{v_p} \end{bmatrix} \tag{3.31}$$

where f, d and p are the free ,dependent and prescribed degrees of freedom of the boundary of the RVE respectively. Hence the global constraint matrix can be expressed as

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C_f} & \boldsymbol{C_d} & \boldsymbol{C_P} \end{bmatrix} \tag{3.32}$$

hence Equation (3.29) can be written as

$$\begin{bmatrix} \boldsymbol{C_f} & \boldsymbol{C_d} & \boldsymbol{C_P} \end{bmatrix} \begin{bmatrix} \boldsymbol{v_f} \\ \boldsymbol{v_d} \\ \boldsymbol{v_p} \end{bmatrix} = 0. \tag{3.33}$$

By removing the rigid body displacement in 3D we have 6 scalar equations and $3m - 6$ variables where m is the number of nodes on the boundary. hence the number of dependent variables is 6 therefore constraint equation can be reduced to

$$\boldsymbol{v_p = 0} \tag{3.34}$$

so that

$$\begin{bmatrix} \boldsymbol{C_f} & \boldsymbol{C_d} \end{bmatrix} \begin{bmatrix} \boldsymbol{v_f} \\ \boldsymbol{v_d} \end{bmatrix} = 0 \tag{3.35}$$

and after manipulation of Equation (3.35) $\boldsymbol{v_d}$ can be expressed in terms of $\boldsymbol{v_f}$ as

$$\boldsymbol{v_d} = \boldsymbol{\alpha v_f} \tag{3.36}$$

and

$$\boldsymbol{\alpha} = -\boldsymbol{C_d^{-1} C_f} \tag{3.37}$$

Taking the above into account, the discrete space of fluctuation and virtual displacement of the RVE are defined by

$$\mathscr{V}_{\mu'}^{h} = \left\{ \boldsymbol{v} = \begin{bmatrix} \boldsymbol{u_i} \\ \boldsymbol{u_f} \\ \boldsymbol{u_d} \end{bmatrix} | \boldsymbol{u_d} = \boldsymbol{\alpha v_f} \right\}. \tag{3.38}$$

Considering Equation (3.38) and splitting the corresponding vetors and tangental stiffness matrix as per the above analogy

$$\left\{ \begin{bmatrix} \mathbf{R}_i \\ \mathbf{R}_f \\ \mathbf{R}_d \end{bmatrix}^{k-1} + \begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{if} & \mathbf{k}_{id} \\ \mathbf{k}_{fi} & \mathbf{k}_{ff} & \mathbf{k}_{fd} \\ \mathbf{k}_{di} & \mathbf{k}_{df} & \mathbf{k}_{dd} \end{bmatrix}^{k-1} \begin{bmatrix} \delta\boldsymbol{\tilde{u}_i} \\ \delta\boldsymbol{\tilde{u}_f} \\ \boldsymbol{\alpha}\delta\boldsymbol{\tilde{u}_f} \end{bmatrix}^{k} \right\} \cdot \begin{bmatrix} \boldsymbol{\eta_i} \\ \boldsymbol{\eta_f} \\ \boldsymbol{\alpha\eta_f} \end{bmatrix} = 0 \quad \forall \eta_i, \eta_f \tag{3.39}$$

which results in

$$\begin{bmatrix} \mathbf{k}_{ii} & \mathbf{k}_{if} + \mathbf{k}_{id}\boldsymbol{\alpha} \\ \mathbf{k}_{fi} + \boldsymbol{\alpha^T}\mathbf{k}_{di} & \mathbf{k}_{ff} + \mathbf{k}_{fd}\boldsymbol{\alpha} + \boldsymbol{\alpha^T}\mathbf{k}_{df} + \boldsymbol{\alpha^T}\mathbf{k}_{dd} \end{bmatrix}^{k-1} \begin{bmatrix} \delta\boldsymbol{\tilde{u}_i} \\ \delta\boldsymbol{\tilde{u}_f} \end{bmatrix}^{k} = \begin{bmatrix} \mathbf{R}_i \\ \mathbf{R}_f + \alpha^T\mathbf{R}_d \end{bmatrix}^{k-1} \tag{3.40}$$

after some manupulations. For linear boundary displacement that is where degrees of freedom (fluctuations) of the boundary is zero the solutions follows the general linear solid mechanics problem. Elastic materials are those that deform reversibly under applied loads and return to their original shape once the load is removed. Therefore, the stress field depends on the small strain tensor, leading to the formulation of a constitutive function $\boldsymbol{\sigma_m}$ where

$$\boldsymbol{\sigma_m} = \boldsymbol{\sigma(\epsilon)} \tag{3.41}$$

From Equation (3.1) we get the boundary value problem an equation with kinematically admissible deformation $\eta \in \mathscr{K}_\mu$ as

$$\int_{\Omega_\mu^s} \boldsymbol{\sigma} : \boldsymbol{\nabla^s \eta} \mathrm{dV} - \int_{\Omega_\mu^s} \mathbf{b}.\eta \mathrm{dV} - \int_{\partial\Omega_\mu} \mathbf{t}.\boldsymbol{\eta} \mathrm{dA} = 0 \tag{3.42}$$
$$\forall \eta \in \mathscr{V}_\mu.$$

The set η and $\mathscr{V}_\mu$ are replaced by finite element discritisised subsets as explaind earlier in Section 2.6 Such that the boundary value problem is reduced to global internal and external force vectors $\mathbf{T}, \mathbf{F}$ respectively such that,

$$\mathbf{R} = \mathbf{T} - \mathbf{F} = 0 \tag{3.43}$$

where global forces are given by the finite elements where the virtual work for each element at node 1 element 1 is given by

$$\mathbf{T}_1^1 = \int_{v^1} \sigma \nabla \mathbf{N}_1 \mathrm{dV} \tag{3.44}$$
$$\mathbf{F}_1^1 = \int_{v^1} \mathbf{b}\mathbf{N}_1 \mathrm{dV} + \int_{\partial v} \mathbf{t}\mathbf{N}_1 \mathrm{dA}$$

Summation of all nodal forces $\{1, 2, 3, \ldots, n\}$ for all elements $\{1, 2, 3, \ldots, e\}$ containing node 1 and subsequently assembling these nodal forces to formulate the global virtual work equation, where

$$\mathbf{T} = \bigcup_{j=1}^{n} \sum_{i=1}^{e} \mathbf{T}_j^i \tag{3.45}$$
$$\mathbf{F} = \bigcup_{j=1}^{n} \sum_{i=1}^{e} \mathbf{F}_j^i$$

Newton-Raphson Solution is an iterative technique used to solve a system of non-linear equations due to non-linear materials like plastic material, elasto-plastic, visco-elastic and for large strain analysis, however when dealing with complex geometric shapes non linearity may also arise. This method works by providing the initial guess of the solution and correction factor to adjust the guess. As a result Equation (3.46) may results in a set of residuals as the solution converges quadratically.

$$\mathbf{K}^{(i)}_{(j+1)}\delta\mathbf{u}^{(i+1)}_{(j+1)} = -\mathbf{R}^{(i)}_{(j+1)} \quad \text{With solution :} \quad \mathbf{u}^{(i+1)}_{(j+1)} = \boldsymbol{u}^{(i)}_{(j+1)} + \delta\mathbf{u}^{(i+1)}_{(j+1)} \tag{3.46}$$

Where $\mathbf{K}$ is the global tangent stiffness matrix made up of nodal force change $\mathbf{T}^i_1$ and $\mathbf{F}^i_1$ from node 1 to $\mathbf{T}^i_2$ and $\mathbf{F}^i_2$ in node 2 which can be broken further into the constitutive $\mathbf{K}^i_{c_{12}}$ and initial stress $\mathbf{K}^i_{\sigma_{12}}$ components where,

$$\begin{aligned} \mathbf{K}^i_{12} &= \mathbf{K}^i_{c_{12}} + \mathbf{K}^i_{\sigma_{12}} \\ \mathbf{K}^i_{c_{12}} &= \int_v \nabla N_1 \mathbf{D} \nabla N_2 \mathrm{dV} \\ \mathbf{K}^i_{\sigma_{12}} &= \int_v \nabla N_1 \cdot \boldsymbol{\sigma} \nabla N_2 \mathrm{dV} \end{aligned} \tag{3.47}$$

Where N is the shape function associated with that node and $\mathbf{D}$ is the elastic matrix of material properties such that contribution of all node connections co $\{co_1, co_2, co_2, ...co_n\}$ is given by

$$\mathbf{K} = \bigcup_{co=1}^{n} \sum_{i=1}^{e} \mathbf{K}^i_{co} \tag{3.48}$$

For linear boundary displacement that is where degrees of freedom (fluctuations) of the boundary is zero the solutions follows the general linear solid mechanics problem, for periodic Boundary condition and minimally constrained models as explained by Peric et al. [38] the boundary condition of the RVE are non-conventional.

3.2 Automated Mesh Generation

For mesh auto-generation, a geometry file (`.geo`) like the one in Appendix [A.1] is created in `C++` based on the steps explained below by Figure (3.6).

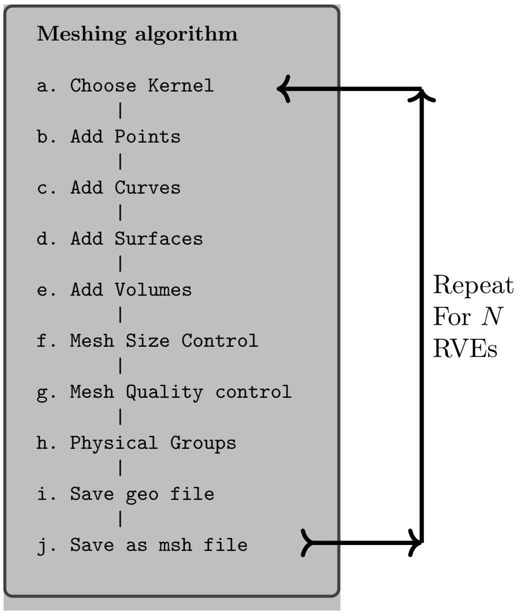


Figure 3.6: Algorithm Steps for Meshing

This geometry code file is constructed from a loop which varies a particular RVE parameter (like radii of voids) to generate different RVEs, each file is saved with a different name. One of the loops used for generating the RVE geometry in this study involves specifying the radii of spherical inclusions, with separate loops for voids and inclusions. For RVEs with long cylindrical fibres, an additional loop is employed to define both the fiber radius and the fibre spacing in the longitudinal and transverse directions. See the Appendix [C.2, C.3, C.4, C.5, C.6] for further details . The final stage is to get a mesh file (`msh`) for each RVE by systematically passing a geometry file to the `Gmsh` software, which then automatically generates the corresponding mesh file from the geometry. Geometry file is created using `SetFactory("OpenCASCADE")` kernel at the top of the file. This kernel provides libraries and tools for constructing and manipulating geometries. The geometry of the model is then built from `Points, Lines, Surfaces` as described by Figure 3.6 as per Geuzaine and Remacle [13].

`Point ( tg ) = {x, y, z, lc}` where lc represent the mesh size around that point and x, y, z represent the coordinates of the points. tg represents the tag number of the point. It's noteworthy to mention that the last entry lc, is optional and may be omitted, considering that there are alternative methods for controlling the mesh size, as we will explore later. Figure 3.7 shows how the points appears in the preview.

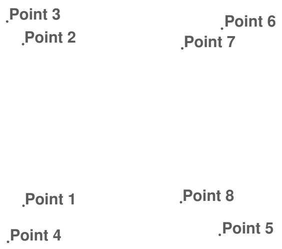
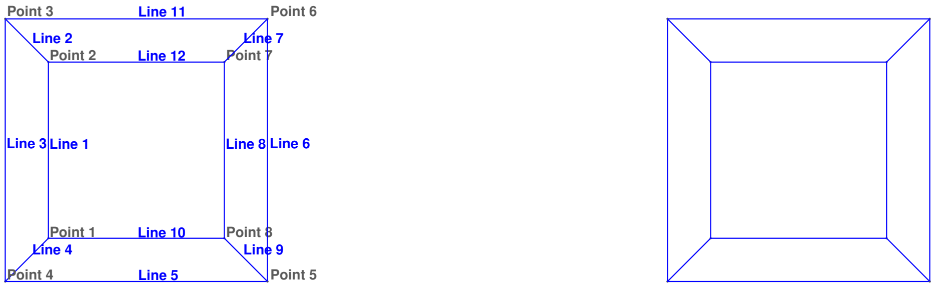


Figure 3.7: Edge Nodes of the RVE with their tag numbers

`Line ( tg ) = {`P_{tg1}`, `P_{tg2}`}` where tg represent the line tag number while P_{tg1} and P_{tg2} are the tag numbers of the two points connected. For example `line(1)={1,2}` connects Point 1 and 2 to form Line 1 in Figure 3.8a.



(a) Edges of the microscopic RVE

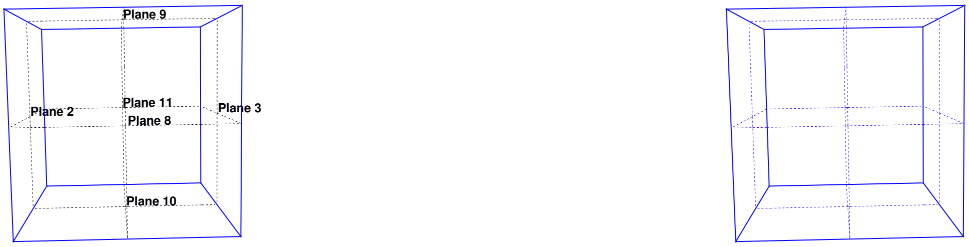
(b) Connected Edges of the RVE

Figure 3.8: Geometry generation of the RVE

`Circle ( tg ) = {x, y, z, ... }` and tg defines the circle arc tag number while x, y, and z represent the start point tag number, centre point tag number, and end point tag number, respectively. But if 4 or 6 expressions are given the first 3 will be the coordinates of the centre of the circle followed by the radius while the next 2 (which are optional) indicate the angles at the start and end.

`Curve Loop ( tg ) = { tag-list }` command create a closed loop where tg gives the tag number of the curve loop and tag-list provide a list of all curves to be used in a closed wire. This curve loop is used to create surfaces. In Figure 3.8 Lines 1 to 4 can be connected together as `Curve Loop(2) = {1, 2, 3, 4}` to form plane surface 2 in Figure 3.9. All lines must be converted to curve loops as demonstrated by Figure 3.8b .

`Plane Surface ( tg ) = {tag-list}`, this creates a plane surface where tag number tg is created from a list of curve loops. The first curve loop in the list describes the outer frame of the surface while the the following curve tags in the list are considered voids in this boundary. For a plane surface defining a void or inclusion, it should be defined separately since it has no common curves with the external surface. In Figure 3.9a `Plane Surface(2) = { 2 }` since the surface has no voids. Other plane surfaces of the RVE were created in a similar manner.



(a) Plane Surfaces of the RVE

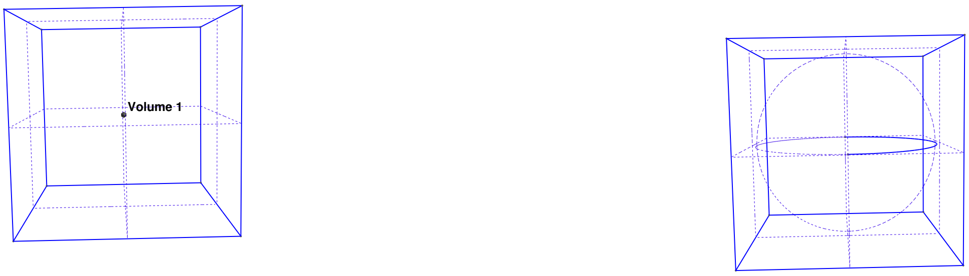
(b) RVE shell/wire

Figure 3.9: RVE Surface Generation

`Surface Loop ( tg ) = {tag-list}` create a shell for a surface loop with tag number tg from a list of plane surfaces to make a desired shell. Surface loops are in turn used to create volumes. Figure 3.9b was created by connecting all the plane surfaces in Figure 3.9a as `Surface Loop(12) = { 2, 3, 8 , 9, 10 , 11}`.

`Volume ( tg ) = {tag-list}` This command generate a volume with tag number tg from shells/ surface loop provided in the list. Similar to plane surfaces the first tag in the list represent the outer domain while the rest will represent voids for this volume. Volumes representing inclusions are defined separately. Figure 3.10a shows a volume created from surface loops.

`Sphere ( tg ) = { x, y, z, r...}` in which tg is the sphere tag number,x, y, and z are the centre coordinates of the sphere with radius r. Figure 3.10b represents an RVE with a spherical inclusion.



(a) Elementary Volume to be assigned Physical Name "matrix"

(b) RVE with a spherical inclusion

Figure 3.10: RVE Volumes

In Figure 3.10b The inclusion's surface loop is defined separately and given name tag 13 such that the matrix's volume is defined as Volume(2) = {12, 13} where the 13 is the sphere's surface loop tag number, 12 is the cube or the outer domain surface loop tag number.

`Cylinder ( tg ) = { x, y, z, dx, dy, dz, r...}` this command creates a cylinder with the centre coordinates of the first circular face as (x, y, z) followed by the 3 numbers dx, dy and dz defining its axis ,then the radius of the circular face. Extra expressions can be added or left out to describe the opening angle.

(a) Before Boolean Command

(b) After Boolean Command

Figure 3.11: Boolean operation for trimming the RVE

Boolean operations are used for creating intersections, unions, or differences as shown by Figure 3.11 where the command used `BooleanIntersection{Volume{tg};Delete;}{Vulume{tag} .......;Delete;}` was used to trim unwanted volumes. The first list is identified as the tool, while the second list is the object. With deletion, either the tool or the object can be removed, thereby creating new volumes, surfaces, or curves. This operation is used to trim unwanted surfaces or volumes. Mesh size control is crucial for determining the appropriate element size in finite element analysis. There are several ways the mesh size can be controlled, with five options explained fully by Geuzaine and Remacle [13].

One of the Gmash operation used in this chapter is `Mesh.MeshSizeFromPoint` and `Mesh.MeshSizeFromBoundary`. These involve specifying the desired mesh size at the desired nodes of the geometry. The mesh is then interpolated around the point or boundary. The `Mesh.MeshSizeFromCurvature` is one of the alternatives used in this study. This gives the number of elements per 2π radians based on the curvature of the underlying geometry. Normally it is set to zero by default in `Gmsh`. Using the `Mesh.MeshSizeExtendFromBoundary` operation the mesh sizes are determined by interpolating from the surfaces or volumes of the geometry. Other mesh control operations considered in this study involves using mesh size fields. This method specifies a general target mesh size which can be donated `Box`, `Distance`, `MathEval`, `PostView` and `Min`.

Point 2
Point 3
Point 6
Point 9
Point 10
Point 5
Point 4
Point 2
Point 3
Point 6
Point 9
Point 10
Point 5
Point 4

(a) Mesh size from point

(b) Mesh size field

Figure 3.12: 3.12a $lc = 0.1$ at point 2 and $lc = 0.05$ at point 4 and $lc = 0.2$ elsewhere, In Figure 3.12b `field[1]=Box` where the box is the RVE with a uniform mesh size 0.2

For rectangular or triangular meshes `Transfinite Line and extrusion` commad generates triangles by default and quadrangles after using `recombine`. The `Transfinite Line` command produces a structured mesh for a 4-sided shape after specifying how many grid points in the two opposite faces. A commad `Using Progression 1` can be used for uniform grading. Then each surface is specified to be a structured mesh using `Transfinite Surface` and then `recombine` is used to get quadrilaterals.

Point 2
Point 3
Point 6
int 9
Point 5
Point 4
Point 2
Point 3
Point 6
int 9
Point 5
Point 4

(a) Transfinite command

(b) Recombine command

Figure 3.13: Hexahedron Mesh

Mesh quality and optimisation help to smoothen out the mesh by removing ill-shaped elements, some of the commands used for quality control of the mesh include: `Mesh.ElementSizeFactor` which is used to control global element size factor relative to the geometry, `Mesh.CharacteristicLengthFactor` characteristic length of mesh element can be controlled using this commad. `Mesh.OptimiseQuality` is switched on or off to enable the `Gmsh` default to get improved elements hence improving quality. `Mesh.Smoothing`, using this command the number of iteration to improve element quality can be set. `Gmsh` software offers various meshing algorithms that can be selected to improve mesh quality, `Mesh.Algorithm` is used to choose from the likes of Delaunay, Frontal etc. There are other several ways to improve mesh quality, all of which are fully explained by Geuzaine and Remacle [13].

Elementary entities (spheres, lines, points, surface and volumes etc) are combined to get the required physical groups (inclusion, voids and matrix). All this provides the RVE domain required for the problem and all this is handled by `Gmsh`'s geometry module. However by `Gmsh` default some file formats such as mesh file only contain the elements that belongs to the specified Physical group. Therefore, voids can be created by omitting certain entities when defining physical groups..

3.3 Using RVEs

Dettmer's Mpap3 (Multi Physics Analysis Program) is utilised to handle an arbitrary number of 3D RVEs which uses the mesh generated by `Gmsh`. The Mpap3 interface allows for the meshes to be converted from 4-noded elements to 10-noded elements, hence improving the results further as 10-noded(quadratic elements) are better when accuracy and curved geometry representation matter while the 4-noded (linear) elements are faster but less accurate. Mpap3 uses the finite element method, and when small strains are applied to the RVE, the software generates the output stresses which can be used to predict the effective composite material elastic properties. Figure 3.16 and 3.15 demonstrate how the mesh size and order of an RVE can influence the results. It should be noted that the accuracy of the results is not crucial at this stage. This analysis aims to illustrate the significant impact that varying mesh sizes and order on the Young's modulus, providing insights into the sensitivity of the model to these parameters. Understanding these influences is essential for refining the modelling approach and optimising future simulations. Figure 3.14 shows the two meshes used to illustrate the influence of mesh size on the elastic properties.

Figure 3.15 shows the variation Young's modulus with mesh size as fibre volume ratio to the matrix increase while Figure 3.16 shows the variation of Poission's ratio with mesh size with is a comparison between 10-noded and 4-noded elements.

(a) Fine mesh

(b) Coarse mesh

Figure 3.14: A slice through a mesh of the RVEs with spherical inclusion used for experimentation

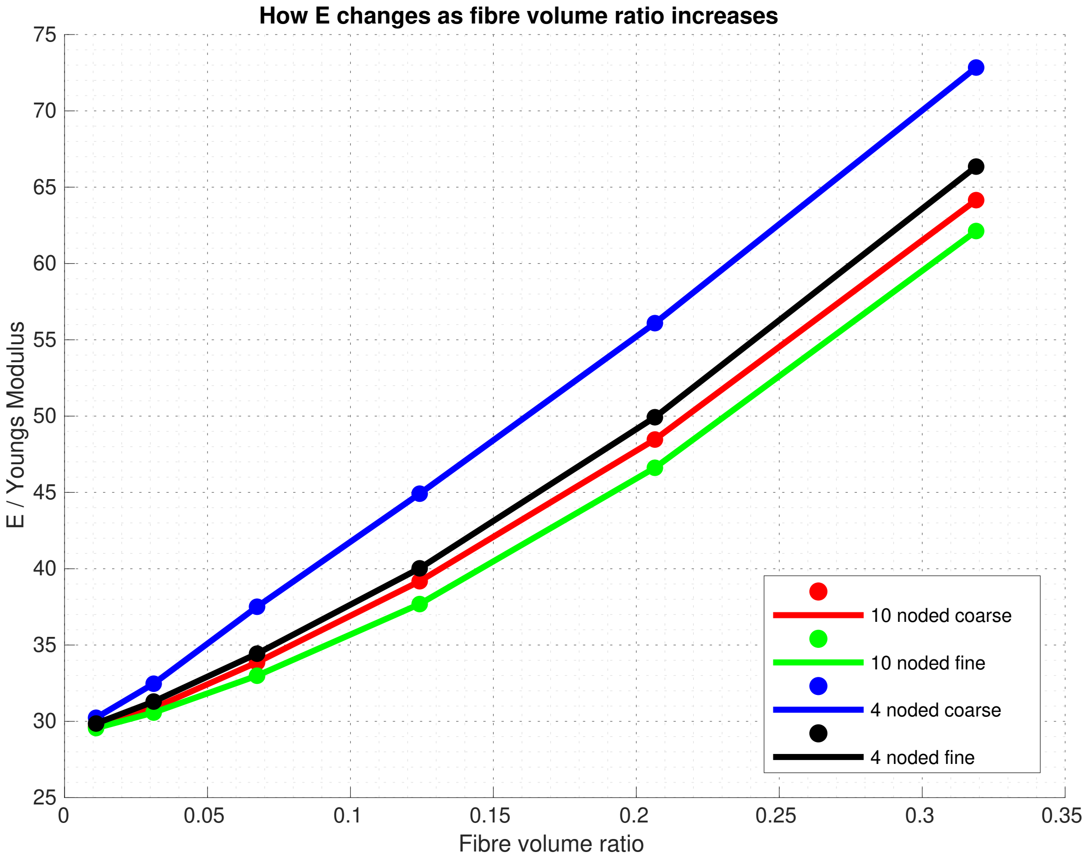


Figure 3.15: Linear boundary analysis of an RVE with spherical inclusion

Four noded linear elements tends to overestimate the values of the Young's modulus as illustrated by Figure 3.16 and underestimates the Poisson's ratio as shown by Figure 3.16. At very high fibre volume ratios the accuracy tends to decrease as compared to lower fibre volume ratios. For lower volume ratios both 4 noded and 10 noded tetrahedrons tends to give similar results. However a 10 noded coarse mesh with quadratic elements tends to be superior to a 4 noded fine mesh with linear elements. Since the fine 10 noded meshes generate more accurate results, a 10 noded element with fine mesh is used for the following computations throughout the study.

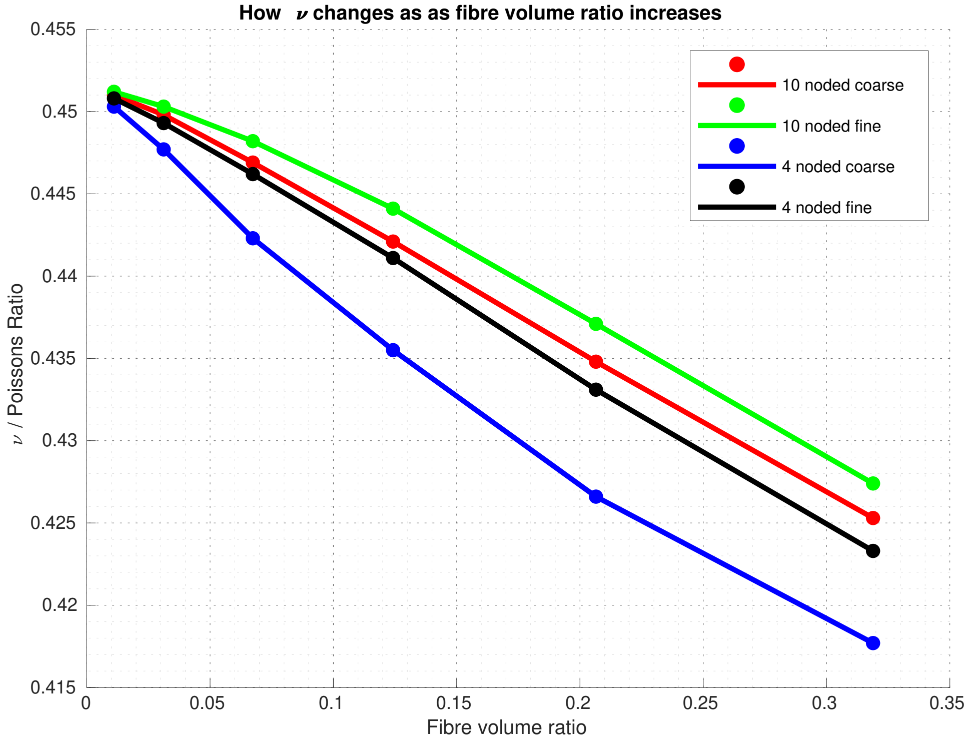


Figure 3.16: Linear boundary analysis of an RVE with spherical inclusion

3.4 Example 1: Homogeneous Block

A homogeneous block of material of bulk modulus K of $100GPa$ and Shear modulus $10GPa$ was used for this example. The block is subjected to five experiments and the Youngs modulus E and Poisson's ratio ν were calculated. That is five sets of different strain vectors $\boldsymbol{\epsilon}$ are used to generate the corresponding five set stresses $\boldsymbol{\sigma}$. 4-noded coarse and fine meshes and 10-noded coarse and fine meshes are used to compute $\boldsymbol{\sigma}$. The results are then compared in order to select the right mesh. Figure 3.17 shows the RVE used for experiments.

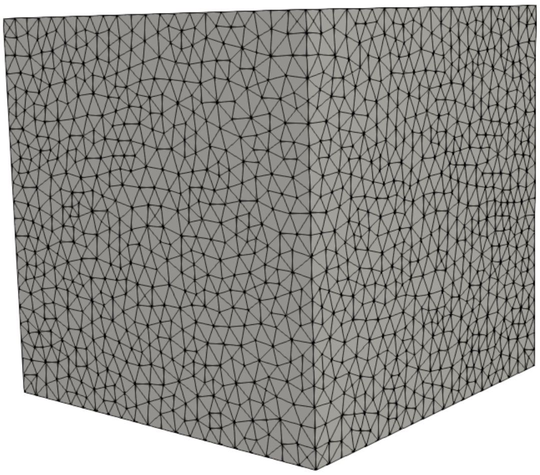


Figure 3.17: A mesh of a homogeneous RVE block

The homogeneous RVE block gave a value of $29.03GPa$ for Youngs modulus and Poisson's ratio of 0.4516, crucially this is the same values obtained from the general Hooke's law equation. The RVE produced same results for both fine and coarse meshes as well as for both 4 and 10 noded (linear and quadratic) elements. The size of the RVE does not affect the either the Young's modulus or Poisson's ratio.

3.5 Example 2: Spherical Inclusions

The study investigates only one type of inclusions, modelled as spheres with same radii embedded inside the matrix. The number of inclusions is varied hence a diverse range of Representative Volume Elements (RVEs) of increasing size are generated. Each RVE undergoes five distinct experiments that is each RVE is exposed to five different strains, and a total of six RVEs were considered. The

manipulation of the size of the cube allows for precise control of inclusion volume ratio which was kept constant throughout this experiment. It's noteworthy that inclusions enhance the composites' strength. In this experiment, inclusions are assigned higher bulk and shear moduli compared to the matrix. For this part of research, we focus on six key parameters, inclusion radius, and center-to-center spacing, as illustrated in Figure 3.18. Additionally, we set specific values for the remaining four parameters, matrix shear modulus G_m and bulk modulus K_m were assigned values of $10GPa$ and $100GPa$, respectively, while inclusion shear modulus G_f and bulk modulus K_f were assigned values of $100GPa$ and $1000GPa$, respectively.

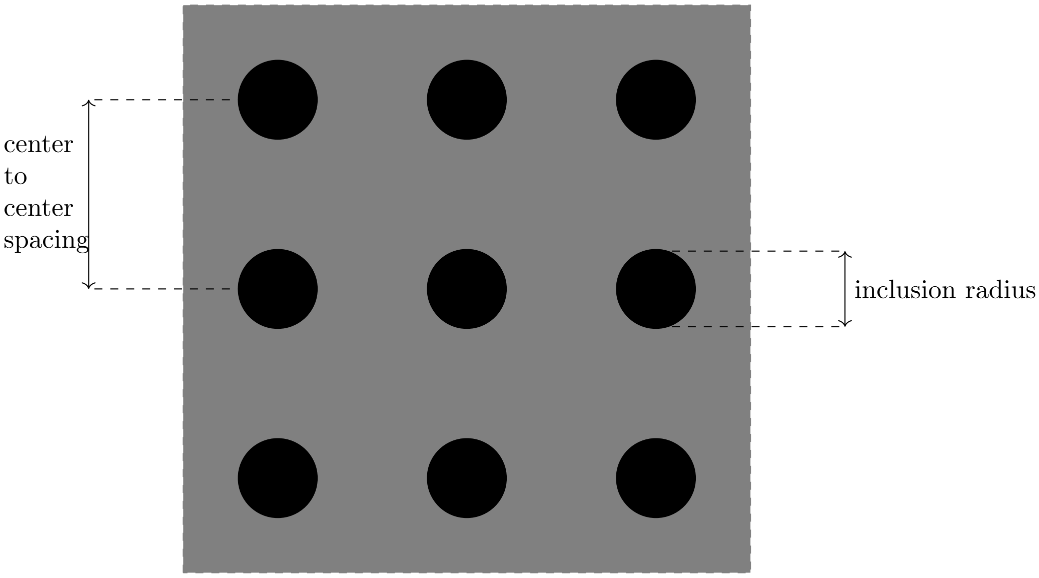
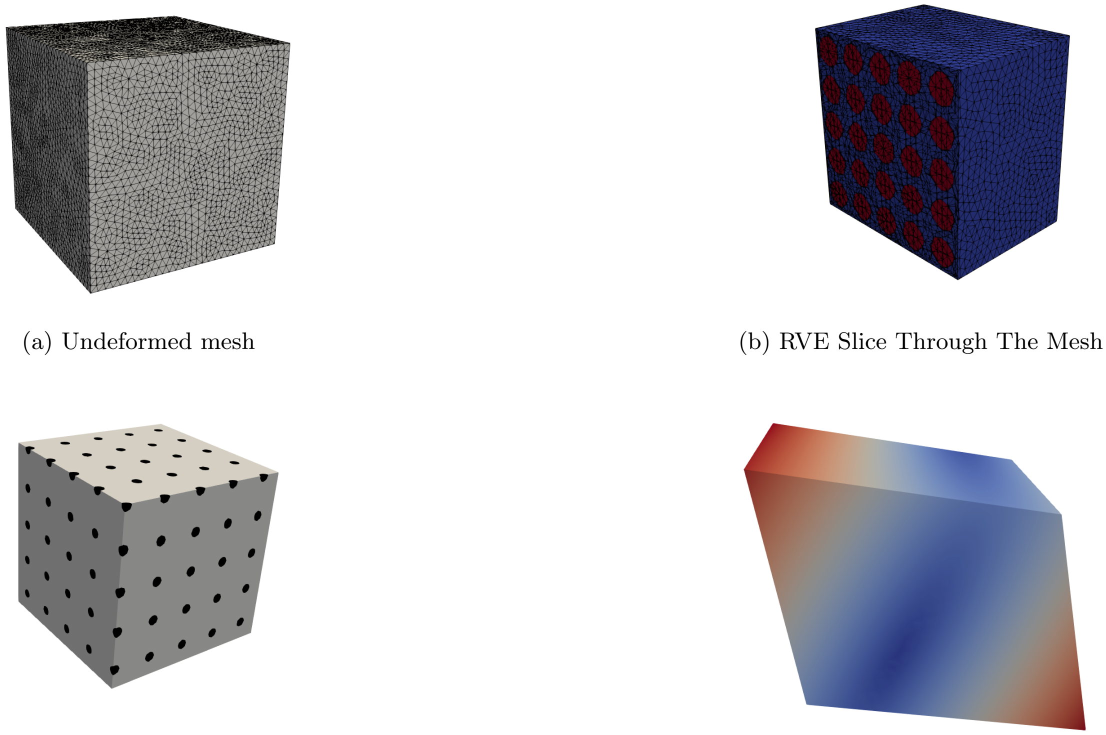


Figure 3.18: Parameters of interest from the RVE.

Figure 3.19 below shows RVE models for spherical inclusions in a matrix used in this experiment. Figure 3.20 illustrates the application of linear analysis to study the variation in elastic properties as the Representative Volume Element (RVE) size increases, focusing on spherical inclusions. The graphs reveal that larger RVEs yield larger overage stresses consequently more accurate results while using the linear boundary condition. The convergence is quicker, and the graph remains nearly horizontal with larger RVEs. However, it is essential to note that larger RVEs come at a cost, as the calculations become more time-consuming.

It takes about 24 hours to run a larger RVE that for a 10-noded element for five experiments. Therefore, achieving a balance is crucial in optimising computational efficiency. From the graphs in Figure 3.20 , Figure 3.20b depict the variations in Poisson's ratio ν with increasing inclusions. In this experiments, the volume ratio of fibres was maintained at a constant value of 0.053.
Figure 3.20 shows how the size of the RVE affects the elastic properties of the composite.

It can be inferred that the Poissoin's ratio values increase and eventually plateau as the Representative



(a) Undeformed mesh

(b) RVE Slice Through The Mesh

(c) Inclusion Arrangement After Cutting

(d) Deformed shape

Figure 3.19: The RVE of a isotropic block with spherical inclusions

Volume Element (RVE) size increases. Regarding Figure 3.20*a*, 3.20*c* and 3.20*d*, which represent Young's modulus E, Shear modulus G and Bulk Modulus K respectively, the values start at a higher magnitude and decrease before levelling off at a lower values. Consequently, it can be concluded that larger RVEs can yield more accurate elastic properties for this specific experimental setup.

As the RVE increases in size the results improve because a small one may include few inclusions not enough to represent the microstructure, as RVE grows it includes more heterogeneities giving a more statistically representative sample of the material. Smaller RVEs are strongly influenced by boundary conditions while larger RVEs reduce this impact on the overall response so that the results approach bulk behaviour. For Larger RVEs local fluctuations average out hence the calculated effective properties tend to converge towards the true homogenised value of the material. In general increasing the number of macrostructural features inside the RVE makes average response more stable and less noisy hence improving accuracy.

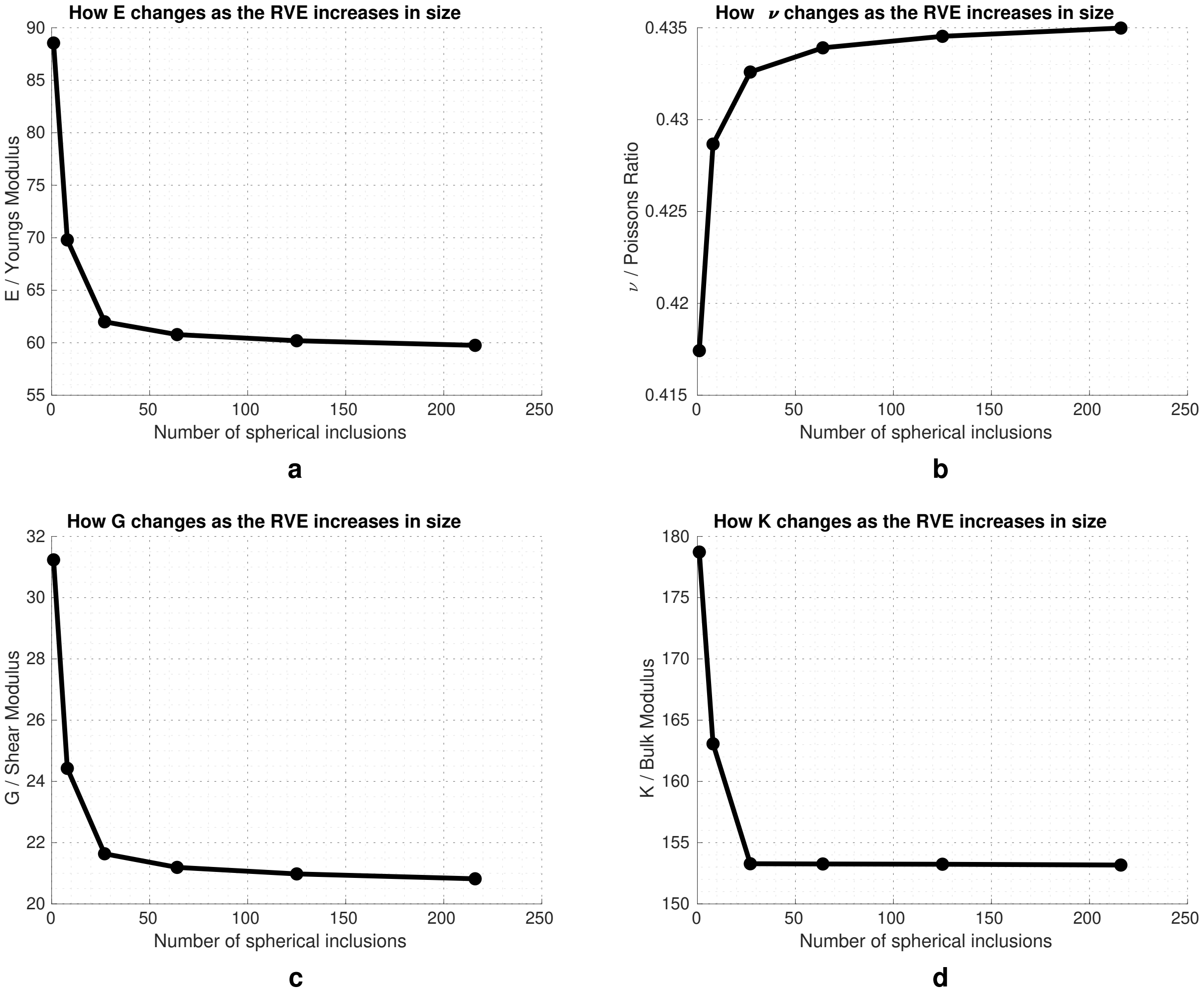


Figure 3.20: How increasing RVE size by increasing number of spherical inclusions in the RVE affect mechanical properties

3.6 Example 3: Matrix and Fibres

This example explores long cylindrical isotropic inclusions that extend throughout the entire RVE. These lead to a transversely isotropic composite, with inclusions modelled as cylinders with the same radii. The number of inclusions is varied, resulting in different Representative Volume Elements (RVEs). Each RVE undergoes five distinct experiments, with a total of six RVEs considered. The inclusion volume ratio is kept constant throughout this example. No voids are considered in this experiment, and the inclusions are assigned higher bulk and shear moduli compared to the matrix.

For this transversely isotropic RVE, we focus on six key parameters namely the inclusion radius and center-to-center spacing, as previously illustrated in the diagram in Figure 3.18. Matrix shear modulus G_m and bulk modulus K_m were assigned values of $10GPa$ and $100GPa$, respectively, while inclusion shear modulus G_f and bulk modulus K_f were assigned values of $100GPa$ and $1000GPa$, respectively. Figure 3.21 below illustrates the RVE model for cylindrical inclusions with the matrix.

Figure 3.22 investigation into the Influence of RVE size on Elastic Properties. From Figure 3.22, Figure 3.22a, 3.22b, 3.22c and 3.22d represent the variations in elastic properties namely the longitudinal Young's modulus E_1, transverse Young's modulus E_2, longitudinal shear modulus G_{12}, and transverse shear modulus G_{23} with an increasing RVE size, respectively. In this example, a constant fibre volume ratio of 0.373 was maintained. For these first four Figures 3.22a, 3.22b, 3.22c and 3.22d, it can be deduced that both Young's modulus and shear modulus exhibit an initial faster decrease from higher values, followed by a slower decrease, eventually levelling out as the RVE size increase.

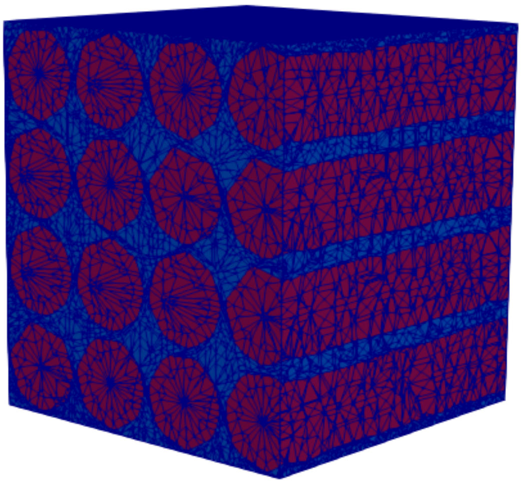


Figure 3.21: Slice through the RVE

This observation aligns with the notion that larger RVEs contribute to more accurate results. However there is last point in 3.22b appears to be slightly above which may be due to some calculation errors. As for 3.22e representing longitudinal Poisson's ratio the graph remains constant as the RVE size has no impact on the Poisson's ratio in longitudinal direction because the response in this direction is dominated by the continuous fibres within the matrix which carry most of the load and dominates the longitudinal behaviour. Increasing the RVE add more repeating microstructure, but the longitudinal behaviour is already fully captured even in a small RVE. Figure 3.22f, representing transverse Poisson's ratios ν_{13} indicates that the values start at lower magnitudes and gradually increase, forming a plateau. Consequently, it can be deduced that larger Representative Volume Elements (RVEs) can produce more accurate and better elastic properties for this specific experimental setup.

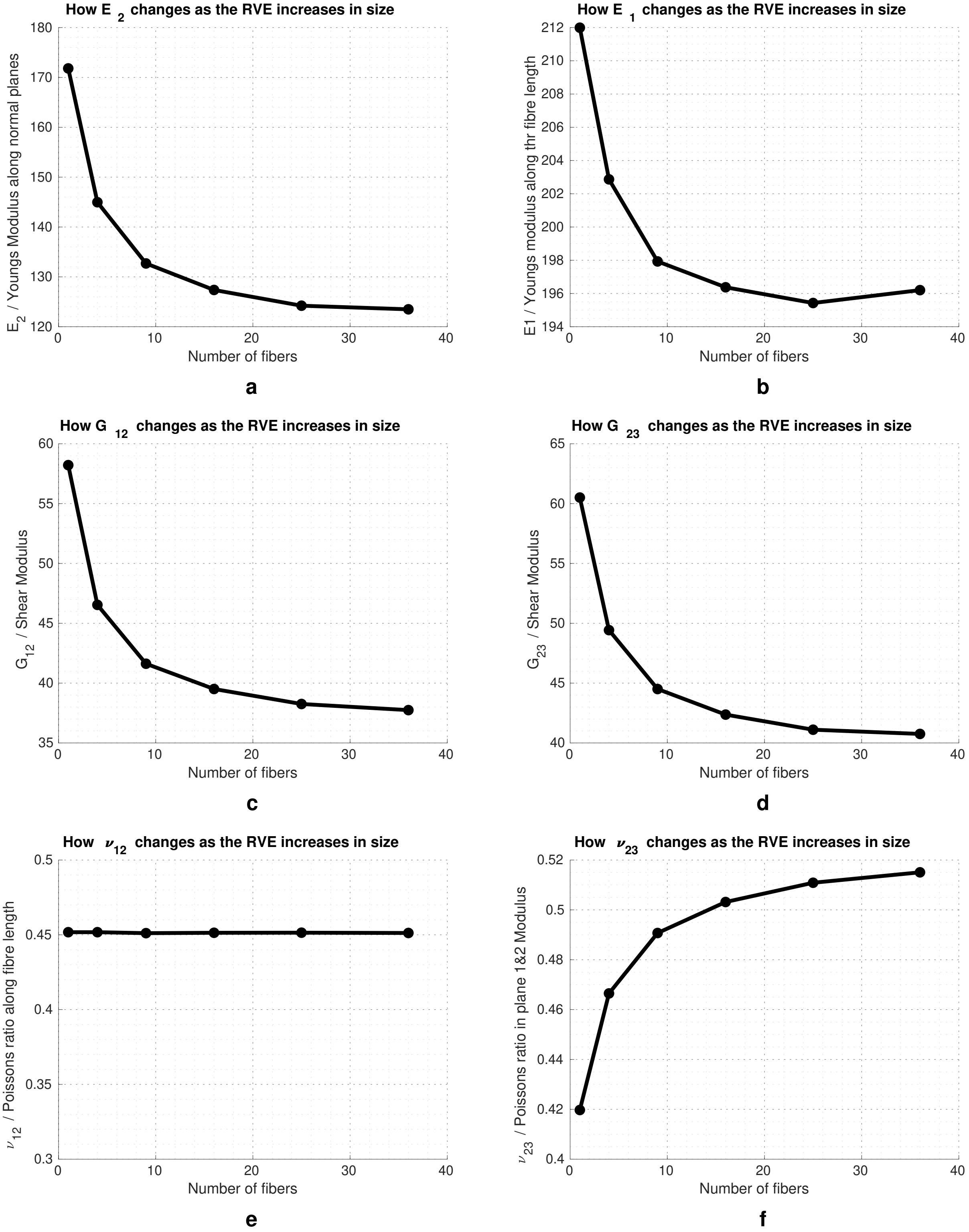


Figure 3.22: Linear boundary analysis of an RVE with cylindrical inclusions

Chapter 4

Homogenisation

This chapter focuses on treating materials within a composite with different constituents or similar constituents as a homogeneous material with effective properties.This process determines the macroscopic or effective elastic properties such as Young's modulus E and Poisson's ratio ν which are based on the distribution and properties of the microscopic constituents. This process tend to simplify the analysis and design of materials. Homogenisation may in general include determining effective thermal conductivity or electrical conductivity however in this chapter the focus is only on effective elastic properties. The method used in this chapter uses the Representative Volume Element (RVE) sample of the material together with FE method to develop homogenisation technique for the overall elastic properties using the least square method to solve equations in Chapter 2.

Various models have been developed previously to predict macroscopic composite elastic properties empirically such as those explained by Daniel and Ishai [7], Halpin and Kardos [17].The methods uses fibre volume fraction, matrix and fibre properties and ignores the shape and distribution in the RVE hence better predictions are made in the longitudinal direction but they underestimate the transverse properties Daniel and Ishai [7]. Huang et al. [20], Sun and Vaidya [46] indicated that the prediction of transverse properties using these empirical models produces scattered results. These methods normally use regular shapes to make predictions even though the use of irregular fibres is becoming more common. The most commonly used technique to determine the effective elastic properties of composite is through experimental methods which is expensive and difficult since mechanical properties depend on geometry therefore finite element method has been used as an alternative method as varies geometries can be analysed quicker and simpler Huang et al. [20].

The method implemented in this chapter is an iterative method in the MATLAB built-in background algorithm to approximate the solution. The values of the Young's modulus and the Poisson's ratio are a good approximation provided the residual is greatly reduced. Least square method provides accurate results, can handle larger data sets and the implementation is easier using MATLAB package than other programming languages. The method is quicker than other homogenisation methods which are already in use and very easy to understand.

The MATLAB function used in this chapter is an already developed built in function in MathWorks. The function used, `lsqr`, is not looked into deeper details in this study rather a summary of an algorithm used to solve equations of the type $Ax = rhs$ is discussed briefly before making use of the built in function. The equations described in Chapter 2 are rearranged and expressed in the in the form $Ax = rhs$ to solve for the unknown parameters Poisson's ratio ν and Young's modulus E which are constants in the elements of vector x. rhs represents a vector containing the strains ϵ and A represent a matrix constructed from stresses σ output of the corresponding strains from FE method. Matlab scripts files are then developed for the isotropic and transverse isotropic material to compute the effective Youngs modulus E and the Poisson's ratio ν of the macrostructure in longitudinal and transverse directions. Figure 4.1 shows steps followed towards this homogenisation.

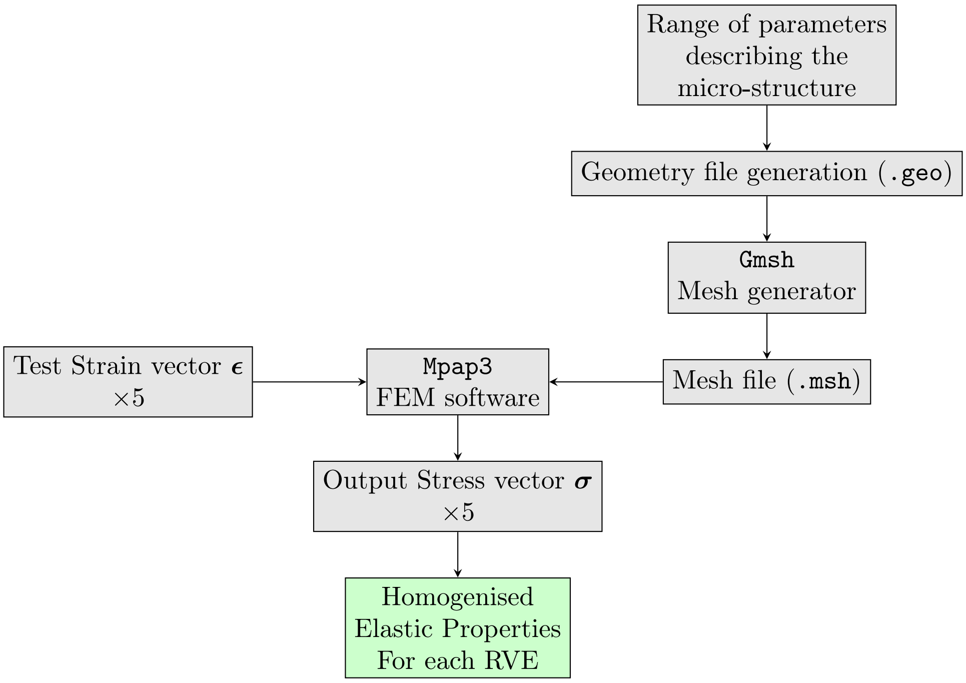

Range of parameters describing the micro-structure
Geometry file generation (.geo)
Gmsh
Mesh generator
Test Strain vector $\boldsymbol{\epsilon}$
×5
Mpap3
FEM software
Mesh file (.msh)
Output Stress vector $\boldsymbol{\sigma}$
×5
Homogenised Elastic Properties For each RVE

Figure 4.1: Homogenisation Methodology Utilized

4.1 Isotropic Elasticity

From Equation (2.38), it follows that the stress-strain relation for an isotropic material can be expressed as

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{E} & -\frac{v}{E} & -\frac{v}{E} & 0 & 0 & 0 \\ -\frac{v}{E} & \frac{1}{E} & -\frac{v}{E} & 0 & 0 & 0 \\ -\frac{v}{E} & -\frac{v}{E} & \frac{1}{E} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G} \end{bmatrix} \times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}. \tag{4.1}$$

From Equation (4.1) above, we obtain six equations to solve for two unknowns. Therefore, only one experiment should be sufficient. The following relations are obtained

$$\begin{aligned} \varepsilon_1 &= \sigma_1 \frac{1}{E} - \sigma_2 \frac{\nu}{E} - \sigma_3 \frac{\nu}{E} \\ \varepsilon_2 &= \sigma_2 \frac{1}{E} - \sigma_1 \frac{\nu}{E} - \sigma_3 \frac{\nu}{E} \\ \varepsilon_3 &= \sigma_3 \frac{1}{E} - \sigma_1 \frac{\nu}{E} - \sigma_2 \frac{\nu}{E} \\ \gamma_4 &= 2\tau_4 \frac{1+\nu}{E} \\ \gamma_5 &= 2\tau_5 \frac{1+\nu}{E} \\ \gamma_6 &= 2\tau_6 \frac{1+\nu}{E}. \end{aligned} \tag{4.2}$$

Equations (4.2) can be rewritten further as

$$\begin{aligned}
\varepsilon_1 &= \sigma_1\frac{1}{E} - [\sigma_2+\sigma_3]\frac{\nu}{E} \\
\varepsilon_2 &= \sigma_2\frac{1}{E} - [\sigma_1+\sigma_3]\frac{\nu}{E} \\
\varepsilon_3 &= \sigma_3\frac{1}{E} - [\sigma_1+\sigma_2]\frac{\nu}{E} \\
\gamma_4 &= 2\tau_4[\frac{1+\nu}{E}] \\
\gamma_5 &= 2\tau_5[\frac{1+\nu}{E}] \\
\gamma_6 &= 2\tau_6[\frac{1+\nu}{E}].
\end{aligned} \tag{4.3}$$

Hence, we can generate a 6×2 matrix A for which the columns are multiplied by a vector made of coefficients containing $\frac{1}{E}$ and $\frac{\nu}{E}$respectively, which can be used together with the `lsqr` method after several experiments to solve for ν and E. Least Square QR (LSQR) is a mathematical technique used to solve linear least squares problems. The method is well-suited for large, sparse linear systems where traditional direct methods may not be suitable due to memory constraints. This iterative method works by minimising the 2-norm residuals at each step. This system of equations takes the form

$$\begin{bmatrix} \sigma_1 & \sigma_2+\sigma_3 \\ \sigma_2 & \sigma_1+\sigma_3 \\ \sigma_3 & \sigma_1+\sigma_2 \\ \tau_4 & \tau_4 \\ \tau_5 & \tau_5 \\ \tau_6 & \tau_6 \end{bmatrix} \times \begin{bmatrix} \frac{1}{\mathbf{E}} \\ \\ \frac{\nu}{\mathbf{E}} \end{bmatrix} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} \tag{4.4}$$

which can also be written as

$$Ax = rhs. \tag{4.5}$$

where x is a vector of unknowns relating to input vector rhs, and A is a matrix made of known outputs, the LSQR algorithm tries to solve the Equation (4.5) for x using least squares to minimise the normal residual r if A is consistent. Otherwise, the method solves a least squares solution for x that minimises $\frac{\|rhs-ax\|}{\|rhs\|}$ after several iterations for inconsistent coefficients of A. A consistent system has a solution while inconsistent system has no solution in the column space of A.

$$r = rhs - Ax \tag{4.6}$$

$$minimise\frac{\|rhs - Ax\|}{\|rhs\|} \tag{4.7}$$

From the `[x, flag, r] = lsqr(A, rhs)` in MATLAB codes above x is the solution vector representing the solution of unknowns in Equation (4.4), `flag` is used to show convergence status of `lsqr` algorithm while r is the residual. The matlab built in function `lsqr` uses the algorithm which is fully explained by Paige and Saunders [36]. Algorithm 1 have been borrowed from Paige and Saunders [36] and these are the steps followed by the MATLAB built in function `lsqr` to solve Equation (4.5). This chapter does not focus on full details of the procedures, rather a summary of the algorithm is provided . Paige and Saunders [36], MathWorks [31], Barrett et al. [2] provided a more detailed methodological protocol .

From Algorithm 1, v is a sequence of vectors and α and β are scalars such that $\mathbf{A}$ is a triagonal form. Paige and Saunders [36] explained that this scalars are chosen such that $\alpha_i \geq 0$ and $\beta_i \geq 0$ such that $\|u_i\| = \|v_i\| = 1$ where $\alpha_1 v_1 = \mathbf{A^T} u_1$ means $\bar{v}_1 = \mathbf{A}^T u_1$, $\alpha_1 = \|\bar{v}_1\|$ and $v_1 = (1/\alpha_1)\bar{v}_1$. rhs is the the starting vector and is used with $\mathbf{A}$ in the first bidiagonalisation to generate quantities which will be used to solve $\min\|rhs - Ax\|$. This quantities eventually lead to a natural least square problem $\min\|\beta_1 e_1 - B_k y_k\|$ which is explained in details by Paige and Saunders [36]. The algorithm is continued until a certain tolerance achieved where

Algorithm 1 `lsqr` algorithm

```
1: Initialise
2: $\beta_i u_i = rhs, \alpha_1 v_1 = \mathbf{A}^T u_1, w_1 = v_1, x_0 = 0, \bar{\rho}_1 = \alpha_1, \bar{\phi}_1 = \beta_1$.
3: For $i = 1, 2, 3$,....repeat steps 4-
4: Continue Bidiagonalization.
5: $\beta_{i+1} u_{i+1} = \mathbf{A} v_1 - \alpha_1 u_1$
6: $\alpha_{i+1} v_{i+1} = \mathbf{A}^T u_{i+1} - \beta_{i+1} v_i$
7: Construct and apply next orthogonal transformation
8: $\rho_i = (\rho_i^2 + \beta_{i+1}^2)^{\frac{1}{2}}$
9: $c_i = \frac{\bar{\rho}_i}{\rho_i}$
10: $s_i = \frac{\beta_{i+1}}{\rho_i}$
11: $\theta_{i+1} = s_i \alpha_{i+1}$
12: $\bar{\rho}_{i+1} = -c_i \alpha_{i+1}$
13: $\phi_i = c_i \bar{\phi}_i$
14: $\bar{\phi}_{i+1} = s_i \bar{\phi}_i$
15: Update x and w
16: $x_i = x_{i-1} + (\frac{\phi_i}{\rho_i}) w_i$
17: $w_{i+1} = v_{i+1} - (\frac{\theta_{i+1}}{\rho_i}) w_i$
18: EXIT
```

$$tol = \frac{\|rhs - Ax\|}{\|rhs\|} \tag{4.8}$$

Normally the MATLAB default's tolerance is set to 1×10^{-6} however this can be adjusted to improve the accuracy. From the experiments where the material is subjected to strains and out comes the stresses, matrix $\mathbf{A}$ is constructed from the stresses σ and the rhs is constructed from the strains ϵ values hence the values of the Young's modulus E and poisson's ratio ν can be approximated using this method above. However this algorithm is already built in MATLAB functions (`lsqr`) hence a script file is developed to carry out the computations in MATLAB with stresses and strains as inputs data from the experiments. Although to completely solve this problem one experiment might be enough, five experiments that is five sets of stress and strains vectors were used to further improve the accuracy. This was done to average out random errors to better estimate the true value. Carrying out more than one experiment help to detect and handle outliers hence improving reliability.

Using the MATLAB command `lsqr`, the best approximation of ν and E can be found by solving Equation (4.1) for the lowest normal residual. The solution is achieved by providing strains and their corresponding output stresses from experiments carried out using the finite element method and using those results as input data. This process is summerised in the MATLAB code file in Appendix B.1.

4.2 Transversely Isotropic Elasticity

For a transversely isotropic material, $E_2 = E_3$, $G_{12} = G_{13}$ and $\nu_{12} = \nu_{13}$ therefore equation (2.38) provides the stress-strain relation

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{E_1} & -\frac{v_{12}}{E_2} & -\frac{v_{12}}{E_2} & 0 & 0 & 0 \\ -\frac{v_{12}}{E_1} & \frac{1}{E_2} & -\frac{v_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{v_{12}}{E_1} & -\frac{v_{23}}{E_2} & \frac{1}{E_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{12}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix} \tag{4.9}$$

where $G_{23} = \frac{E_2}{2(1-\nu_{23})}$, from Equation (4.9), we obtain six equations to solve for six unknowns.Therefore, a minimum of one experiment sufficient. Expanding Equation (4.9) the relations

$$
\begin{aligned}
\varepsilon_1 &= \sigma_1 \frac{1}{E_1} - \sigma_2 \frac{\nu_{12}}{E_2} - \sigma_3 \frac{\nu_{12}}{E_2} \\
\varepsilon_2 &= \sigma_2 \frac{1}{E_2} - \sigma_1 \frac{\nu_{12}}{E_1} - \sigma_3 \frac{\nu_{23}}{E_2} \\
\varepsilon_3 &= \sigma_3 \frac{1}{E_2} - \sigma_1 \frac{\nu_{12}}{E_1} - \sigma_2 \frac{\nu_{23}}{E} \\
\gamma_4 &= \frac{2\tau_4}{G_{23}} \\
\gamma_5 &= \frac{2\tau_5}{G_{12}} \\
\gamma_6 &= \frac{2\tau_6}{G_{12}}
\end{aligned}
\tag{4.10}
$$

are obtained. Equations in (4.10) can be rewritten further as

$$
\begin{aligned}
\varepsilon_1 &= \sigma_1 \frac{1}{E_1} - [\sigma_2 + \sigma_3] \frac{\nu_{12}}{E_2} \\
\varepsilon_2 &= \sigma_2 \frac{1}{E} - \sigma_1 \frac{\nu_{12}}{E_1} + \sigma_3 \frac{\nu_{23}}{E_2} \\
\varepsilon_3 &= \sigma_3 \frac{1}{E_3} - \sigma_1 \frac{\nu_{12}}{E_1} + \sigma_2 \frac{\nu_{23}}{E_2} \\
\gamma_4 &= \frac{2\tau_4}{\frac{E_2}{2(1-\nu_{23})}} \\
\gamma_5 &= \frac{2\tau_5}{G_{12}} \\
\gamma_6 &= \frac{2\tau_6}{G_{12}}
\end{aligned}
\tag{4.11}
$$

hence, we can generate a 6×5 matrix A where the columns represent coeficients to $\frac{1}{E_1}$, $\frac{1}{E_2}$, $\frac{\nu_{12}}{E_1}$, $\frac{\nu_{23}}{E_1}$, and $\frac{1}{G_{12}}$ respectively contained in each equation. This matrix can be used together with the `lsqr` function after an experiment to solve for $\nu_{12}, \nu_{23}, E_1, E_2, G_{12}$, and G_{23}. Therefore Equation (4.9) can be rearranged as

$$
\begin{bmatrix}
\sigma_1 & 0 & \sigma_2 + \sigma_3 & 0 & 0 \\
0 & \sigma_2 & \sigma_1 & \sigma_3 & 0 \\
0 & \sigma_3 & \sigma_1 & \sigma_2 & 0 \\
0 & 2\tau_4 & 0 & -2\tau_4 & 0 \\
0 & 0 & 0 & 0 & \tau_5 \\
0 & 0 & 0 & 0 & \tau_6
\end{bmatrix}
\times
\begin{bmatrix}
\frac{1}{\mathbf{E}_1} \\
\frac{1}{\mathbf{E}_2} \\
\frac{\nu_{12}}{\mathbf{E}_1} \\
\frac{\nu_{23}}{\mathbf{E}_2} \\
\frac{1}{\mathbf{G}_{12}}
\end{bmatrix}
=
\begin{bmatrix}
\varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_4 \\ \gamma_5 \\ \gamma_6
\end{bmatrix}
\tag{4.12}
$$

Equation (4.12) can also be written in the form

$$
Ax = rhs. \tag{4.13}
$$

Using the MATLAB command `lsqr`, the best approximation of ν_{12}, ν_{23}, E_1, E_2, G_{12}, and G_{23} can be found by solving Equation (4.12) for the lowest normal residual. Appendix B.2 summarises this computations in a MATLAB code.

4.3 Analytical Models

According to Daniel and Ishai [7], analytical method try to describe the overall/effective elastic properties of an RVE as a function of its geometry and its inclusion's elastic properties. Several methods have been developed the majority of which are good in predicting longitudinal properties however these methods may not be good in transverse directions as they do not take into account the shape and distribution of fibres in this direction. This section uses models developed by Daniel and Ishai [7] . Equation (4.14) expresses the effective property of a composite as function its constituents that is

$$C^* = f(C_F, C_m, V_f, S, A) \tag{4.14}$$

where

C^*	is the average composite stiffness
C_f, C_m	is the fiber and matrix stiffness respectively
V_f	is fiber volume ratio
S, A	is the geometric parameters describing the shape and alignment of the fibres respectively.

The following elastic properties were analysed using this method.

E_1, E_2, E_3	Young's moduli
G_{12}, G_{23}, G_{32}	shear moduli
v_{12}, v_{23}, v_{32}	Poisson's ratio

Parallel model (Voigt) and series model (Reuss) are the approaches used in the mechanics of materials to describe the elastic properties of composites with the assumption that both the matrix and inclusion undergo uniform strain. Hence parallel model with the relation

$$C^* = V_f C_f + V_m C_m \tag{4.15}$$

is obtained where C^*, C_f, C_m is the composite, fibre and matrix stiffnesses respectively. In compliance form

$$S^* = V_f S_F + V_m S_m \tag{4.16}$$

where S^*, S_f, S_m is the compliances, composite fibre and matrix stiffness respectively. Since the compliance matrix is the inverse of the stiffness then it is true to assume $S^* = \frac{1}{C^*}$. Hence we are left with series model where

$$C^* = \frac{1}{\frac{V_f}{C_f} + \frac{V_m}{C_m}}. \tag{4.17}$$

In practice stresses and strains in a composite material are not uniform therefore upper and lower bounds of the stiffness are normally used as the solution lies between these bounds. As a results the following limits as provided by Daniel and Ishai [7] are used

$$\frac{1}{\frac{V_f}{C_f} + \frac{V_m}{C_m}} \leq C^* \leq V_f C_f + V_m C_m. \tag{4.18}$$

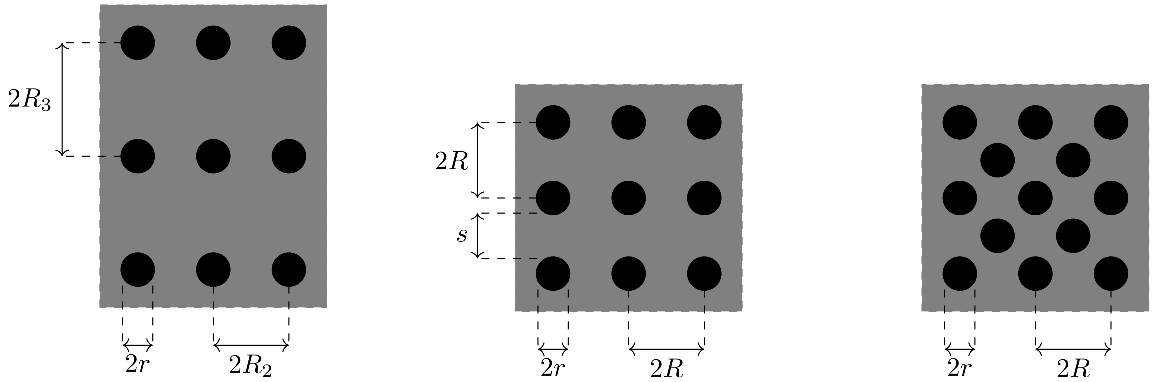
Halpin Tsai relations interpolate between lower and upper bounds. The Halpin Tsai equations are based on self-consistent micro-mechanics model. Halpin Tsai developed a reduced and simpler method that accommodate a variety of geometries by showing that materials can be modelled mathematically as laminated systems. Therefore the relations can be rearranged to give the following simpler relations for a generalised elastic property P^* as explained by Daniel and Ishai [7] and Halpin and Kardos [17]. This relations can also be used to determine the properties of composites with discontinuous fibres orientated in the loading direction Vinod and Aradhya [50]. This relations can be summarised as

$$\frac{P^*}{P_m} = \frac{1 + \zeta \eta V_f}{1 - \eta V_f} \tag{4.19}$$

where

$$\eta = \frac{P_f/P_m - 1}{P_f/P_m + \zeta} \tag{4.20}$$

P^* represents the composite properties for example $E_2, G_{12}, G_{23}, V_{12}$ and many more while the subscript m and f represent matrix and fibres respectively while V_f is the fibre volume ratio. ζ is a reinforcing efficiency and depends on fibre geometry, distribution and loading condition. The properties of a unidirectional lamina depend on the fibre volume ratio, which in turn depends on the packing order of the fibre. Figure 4.2a shows examples of some generalised packing geometries and their relationship with inclusion volume ratio in terms of spacing and inclusion radius.



(a) Rectangular Packing (b) Square Packing (c) Hexagonal Packing

Figure 4.2: Packing Order Geometries

$$V_f = (\tfrac{\Pi}{4})(\tfrac{r}{R_1R_2}) \qquad \text{(rectangular packing)}$$
$$V_f = (\tfrac{\Pi}{4})(\tfrac{r}{R})^2 \qquad \text{(square packing)}$$
$$V_f = (\tfrac{\Pi}{(2\sqrt{3})})(\tfrac{r}{R})^2 \qquad \text{(hexagonal packing)}$$

The direction parallel to the fibres is referred to as longitudinal direction see Figure 4.3. Fibres tend to enhance the effective properties of the composite when the loading is parallel to the fibres. A parallel model also known as rule of mixtures results in Equation (4.21) and (4.21) below from Daniel and Ishai [7] with the assumptions that there is no slipping in the matrix to fibre interface. Using the parallel model, the longitudinal Poisson's ratio and longitudinal modulus are

$$v_{12} = V_f v_{12f} + V_m v_m \tag{4.21}$$

$$E_1 = V_f E_{1f} + V_m E_m \tag{4.22}$$

where ν_{12f} is the longitudinal Poisson's ratio of the fibres and ν_m is the Poisson's ratio of the matrix. For isotropic material E_{1f} and ν_{12f} are taken as $E_f$and ν_f respectively. It should be noted that these mathematical models for longitudinal properties of unidirectional composite is quite accurate Sun and Vaidya [46]. Using the self-consistent method, which involves strain-displacement, constitutive relations, continuity conditions at the fibre-matrix interface, and regularity conditions at the centre of fibres, Daniel and Ishai [7] derived a more accurate expressions for the longitudinal modulus and Poisson's ratio as

$$E_1 = V_f E_{1f} + V_m E_m + \frac{4(\nu_m - \nu_{12f})^2 K_f K_m G_m V_m V_f}{K_f K_m + G_m(V_f K_f + V_m K_m)} \tag{4.23}$$

$$\nu_{12} = V_f \nu_{12f} + V_m v_m + \frac{(\nu_m - \nu_{12f})(K_m - K_f) G_m V_m V_f)}{K_f K_m + G_m(V_f K_f + V_m K_m)} \tag{4.24}$$

where k_f is the bulk modulus of the inclusions and k_m is the bulk modulus of matrix. For most of composites the last term of the two equations above is not important hence making the rule of mixtures a good approximation. In case of an isotropic material with Young's modulus E and Poisson's ratio ν, its plane bulk modulus can be expressed as shown by Equation (4.25).

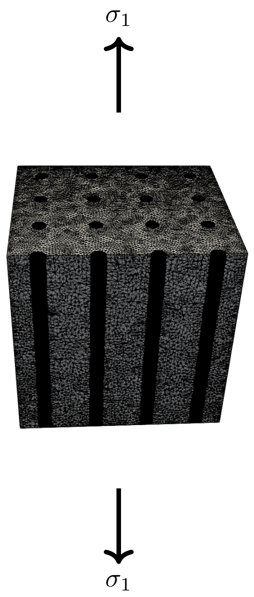


Figure 4.3: Longitudinal loading.

$$K_\varepsilon = K + \frac{G}{3} = \frac{E}{3(1-2v)} + \frac{E}{6(1+v)} = \frac{E}{2((1+v)(1-2v))} \tag{4.25}$$

Transverse elastic properties refers to the direction that is perpendicular to the fibre direction as illustrated by Figure 4.4 . The fibres considered here are assumed to be of uniform diameter and properties. The equations developed here assumes a model made from a layer of fibre and matrix material where each layer is perpendicular to the direction of loading with each layer carrying equal load Sun and Vaidya [46]. Using the series model and assuming the layers deform elastically the transverse modulus is

$$\frac{1}{E_2} = \frac{V_f}{E_{2f}} + \frac{V_m}{E_m} \tag{4.26}$$

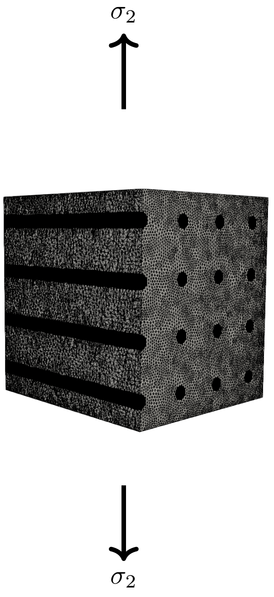


Figure 4.4: Transverse loading.

that is

$$E_2 = \frac{E_{2f}E_m}{V_f E_m + V_m E_{2f}} \tag{4.27}$$

where E_{2f} is the transverse modulus of fibres. Daniel and Ishai [7] indicated that E_m is usually replaced by

$$E_m^{'} = \frac{E_m}{1 - v_m^2} \tag{4.28}$$

where ν_m is the Poisson's ratio of the matrix. This is intended to account for limitations imposed on the matrix by the fibres in the fibres direction. However, this equation tends to underestimate the transverse modulus hence Halpin-Tsai relations are introduced

$$E_2 = \frac{E_m(1 + \zeta\eta V_f)}{1 - \eta V_f} \tag{4.29}$$

Where

$$\eta = \frac{E_{2f} - E_m}{E_{2f} + \zeta E_m} \tag{4.30}$$

ζ is the curve fitting parameter obtained experimentally on values of E_2. ζ is usually between 1 and 2. For hexagonal packing ζ is assumed to be 1 and 2 for square arrays. Sun and Vaidya [46] indicated that ζ of 2 may be used for fibres with circular and square cross sections. Therefore when $\zeta = 1$ the equation for E_2 above can be written as

$$E_2 = E_m \frac{(1 + V_f)E_{2f} + V_m E_m}{V_m E_{2f} + (1 + V_f)E_m} \tag{4.31}$$

In special case of isotropic fibre E_{2f} can replaced by E_f As results transverse modulus as function of fibre volume ratio can be obtained by Halpin Tsai method. Where $m = \frac{E_{2f}}{E_m}$, Figure 4.5 shows the variation of the in-plane Young's modulus with fibre volume ratio for 3 composites with $m = 1$, $m = 10$ and $m = 20$. This plot illustrates how the relative stiffness of the fibre and matrix, combined with fibre content, influences the overall stiffness of the composite and guides efficient, optimised design.

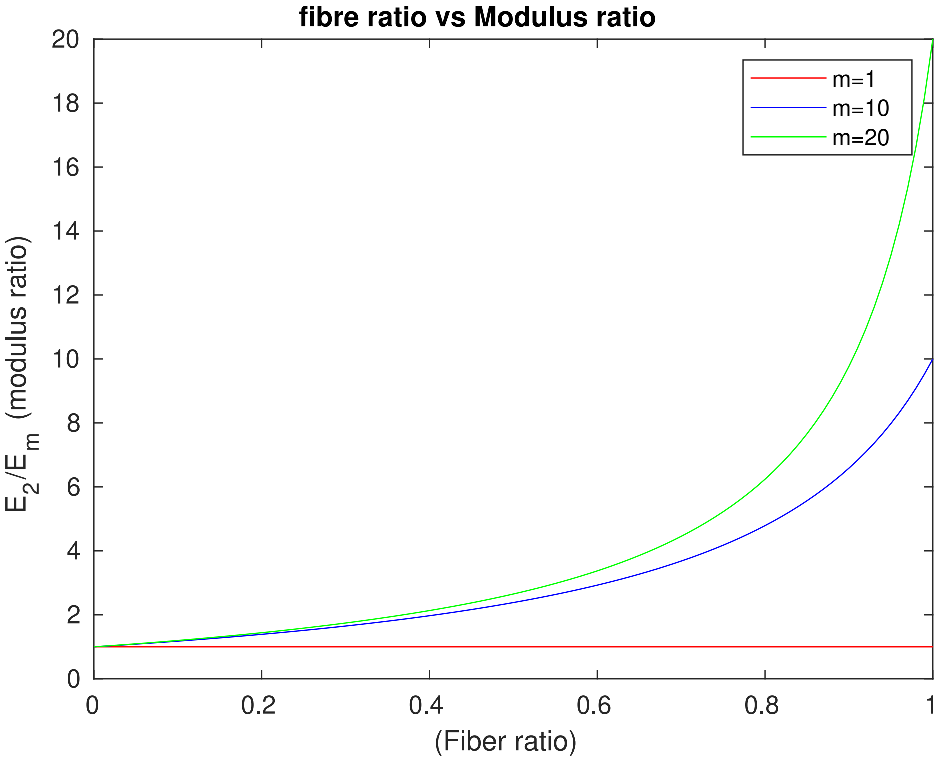


Figure 4.5: Modulus ratio vs fibre ratio

Using the self-consistent model, the transverse modulus can be related to the bulk and transverse shear modulus as explained by Daniel and Ishai [7] as

$$E_2 = \frac{1}{\frac{1}{4K_2} + \frac{1}{4G_23} + \frac{\nu_{12}^2}{E_1}} \tag{4.32}$$

where

$$K_2 = \frac{(K_{2f} + G_m)K_m + (K_{2f} - K_m)G_m V_f}{(K_{2f} + G_m) - (K_{2f} - K_m)V_f} \tag{4.33}$$

and

$$G_{23} = \frac{G_m(K_m(G_m + G_{23f}) + 2G_{23f}G_m + K_m(G_{23f} - G_m)V_f)}{K_m(G_m + G_{23f}) + 2G_{23f}G_m - (K_m + 2G_m)(G_{23f} - G_m)V_f} \tag{4.34}$$

Also the transverse Poisson's ratio ν_{23} can be related to K_2, E_1, E_2 and ν_{12} by

$$\nu_{23} = 1 - \frac{E_2}{2K_2} - 2\nu_{12}^2 \frac{E_2}{E_1}. \tag{4.35}$$

For materials with transverse isotropy in planes 2 and 3, it has already been proven that the transverse shear can be calculated as

$$G_{23} = \frac{E_2}{2(1 + \nu_{23})}. \tag{4.36}$$

In the context of shear modulus, one can use a similar series model based on the mechanics of materials, involving an interchange between the matrix and fibre. However, it is noteworthy that in the case of shear deformation γ_{12f} and γ_m, the dominance leans more towards the matrix γ_m. Therefore the average RVE shear deformation γ_{12} as indicated by Daniel and Ishai [7] is given as

$$\gamma_{12} = \gamma_{12f} V_f + \gamma_m V_m \tag{4.37}$$

or

$$\frac{\tau_{12}}{G_{12}} = \frac{\tau_{12}}{G_{12f}} V_f + \frac{\tau_{12}}{G_m} V_m \tag{4.38}$$

hence

$$\frac{1}{G_{12}} = \frac{V_f}{G_{12f}} + \frac{V_m}{G_m}. \tag{4.39}$$

Equation (4.39) can be written as

$$G_{12} = \frac{G_{12f} G_m}{V_f G_m + V_m G_{12f}}. \tag{4.40}$$

From Halpin Tsai relations

$$G_{12} = \frac{G_m(1 + \zeta \eta V_f)}{1 - \eta V_f} \tag{4.41}$$

where

$$\eta = \frac{G_{2f} - G_m}{G_{2f} + \zeta G_m} \tag{4.42}$$

Therefore when $\zeta = 1$

$$G_{12} = G_m \frac{(G_{12} + G_m) + V_f(G_{12f} - G_m)}{(G_{12f} + G_m) - V_f(G_{12f} - G_m)} = G_m \frac{(1 + V_f)G_{12f} + V_m G_m}{V_m G_{12f} + (1 + V_f)G_m} \tag{4.43}$$

and from the self-consistent model

$$G_{12} = G_m \frac{(1 + V_f)G_{12f} + V_m G_m}{V_m G_{12f} + (1 + V_f)G_m}. \tag{4.44}$$

Interchanging matrix and fibre designations gives the following bounds

$$G_m \frac{(1+V_f)G_{12f} + V_m G_m}{V_m G_{12f} + (1+V_f)G_m} \leq G_{12} \leq G_{12f} \frac{(1+V_m)G_m + V_f G_{12f}}{V_f G_m + (1+V_m)G_{12f}} \tag{4.45}$$

4.4 Example 1: Homogeneous Block

This is an example of an homogeneous RVE block with uniform properties throughout the RVE see Figure 4.6, that is the matrix only of bulk modulus of $100GPa$ and a shear modulus of $10GPa$. The purpose of this experiment is to compare the results from the least squared (`lsqr`) method and analytical method.

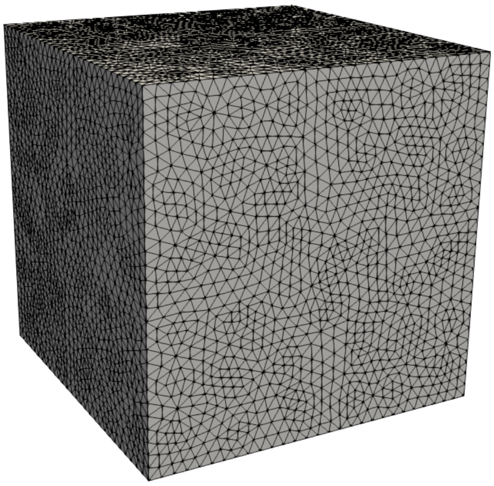


Figure 4.6: Homogeneous Block RVE mesh

Remarks
The Youngs modulus of exactly value of $29.03GPa$ together with Poisson's ratio of 0.4516 was recovered using the least square method. By using the elastic moduli for homogeneous isotropic material formula, $29.03GPa$ and 0.4516 were also recovered. For `lsqr` method the Young's modulus and shear modulus were achieved at a very low residual value of 5.7902×10^{-16}. Changing the size of the mesh or using larger/smaller elements had no effect on the values of E and ν. The use of either 4 noded elements or 10 noded elements also had no effect of the final values of E and ν. For each experiment carried out the same values of E and ν were recovered. The residual correspond to the machine tolerance and the exact parameters E and ν are recovered as expected.

4.5 Example 2: Spherical Inclusions

Figure 4.7 shows a slice of an isotropic block made of two materials being the matrix of bulk modulus K of $100GPa$ and shear modulus G of $10GPa$. The isotropic block is made up of matrix and spherical inclusions with K of $1000GPa$ and G of $100GPa$. The inclusion are arranged in a square packing such that elastic properties are similar for both longitudinal and transverse directions. A $4 \times 4 \times 4$ arrangemeant was used for this test. A total of six representative volume elements (RVE) were used where each RVE has inclusions of a specific radius to give a specific desired inclusion volume ratio. Each RVE is then subject to five different tests to give corresponding stresses. The results from the least square analysis are compared to the Halpin Tsai which takes the ζ to be between 1 and 2 as indicated by Daniel and Ishai [7] and the results are shown in Figure 4.8.

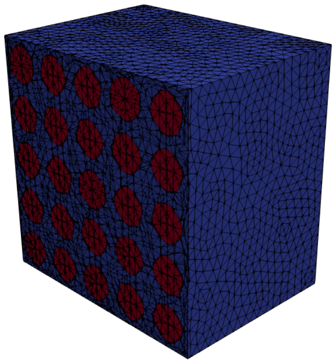
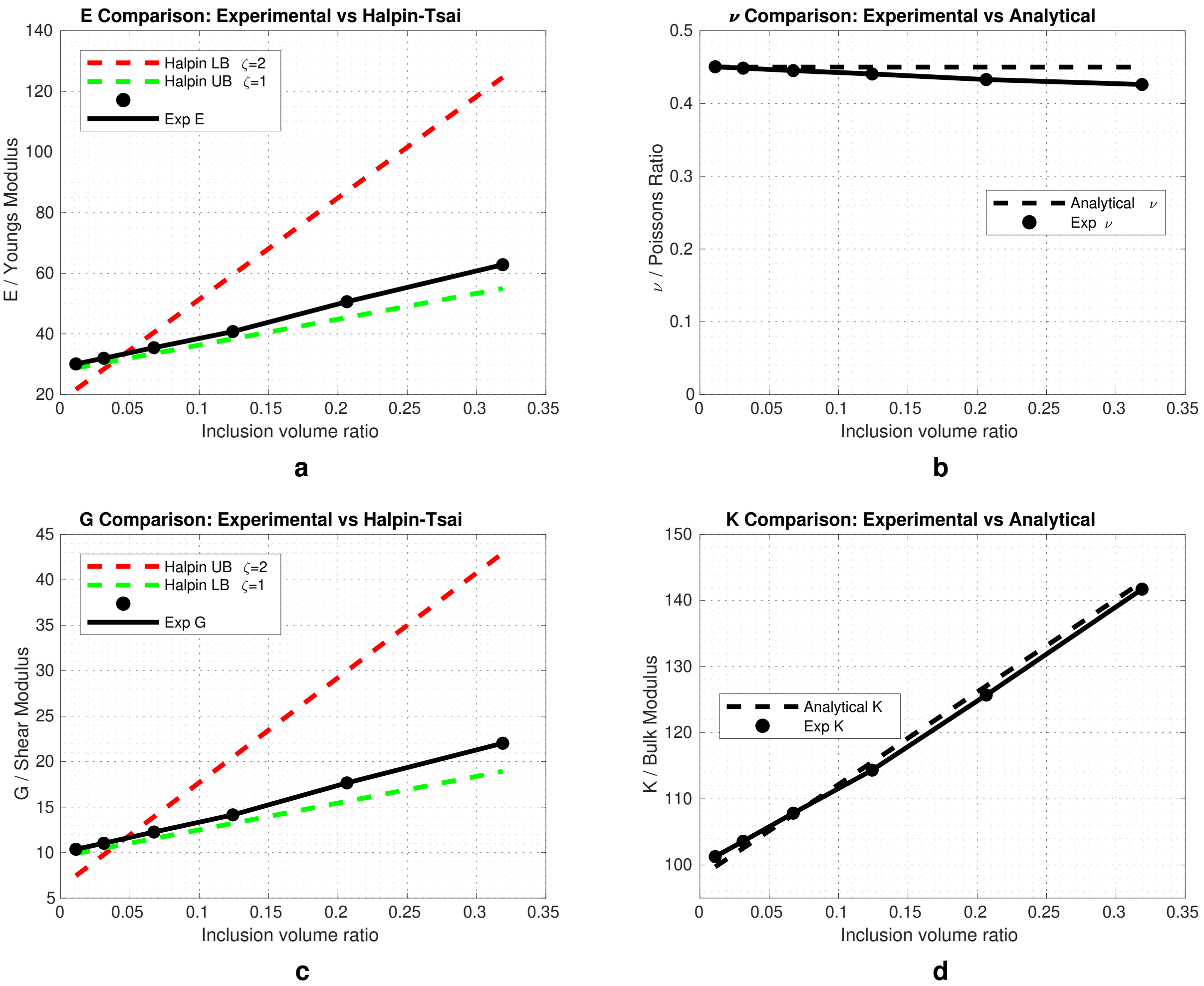


Figure 4.7: A Slice Through Microscopic -RVE with inclusion volume ratio 0.2

Figure 4.8 shows a comparison between the least squares method employed and the analytical methods.



a

b

c

d

Figure 4.8: Comparing the experimental results to Analytical methods for an isotropic material

Figure 4.9 shows how the residuals change as the radii of inclusions increases, all the residuals are smaller and within 5% accuracy hence supporting the reliability of the results.

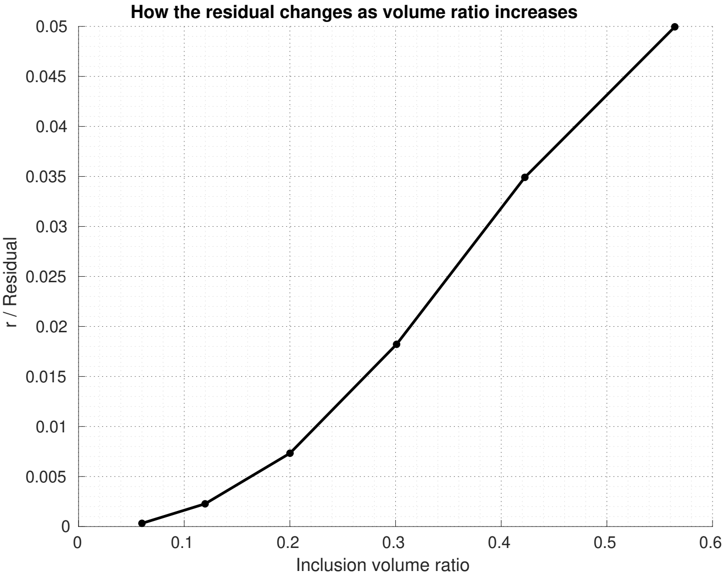


Figure 4.9: Residual as volume ratio increases

Remarks
The results for the Youngs modulus in Figure 4.8a is within the bounds of the halpin Tsai relations. At very low volume ratios the least squared method has produced the results which are within 0.5% accuracy as shown in Figure 4.9. These results support the example in Section 4.4 because at lower volume ratios the diameter of the inclusions are so small to an extent of being unnoticeable hence the material approaches state of being completely dominated by the matrix. In Figure 4.8 as volume ratio increases the Young's modulus in Figure 4.8a and shear modulus in Figure 4.8b of the composite increases. The poisson's ratio of the inclusions and the matrix are equal as results it is expected that ν of the composite to remain within 0.4516 as the volume ratio increases as shown in Figure 4.8b. Increasing the volume ratio results in an increase in the nominal residuals from the least squared method developed. The inclusions becomes noticeable within the matrix hence normal residual increases. An increase from volume ratio of 0.01 to 0.3 lead to 1.8% increase in normal residual. Figure 6.14 shows how the residuals changed as the volume ratio changes. The comparison between the developed homogenisation method and the corresponding analytical methods demonstrates good agreement as the values consistently lie within the theoretical upper and lower bounds. Furthermore the results are achieved at lower normal residuals, indicating both accuracy and stability of the solution. This results validate the reliability of the adopted approach and confirm its capability to capture the expected behaviour within analytical limits.

4.6 Example 3: Matrix and Fibers

For this example a transverse isotropic block of an RVE is considered. The RVE is made of long cylindrical fibres that run from one side to the opposite side and are embedded in the matrix. To control volume fraction ratio the radius of the fibres is varied to get six different volume fraction ratios hence six RVEs were generated. For each experiment in order to improve accuracy five experimental tests were carried out. The constituents has similar elastic properties as in Section 4.4. A square packing order was also used for this experiments and the results of the least square method are compared to analytical methods in Section 4.3.Figure 6.14 below. show the arrangement of continuous fibres in the RVE used for this example.

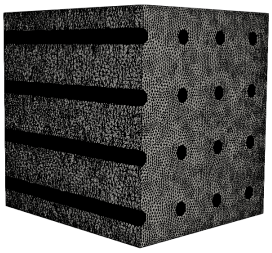


Figure 4.10: Slice through an RVE whose volume ratio is varied between 0.1 and 0.75

Remarks
As explained by Daniel and Ishai [7] mechanics of materials predictions are adequate for longitudinal properties that E_1 and ν_{12} for unidirectional continuous fibre composite as longitudinal properties are not sensitive to shape or packing of the fibres. However this method tends to underestimate transverse properties G_{23} and E_2 as they are sensitive to shape and distribution of fibres. Figure 4.11b and 4.11e shows how well this analytical method coincides with the least square method for the longitudinal properties, however for the graphs for transverse properties as shown by Figure 4.11a, 4.11d, 4.11f and 4.11e these least squared method results are higher than those obtained using Section 4.3. Therefore the shape and distribution of fibres play an important role in transverse direction.
Unidirectional fibres are used in marines, aerospace and automotive. Composite microstructure is dependent on volume fraction, fibre and matrix types such that any change in this properties results in a completely new composite with different properties Huang et al. [20].Carrying out experiment in the laboratories is time consuming and expensive as the properties depends on so many parameters such as shape and size of both fibre and matrix. Sun and Vaidya [46] indicated that the overall behaviour of composite depend also on both physical and chemical interactions therefore one set of experimental measurement results in properties of a fixed fibre matrix system. Sun and Vaidya [46] also indicated that semiempirical methods may not be reliable for component design as the presents difficulties in predicting transverse properties of unidirectional composite. Hence this computer modelling of composites may be the solution.

Huang et al. [20] concluded that when the fibre shape becomes irregular the geometry of the fibres becomes even more important in determining transverse elastic properties. The study went on to propose three geometry parameters namely the square root of ratio of fibre perimeter to fibre cross section, fibre width in the loading direction and fibre width in the non loading direction. It was found out even though higher values of the other two parameters increases the value of transverse properties, higher values of fibre width in non loading direction had a negative influence. Huang et al. [20] concluded that shape should be considered to provide correct transverse modulus.
The residuals are observed to increase as the volume ratio increases. Although there is an increase in the normal residual the accuracy remains within 10% hence reasonable enough to be used as a homogenisation tool for effective elastic properties. Although in Section 4.6 the transverse isotropic material considered has only one orientation there are other factors which my influence both the longitudinal and transverse strength and stiffness. The distribution of the load between the matrix and the fibres is affected by orientation of the fibres. Since inclusions in this study are assumed to have superior properties compared to the matrix, they tend to be effective when they are parallel to the loading loading direction. Fibres strength and stiffness also play a role in the overall composite effective properties. Interfacial conditions also influences the mechanical properties and performance of composites as they regulate the load transfer between the fibres and matrix. There are many other factors which influences the behaviour and effectiveness of composites in reality including but not limited to discontinuous fibres and residual stresses in the constituents and at the interface Sun and Vaidya [46].

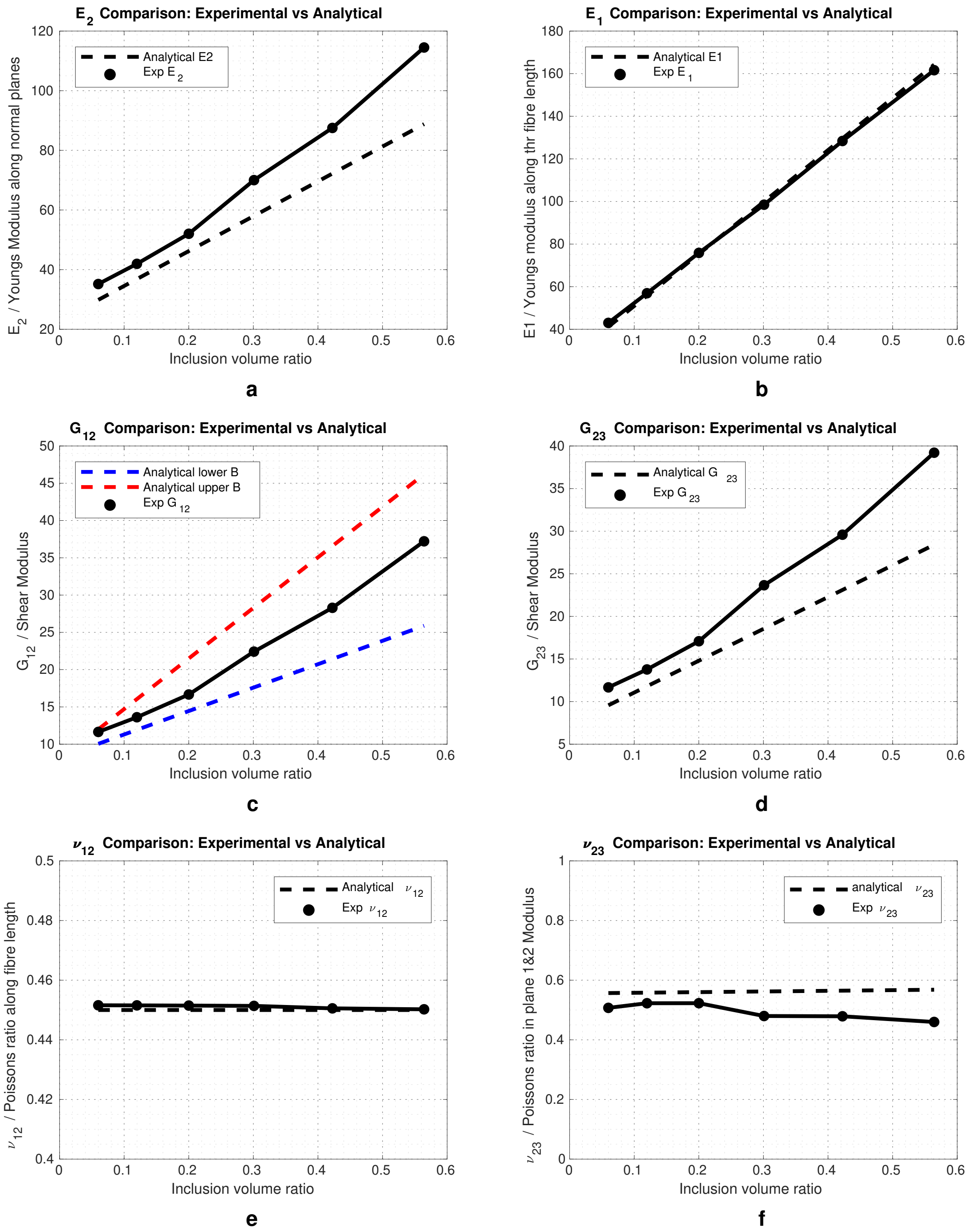


Figure 4.11: Experimental results vs Analytical methods for a transversely isotropic material

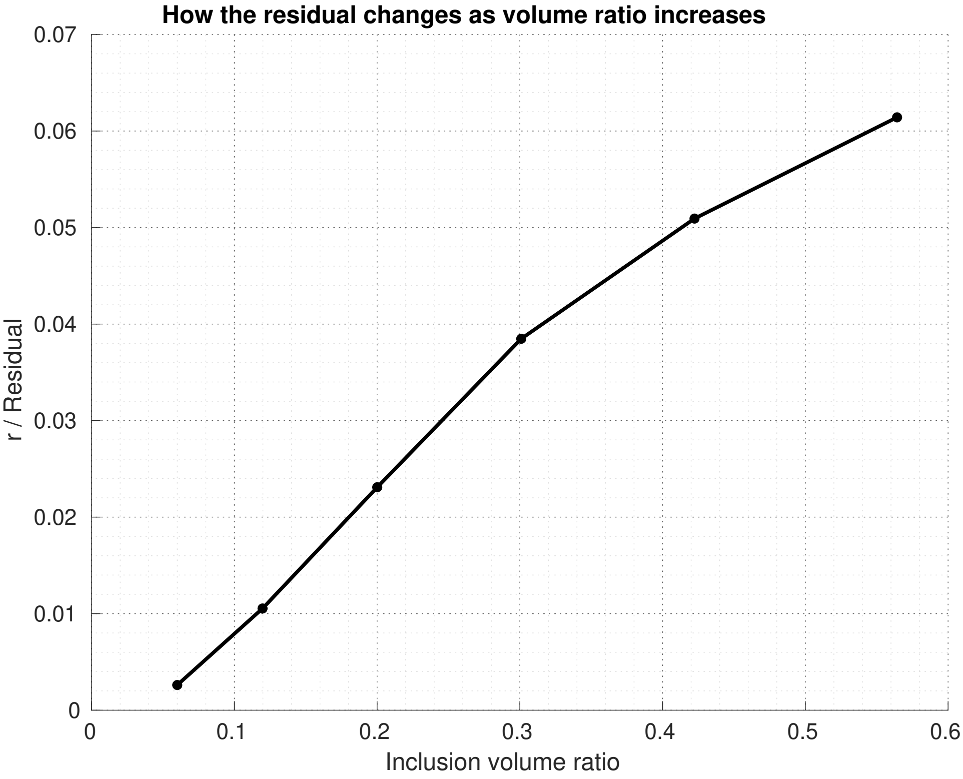


Figure 4.12: Residual as volume ratio increases

Chapter 5

Machine Learning Enhanced Homogenisation

5.1 Objective and Outline of Methodology

The study aims to utilise neural networks to determine the elastic properties of composites using the homogenisation results obtained from Chapter 4 so that effective properties from any combination of different parameters of the RVE can be determined without the need to create a new representative volume element and repeating the procedures in Chapter 4. Neural network can learn complex relationships and nonlinear behaviours directly from the data. Their strength in pattern recognition and prediction enables application across diverse fields of engineering which makes them a perfect choice for enhanced homogenisation compared to traditional modelling approaches. Subsequently, an optimisation tool which utilises this trained neural network is used to design a better material as elaborated in the subsequent chapter. The objective of this enhanced homogenisation is to provide a faster and simpler solution as it reduces computational cost and time consumptions compared to least square homogenisation technique in Chapter 4 which provide elastic properties for particular RVE with specific geometric parameters. The elastic properties here are estimated by this trained network rather than going through the FE method and least squared homogenisation method over again so that more RVEs can be analysed for optimisation in later stage. With this trained network, optimal design parameters can be selected to enhance the elastic properties of the composite in relation to cost, weight, or any parameter of interest.

An automated geometry file (`.geo` file) is generated using a `C++` code file with loops to control the RVE geometry. Any change in RVE geometry/parameter creates a different RVE. The RVE generated has specific elastic properties for that particular geometry and fibre distribution/packing. The geometry file is then systematically loaded into another software called `Gmsh`, which converts the geometry file to a mesh file (`.msh` file). Mesh file is loaded into Dettmer's Multi Physics Analysis Program software (`Mpap3`), which uses the finite element method where small strains $\boldsymbol{\epsilon}$ are applied to the RVE to generate the corresponding stresses $\boldsymbol{\sigma}$. Each RVE is subjected to five experiments and the corresponding output stresses from `Mpap3` are written to a file. Using MATLAB codes in Appendix [B.3, B.4, B.5], stresses and strains values are converted to effective elastic properties for each RVE such as bulk modulus K and shear modulus G.

Section 5.4 in this chapter considers two radii combinations (spherical void and inclusions) to generate different RVEs for training. These radii are converted to volume ratios then used as inputs together with their corresponding K and G as output to train a network so that homogenised properties within that particular radii combination ranges can be estimated. Section 5.5 considers discontinuous cylindrical fibres arranged in a brick like pattern. Non dimentionalized spacing ($sp_{x,y,z} = \frac{sp}{d}$) between bricks in the x, y, z directions and the aspect ratio ($\alpha = \frac{l}{d}$) are varied to generate several RVEs. These four parameters are then used as inputs while their corresponding elastic properties are used as outputs to train a neural network.

With the results validated in Chapter 4 through comparison with other methods, the subsequent steps will involve neural network training. This training aims to extrapolate the elastic properties for the

RVE geometry that were not included in the experimentation. However, it is imperative to acknowledge that the attainment of optimal results is constrained within the bounds of the training data's geometry parameters. For instance, if the minimum and maximum radii of voids and inclusions are set at 1 and 5 units respectively in the training dataset, then the trained network will perform optimally within the range of $1 \leq r \leq 5$. Consequently, the trained model can estimate properties for void radii, such as 1.2 units, and inclusion radii, such as 4.05 units, which were not experimentally tested. This trained neural network can be saved and be used over and over again without the need to train the network again. Neural network tends to produce accurate results and has emerged as powerful tool in engineering. The procedure followed in this chapter can be summarised by the Figure 5.1.

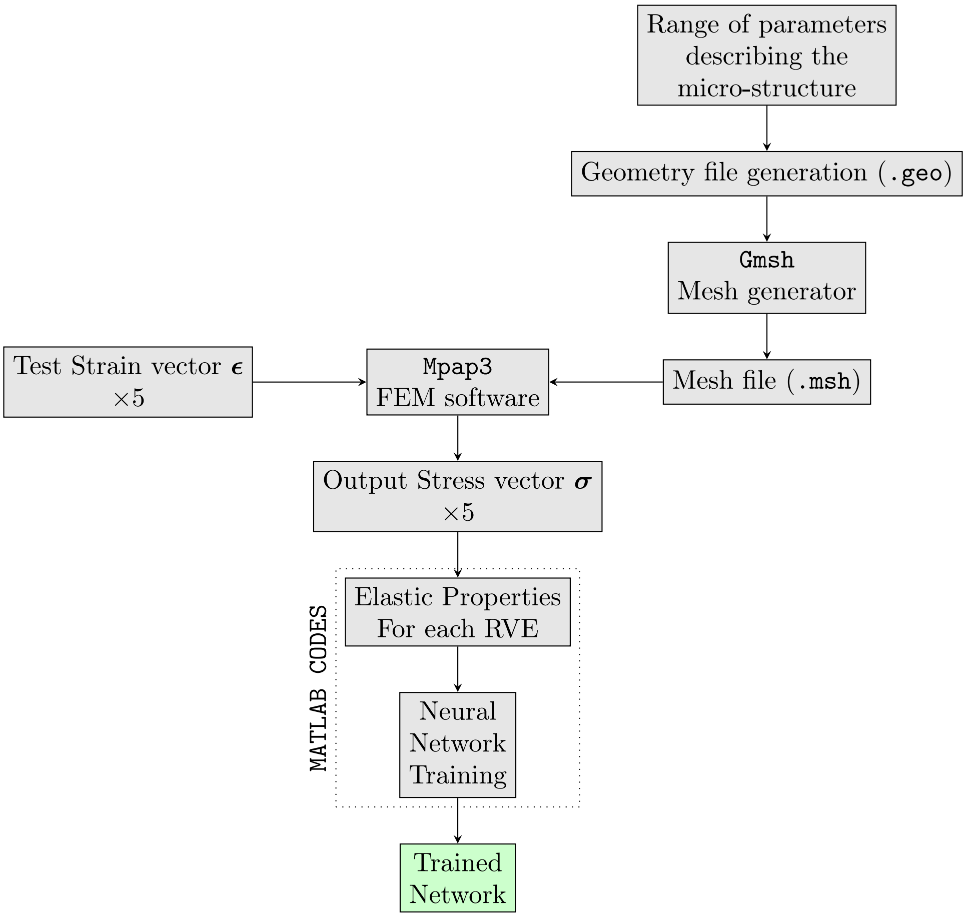

Range of parameters describing the micro-structure
Geometry file generation (.geo)
Gmsh
Mesh generator
Mesh file (.msh)
Test Strain vector ϵ ×5
Mpap3
FEM software
Output Stress vector σ ×5
MATLAB CODES
Elastic Properties For each RVE
Neural Network Training
Trained Network

Figure 5.1: Procedural Methodology Utilized

5.2 Feedforward Neural Networks

A neural network is a computational tool inspired by the information processing mechanisms of the human brain, which mimic how neurones in the brain signal each other. It consists of an input layer, one or more hidden layers, and an output layer, each composed of several neurons or nodes. Neurons in one layer are connected to all neurons in the next layer, facilitating the forward transmission of information. This structure is referred to as a feedforward neural network. Neural networks have emerged as powerful tools with multiple applications, ranging from pattern recognition to classification and optimisation. Figure 5.2 below shows feed forward neural network with inputs V_v and V_i representing the volume ratios of voids and inclusions respectively and output elastic properties G and K for example in Section 5.5 while x_i is to indicate that there could be more inputs in general.

From Figure 5.2 the following expressions, w, x, α, K and G, and $\phi^{()}$ represent the weight functions, input raw data, bias, output, and $\phi^{()}$ is the activation function. Note that the notation $\phi^{()}$ was used, as

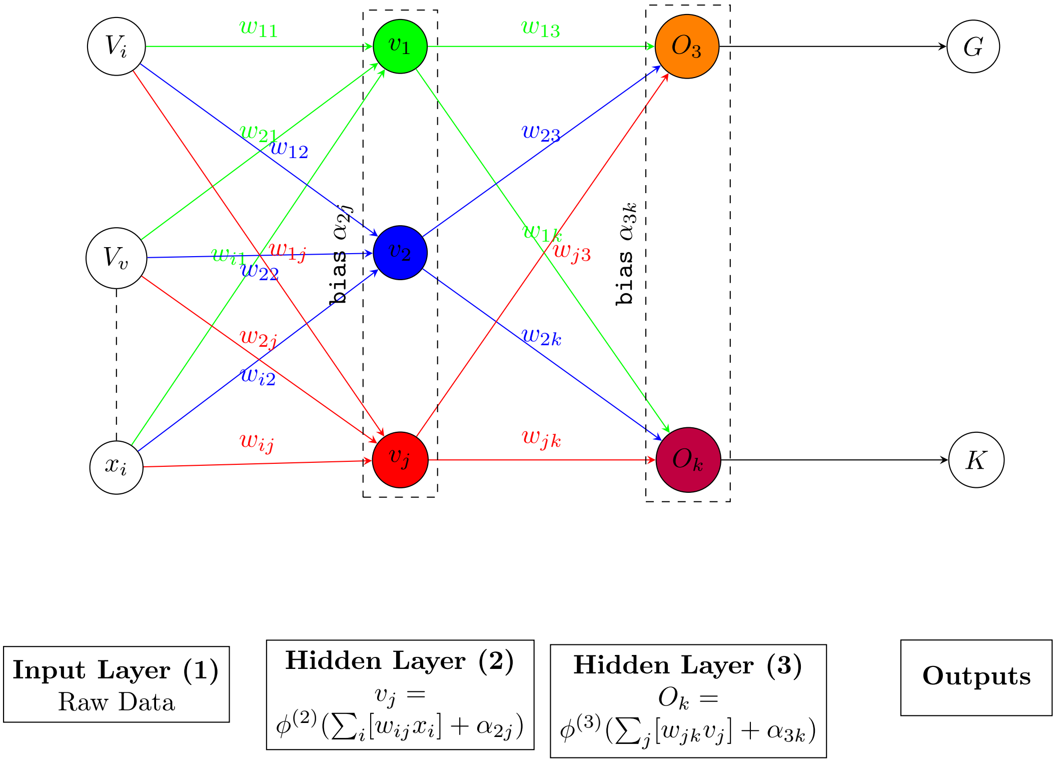


Figure 5.2: Neural Network procedure

different layers may use different activation functions. The output of a neutron in a hidden Layer (2) from Figure 5.2 , can be written as

$$v_j = \phi^{(2)}(\sum_i [w_{ij} x_i] + \alpha_{2j}) \tag{5.1}$$

According to Simon. [44], the bias α_{2j} provides the affine mapping to the linear combiner $\sum_i [w_{ij} x_i]$. α_{2j} can be positive or negative and it is an external parameter applied to neurone j in Layer 2 of Figure 5.2. w_{ij} is described as the synaptic weight of neuron j. It is noteworthy that neurons are activated only when the input signal exceeds a predetermined threshold, which regulates the network's behaviour and information processing capabilities.

Activation function ϕ determines the output of a neuron in relation to v_j, where $v_j = \sum_i [w_{ij} x_i] + \alpha_{2j}$. This function regulates whether a neuron should be activated or not. It decides whether the neuron input is sufficient enough for decision making. Examples of this functions which are explained full by Simon. [44] include the threshold function where

$$\phi(v) = \begin{cases} 1 & \text{when } v \geq 0 \\ 0 & \text{when } v < 1 \end{cases} \tag{5.2}$$

Equation 5.2 is an example of a threshold function, it means that the output of the neuron will either be a one or zero. Sigmoid functions which are an increasing function with the ability to map input values to a continuous range between one and zero. For example

$$\phi(v) = \frac{1}{1 - e^{av}}. \tag{5.3}$$

The are several activation function including the transig, logsig purelin and many more . In supervised learning, the training process entails utilising raw data samples that are pre-classified, a method commonly known as error back-propagation algorithm. This approach involves training the network using input-output samples, where errors are meticulously computed and subsequently leveraged to fine-tune synaptic weights. According to Sathya and Abraham [42], the error back-propagation process unfolds in two distinct passes. Forward pass, the input signal is passed forward through the neuron of the network, and the error is calculated between the desired output and the actual output as

$$e_i = d_i - v_j \tag{5.4}$$

Backward pass process propagates the error back through the network from the output layer, and free parameters are adjusted either example by example basis or using epoch-by-epoch where each epoch contain an entire set of training examples. Network parameters are adjusted step by step under a combined influence of the training vector and error signal. w_{ij} is adjusted using the delta rule, as

$$\Delta w_{ij} = \eta \delta_j x_i \tag{5.5}$$

where

Change in synaptic weight between neurones i and j:	Δw_{ij}
Learning rate:	η
Error signal associated with neuron j :	δ_j
Input from neurone i to j:	x_i

During this process environmental information is transferred to the network and stored in form of w_{ij} and remains as a long term memory. The stored information can then be used on itself independent of the sample. Supervised learning is a powerful tool employed to tackle both linear and non-linear problems, including predictive tasks. The aim of this chapter is to train this network given the outputs $T = [K, G]$ and input $P = [V_v, V_i]$, where V_v, V_i, G and K represent void volume ratio, inclusion inclusion, shear modulus and bulk modulus respectively. The function approximation model adopted here is the same as Simon. [44] which assumes input-output mapping described by the functional relationship

$$T = \phi(P) \tag{5.6}$$

where P is a vector of inputs/ design parameters and T is a vector of outputs elastic properties. Since $\phi(.)$ is unknown we have a set of labelled examples

$$\xi = \{(P_i, T_i)\}_{i=1}^{N}. \tag{5.7}$$

The aim is to train a network such that the approximation $\Phi(.)$ mapping of input-output is close enough to $\phi(.)$ in Euclidean sense over all inputs as given by

$$\|\Phi(P) - \phi(P)\| < e \quad \forall T. \tag{5.8}$$

Since we have so many unknowns, a larger sample of size N, that is more RVEs will be required to solve for all of the unknown parameters, then the approximation error e can be made small enough for the task. The mapping is approximated in two ways, one being system identification where the difference between T_i associated with P_i and the output of the network $\Phi(P_i)$ provide an error signal e_i, this signal is then used to adjust free parameters to minimise the squared difference and is calculated over the entire sample ξ. Second way being the inverse modelling where the model produces a vector P in response to vector T. This can be represented as

$$P = \phi^{-1}(T) \tag{5.9}$$

Where $\phi^{-1}(.)$ is the inverse of $\phi(.)$ and T is the input and P is the desired output in this case. The program is already built-in to MATLAB, and Algorithm 2 code utilises this program.

From Algorithm 2 step 3 a and b are the size of the hidden layers in a neural network see Appendix B.6. a represents number of neurones in first hidden layer while b represent number of neurones in second hidden layer. For data categorisation, MATLAB organises input data at random to be used for training, validation and testing. Levenberg-Marquardt (LM) is used by the MATLAB built in program to update synaptic weights and bias. According to Lv et al. [27] LM algorithm is derived from the Newton's method for minimising functions that are sums of the squares of the nonlinear fuctions. Which states that for $F(x)$

$$\begin{aligned} x_{k+1} &= x_k - A_k^{-1} g_k \\ A_k &= \Delta^2 F(x)|_{x=x_k} \\ g_k &= \Delta F(x)|_{x=x_k} \end{aligned} \tag{5.10}$$

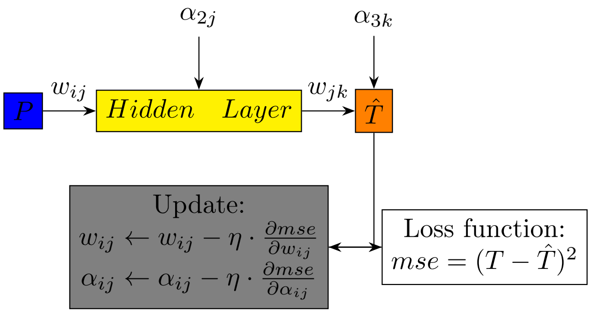
Algorithm 2 Neural Network Training with Two Inputs and Two Outputs

```
Require: $V_v, V_i$   Inputs
Require: $K, G$   Outputs
Initialize $T = \begin{bmatrix} K & G \end{bmatrix}$
Initialize $P = \begin{bmatrix} r_v & r_i \end{bmatrix}$
Create a feedforward neural network $'mlnet'$ with hidden layer sizes $a$ and $b$
Configure the neural network $'mlnet'$ with inputs $P$ and targets $T$
Train the neural network $'mlnet'$ with inputs $P$ and targets $T$
Save the trained neural network model to $'my\_neural\_network\_model.mat'$
```

where $\Delta^2 F(x)$ is the Hessian matrix and $\Delta F(x)$ is the gradient. Therefore the weight functions are updated as

$$w_{k+1} = w_k - \eta \Delta e(w_k) \tag{5.11}$$

where Δe is the error function and η is the learning rate. Figure 5.3 illustrates how these parameters are updated.



α_{2j}
α_{3k}
P
w_{ij}
$Hidden \quad Layer$
w_{jk}
$\hat{T}$
Update:
$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial mse}{\partial w_{ij}}$
$\alpha_{ij} \leftarrow \alpha_{ij} - \eta \cdot \frac{\partial mse}{\partial \alpha_{ij}}$
Loss function:
$mse = (T - \hat{T})^2$

Figure 5.3: Gradient descent updating weight functions and bias in a neural network

Synaptic weights and biases are adjusted to optimise performance. According to **?**] where we have a set of inputs and outputs

$$\{P_1, T_1\}, \{P_2, T_2\}.......\{P_N, T_N\}.$$

The mean square error is calculated as

$$mse = \frac{1}{N}\sum_{k=1}^{N} e_k^2 = \frac{1}{N}\sum_{k=1}^{N}(T_k - \hat{T}_k)^2. \tag{5.12}$$

Figure 5.4 shows an example of a default Matlab feedforward network where $a = 5$ and $b = 1$

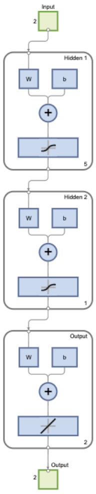
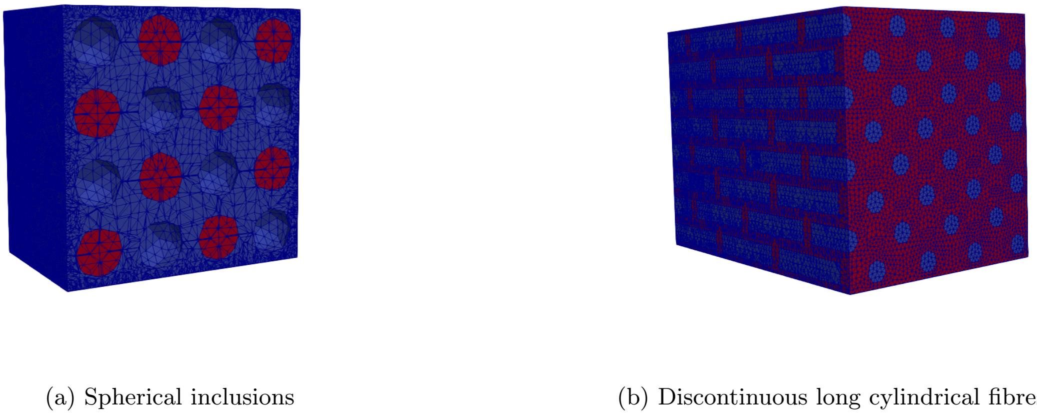


Figure 5.4: Matlab default neural network diagram

5.3 Parameterisation of Micro-Structure

For this investigation, a Representative Volume Element (RVE) consisting of spherical inclusions and voids, and cylindrical discontinuous fibres respectively are used together with their varying geometric parameters such as spacing and radii to generate N samples of RVEs which are analysed and used to generate the training data (elastic properties). Fibres are arranged in patterns as shown by the Figure 6.11a.



(a) Spherical inclusions

(b) Discontinuous long cylindrical fibres

Figure 5.5: RVEs For Training

During this experimental set up inclusions and voids are arranged in a hexagonal packing, while discontinuous cylindrical fibres are arranged in square packing manner as explained in Figure 4.2b and 4.2c. The RVE represents the microscopic structure of the composite and this chapter aims computes elastic properties of any possible combination of volume ratios for this particular inclusion arrangements.

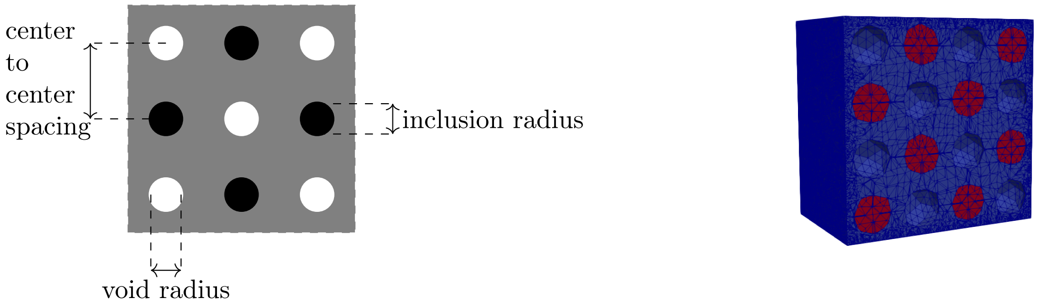
The example in Section 5.4 exhibits isotropic properties, whereas RVEs in Section 5.5 with discontinuous fibres convey transversely isotropic characteristics. It is assumed that there is no slip between the matrix and fibre. The RVEs containing discontinuous fibres is assumed to be devoid of any voids. Linear boundary conditions were employed during the training process for both RVE configurations. The bulk modulus of $100GPa$ for matrix K_m and $1000GPa$ for fibres K_f; the shear modulus of $10GPa$ for matrix G_m and $100GPa$ for fibres G_f are used for this experimental set up.

From Figure 5.6a similar parameters are considered in this chapter. The center-to-center spacing between inclusions/fibers is kept constant for Section 5.4 with a range of $0.1 \leq r \leq 0.5$ used for the radii for both the inclusion and the voids. For Section 5.5 the length of the discontinuous inclusions is varied to generate aspect ratio $6 \leq \alpha \leq 18$, inclusion spacings in x, y, z directions are varied to generate test RVEs. At no point are the inclusions allowed to overlap or touch.

5.4 Example 1: Isotropic Elastic Material

Analysis of a Representative Volume Elements (RVEs) featuring voids and inclusions arranged in a regular pattern. The radii for both voids and inclusions are varied independently to produce several RVEs. Throughout these experiments each group of inclusions are of same radius for each RVE. Voids' radius can be of same value as the inclusion radius however they are varied independently. A radius range $0.1 \leq r \leq 0.5$ was used for neural network training. The radii of voids and inclusion are varied independently within this same range. The experiment in Chapter 4 can only be done for some preselected radii combinations. In order to be able to estimate K and G for any possible combinations within this range a neural network is trained to handle the computations. The other alternative will be to perform the calculations for each possible combinations using Chapter 4 which is time consuming.

The parameters that controls the geometry of the RVE are centre to centre spacing which remains constant throughout the experiments. The radius of voids and radius of inclusions are varied to produce 36 different RVEs. Each RVE is subjected to 5 different sets of test strains $\boldsymbol{\epsilon}$ and through Mpap3 the corresponding sets of stresses $\boldsymbol{\sigma}$ are obtained. Homogenisation is then applied to each RVE to yield 36 pairs of elastic constants (K, G) with their respective radii combinations. The Figure 5.6 summarises the parameters of interest in the RVE.



(a) Parameters of interest from the RVE.

(b) A slice of an isotropic RVE for experiments

Figure 5.6: RVE parameters

The two radii are varied independently to create different RVEs for testing for cost reduction purposes. The overall composite is expected to exhibit similar properties in all directions, both longitudinally and transversely, due to the arrangement of voids and inclusions in the microstructure.

The following contour maps from Figure 5.7 and 5.8 shows how the Youngs modulus and Poisson's ratio varies with changes in both voids and inclusion ratio. As indicated by Figure 5.7 an increase in inclusion volume ratio increases the Young's modulus E while the voids reduces E. These results below are obtained using the techniques in Chapter 4 and they will be used to train a neural network.

Figure 5.8 shows how the Poisson's ratio changes with the void and inclusion ratio.

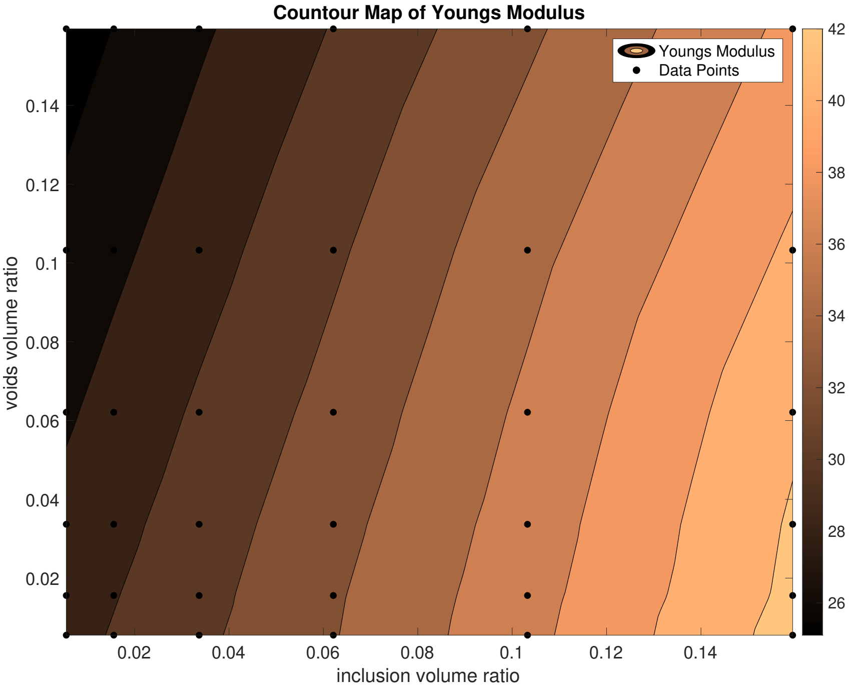


Figure 5.7: Variation of E with voids and inclusions volume ratios

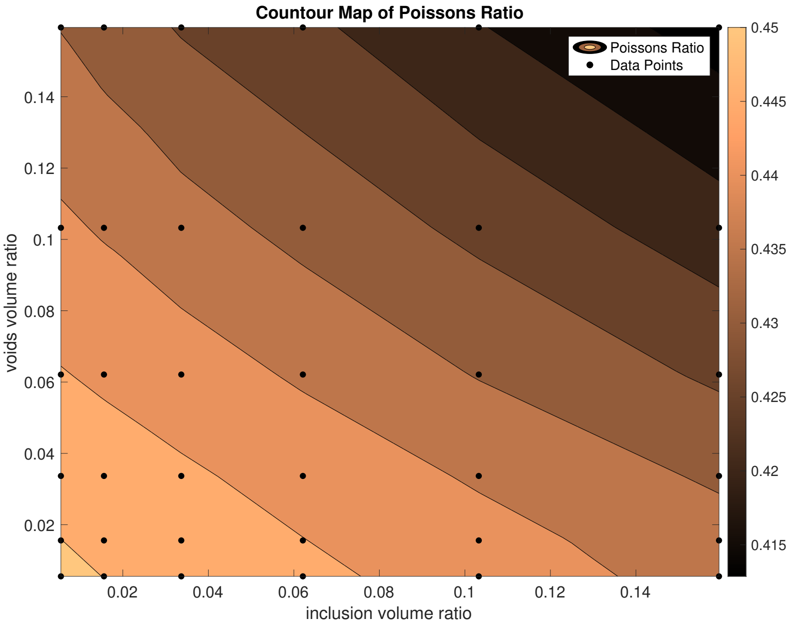


Figure 5.8: Variation of ν with voids and inclusions ratio

Trial tests will be performed in order to choose a suitable network.The number of hidden layers and neurons in a layer will be varied in order to get a better network. Figure 5.9 shows a representation of the neural network used for the first trial training using the results obtained from Figures 5.7 and 5.8. For neural network training the results from Figure 5.7 and 5.8 are converted into G and K for every point in the graphs. A trained network is used to estimate G and K for any point chosen at random within the graphs above. The accuracy of the trained network depends on several things such as the number of neurones and number of hidden layers as indicated by Figure 5.10 and 5.11.

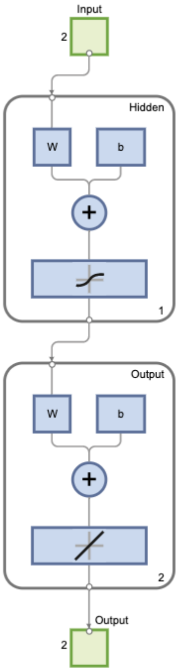


Figure 5.9: One neuron and one hidden layer network diagram

The networks under investigation are a one neuron network consisting of a single hidden layer and another one hidden layer network with ten neurons. The input to the network is comprised of two row vectors, Input 1 $[Vv_1, Vv_2, \ldots, Vv_{36}]$ and Input 2 $[Vi_1, Vi_2, \ldots, Vi_{36}]$, where Input 1 is the void volume ratio and Input 2 is the inclusion volume ratio each containing 36 elements. These input vectors are fed into the network's single neuron network and processed by the hidden layers to generate outputs. The network then produces two corresponding output row vectors shear modulus and bulk modulus, Output 1 $[G_1, G_2, \ldots, G_{36}]$ and Output 2 $[K_1, K_2, \ldots, K_{36}]$, each containing 36 predicted values. This structure allows for efficient mapping of the input vectors to the desired outputs through a single-neuron neural network model.

Figure 5.10 shows the approximations made by a network in Figure 5.9 which has one neuron, one hidden layer while Figure 5.11 is the approximation of a similar network but with ten hidden layers.

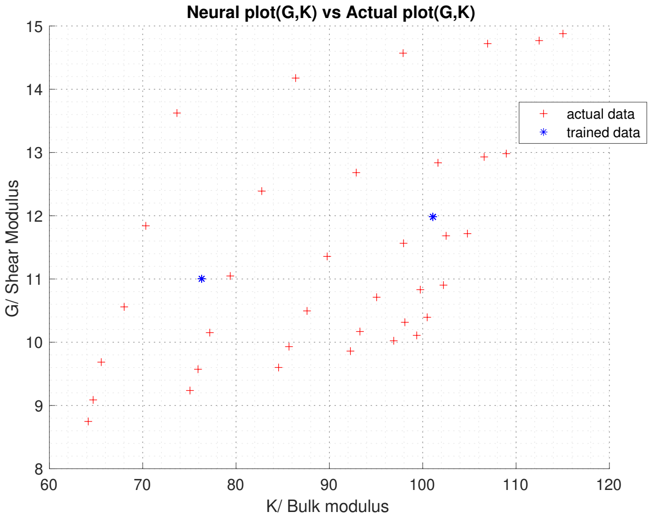


Figure 5.10: One neuron and one hidden layer approximation

A one neuron network leads to linear weight functions and it produces poor results, increase in number

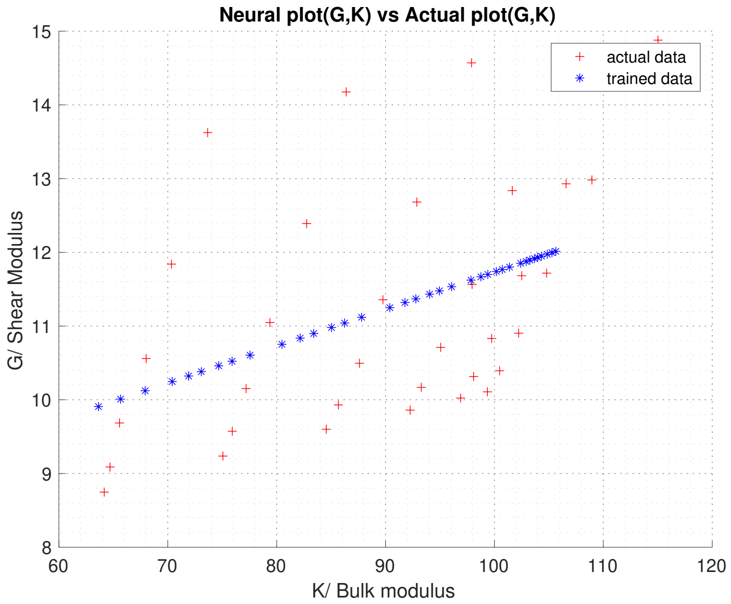


Figure 5.11: One neuron and ten hidden layer approximation

of hidden layers or the use of a sigmoid activation function does not improve the results. Therefore one neuron cannot be used to train this particular network. This takes us to another trial test which explores a neural network with 10 neurons and 1 hidden layer see Figure 5.12 and one network with 10 neurons and 10 hidden layers.

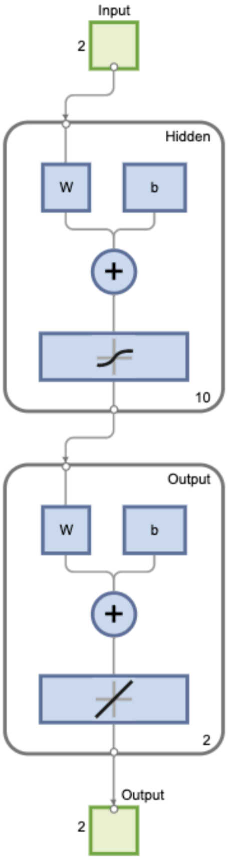


Figure 5.12: Diagram of a ten neurons and one hidden layer network

Figure 5.13 shows the approximations made by the neural network with ten neurons and one hidden layer in Figure 5.12 while Figure 5.14 shows the same results' approximations using a similar network but with ten hidden layers.

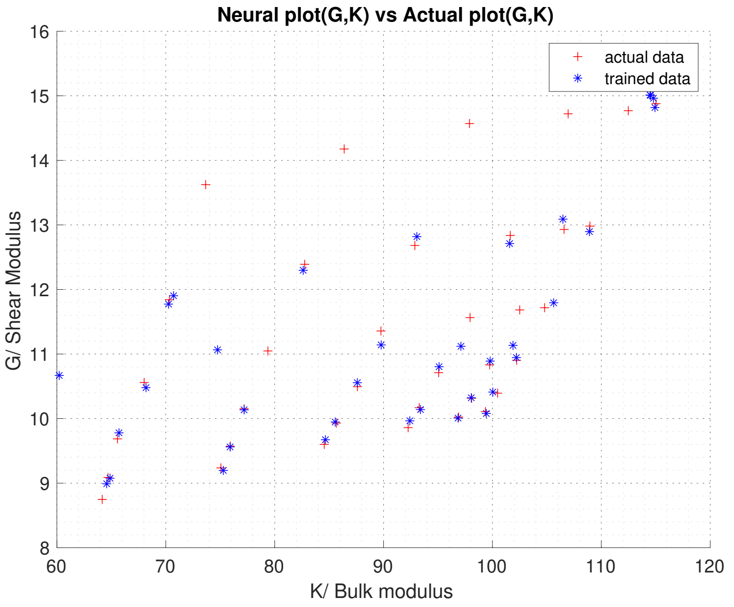


Figure 5.13: Ten neurons and one hidden layer approximation

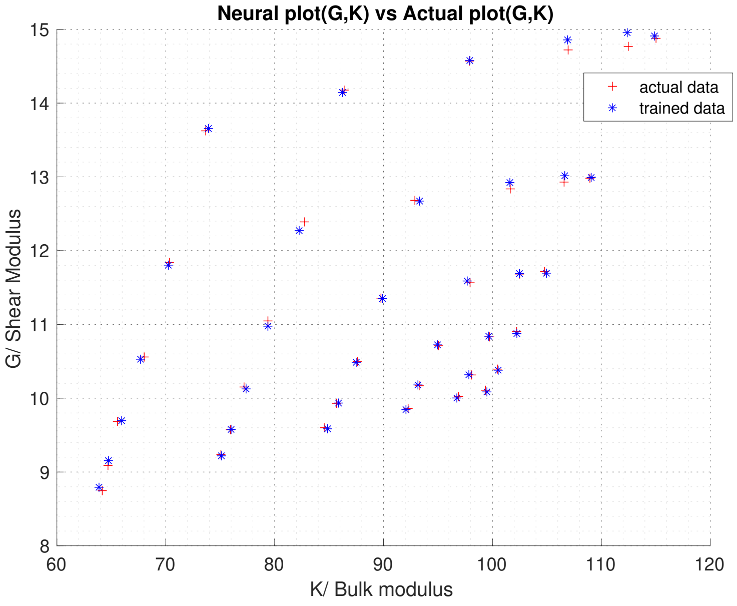


Figure 5.14: Ten neurons and ten hidden layer approximation

When more than one neurons are used for network training, results accuracy improved as compared to the test with just one neurone. Ten neurone with one hidden layer network without an activation function still leads to linear weight functions however it can produce good results after training the neural network several times until most point coincides. Introduction of a sigmoid function will results in non linear weight functions hence better approximations. A ten neuron network with ten hidden layers results in a non linear network hence capable of modelling complex patterns. Therefore nonlinear weight functions (using more than one neurones) can yield better and accurate approximations. Figures 5.15 and 5.16 further evaluates the performance of a ten neurons network with ten hidden layers which resulted in better approximation in Figure 5.14.

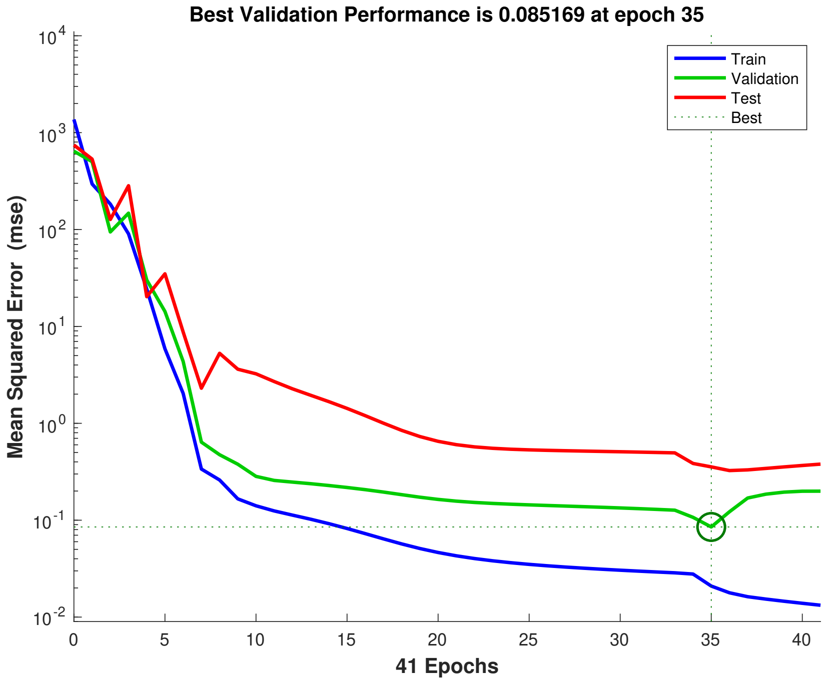


Figure 5.15: The mean squared error with respect to epoch for the network

Figure 5.15 shows MSE variation concerning the epoch for training , validation and testing. One epoch refers to using the entire data set to perform both backward and forward propagation only once. Figure 5.15 indicates that the performance on both the training and validation sets is satisfactory, that is the final mean square error is small and shows optimal validation performance found at epoch 41 with best validation MSE 0.085 and best training performance of 0.355 at 35 epoch. Validation error is greatly reduced at 35 epoch and the weight and biases were used for the trained network. Test and validation set has similar characteristics and there is no overfitting before epoch 35 where the best performance occurs.

Figure 5.16 gives the correlation between targets that is G and K from least squared method in Chapter 4 and the output generated by the trained network. Data categorization for this process is done at random by MATLAB's built-in algorithms. The regression lines are used to monitor the fitness of the approximations. A perfect fit occurs when the R-value is one, indicating that the network has mapped the data accurately, including any background noise, fluctuations and and outliers the data may contain. This ensures that the model's predictions are both reliable and robust in handling various data complexities.

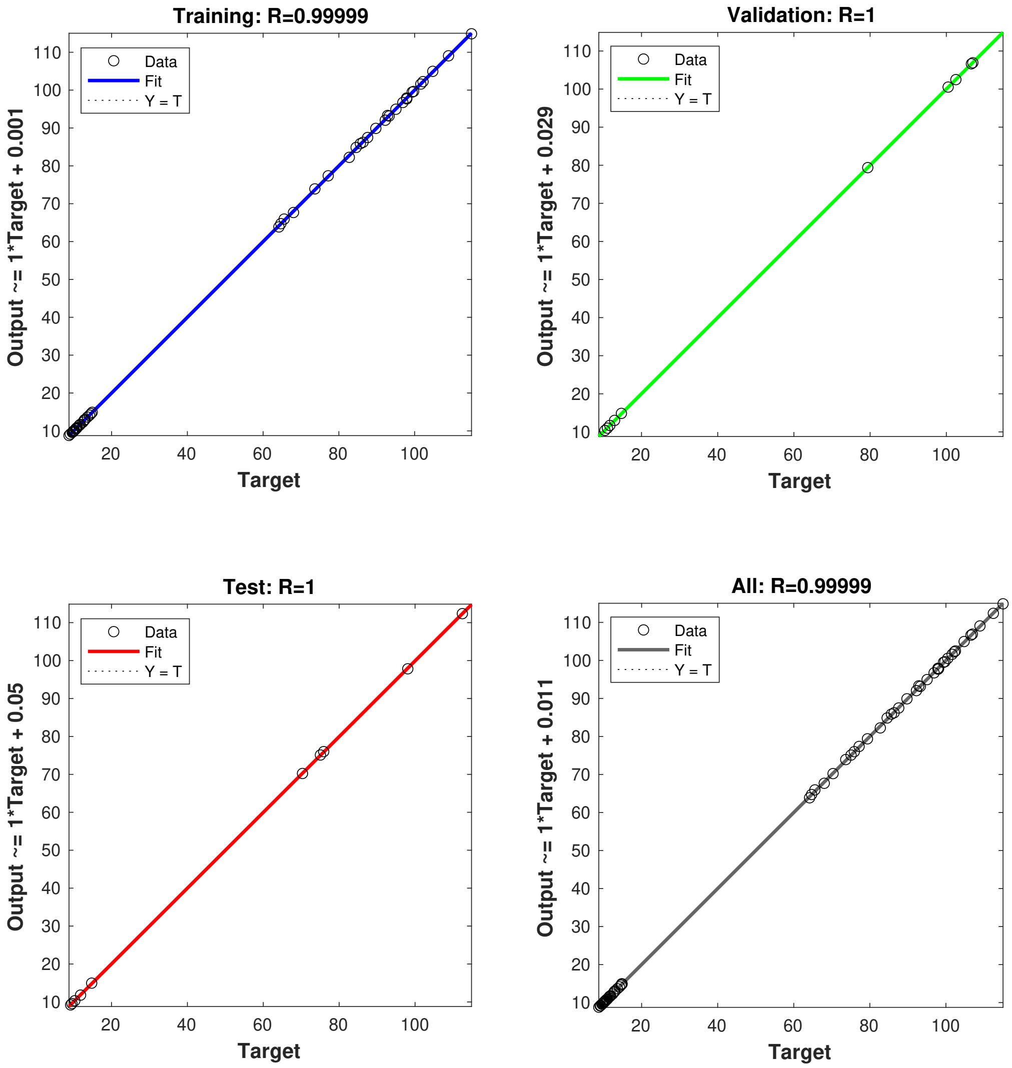
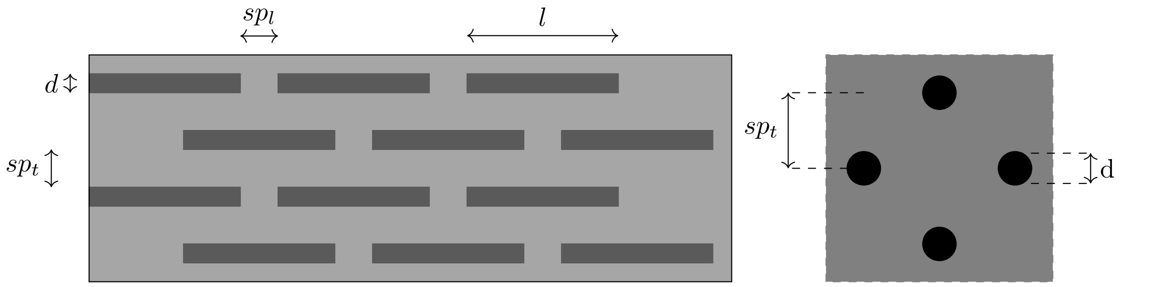


Figure 5.16: Ten neurons -One hidden layer fitting plots

R-value of 0.99 for training was obtained and 1 was obtained for validation and 1 for testing. For a perfect fit data should fall along a 45° angle line with R-value of 1 however R-values above 0.93 are considered accurate enough by Beale et al. [3], Ocampo et al. [35]. The numbers shown next to the target values on the y-axis in Figure 5.16 indicate the deviation of the actual outputs from their corresponding target values. The results of this training are then saved, that is weight and biases of this these network are saved and the network is now considered to be trained. With the weights and biases known any input that is combination of radii in range $0.1 \leq r \leq 0.5$ can be given to the network and out comes the effective elastic properties G and K. This trained network can then be used for other purposes such as optimisation as explained in Chapter 6 ahead.

5.5 Example 2: Transversely Isotropic Material

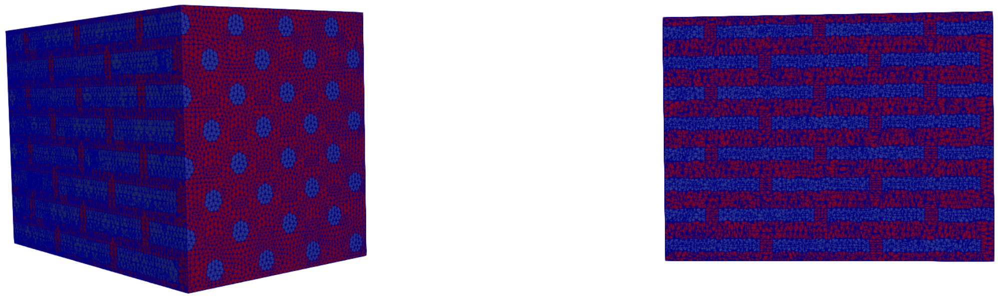
This is an example where fibres are arranged in a discontinuous brick like pattern with a changing spacing between the fibres. A varying non-dimensionalised fibre spacing in the longitudinal direction sp_l/d and transverse direction sp_t/d together with a changing fibre length for varying the aspect ratio $\alpha = l/d$ are used to generate sample microscopic RVEs. Five different lengths of inclusions l and five values each for sp_l and sp_t were experimented with, resulting in 125 test RVEs. This larger sample size allows for all weights and biases to be determined. It is noteworthy that the longitudinal spacing sp_l is parallel to the fibre length l while the transverse spacing sp_t is perpendicular to the fibre length l.Figure 6.11 shows the arrangement of discontinuous fibres used in this example.



(a) Transverse view of fibre arrangement

(b) Longitudinal view of fibre arrangement

Figure 5.17: RVE parameters of interest

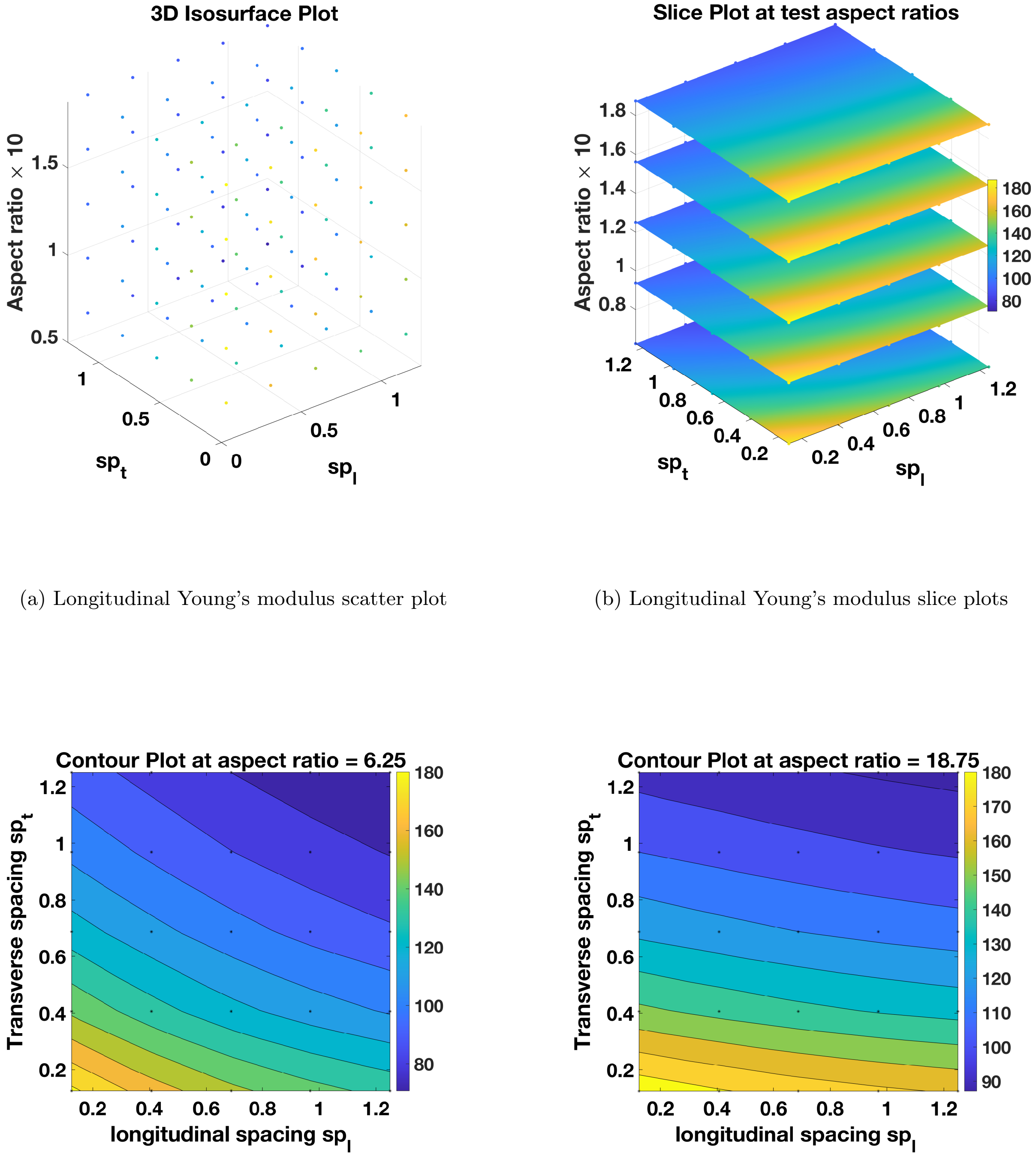


(a) Slice through discontinuous Fibre Microscopic RVE

(b) Same RVE But different view of the microstructure RVE

Figure 5.18: RVEs For Training

The Figure 5.19 below demonstrates how the longitudinal Young's modulus E_1 in MPa changes with the fibre spacing and aspect ratio. Changes in aspect ratio were obtained by varying the fibre length while the fibre diameter remained constant.



(a) Longitudinal Young's modulus scatter plot

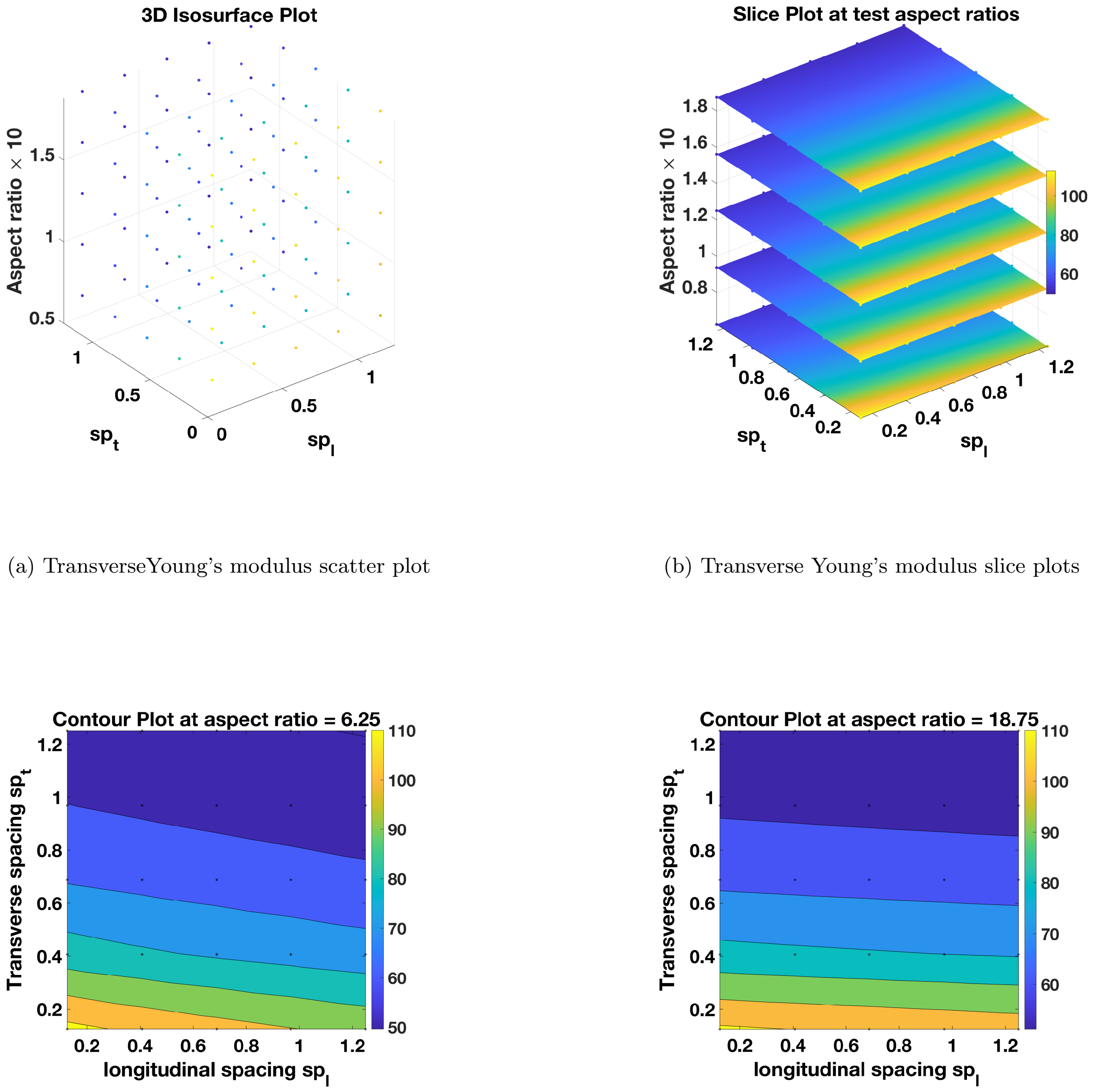
(b) Longitudinal Young's modulus slice plots

(c) Young's modulus at aspect ratio = 6.25

(d) Young's modulus at aspect ratio = 18.75

Figure 5.19: Longitudinal Young's Modulus

Figure 5.19 illustrates the relationship between Young's modulus, aspect ratio, and transverse and longitudinal fibre spacing. As shown by Figure 5.19, reducing the fibre spacing in both the transverse and longitudinal directions increases the overall Young's modulus of the composite. This is because the fibre-matrix volume ratio increases, leveraging the superior properties of the fibres. Short, thicker fibres (lower aspect ratio) with lower fibre volume ratios are significantly affected by longitudinal spacing. However, at higher aspect ratios, longitudinal spacing has less influence, as most of the load is carried by the fibres. Figure 5.20 below presents the transverse Young's modulus E_2 as fibre spacing in both transverse and longitudinal directions change with the aspect ratio.



(a) TransverseYoung's modulus scatter plot

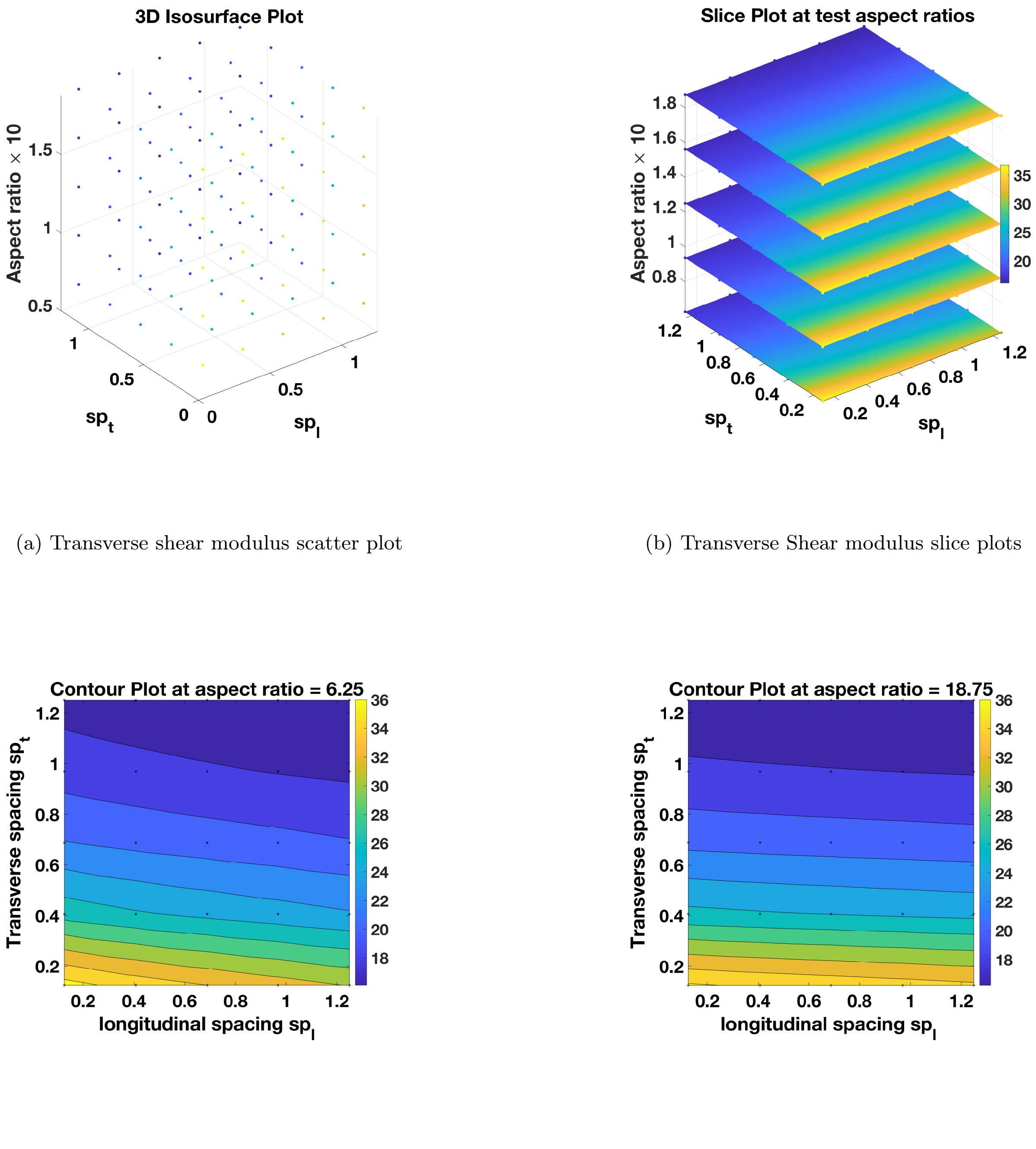
(b) Transverse Young's modulus slice plots

(c) Young's modulus at aspect ratio = 6.25

(d) Young's modulus at aspect ratio = 18.75

Figure 5.20: Transverse Young's Modulus

Figure 5.20 results show that transverse spacing has a significant effect on the transverse Young's modulus across all tested aspect ratios, whereas longitudinal spacing only has a comparable impact at higher sp_l values and lower aspect ratios. This is because most of the load under transverse loading direction is carried by the matrix, which has lower mechanical and elastic properties. As a result, the transverse Young's modulus is lower compared to the longitudinal Young's modulus. Larger spacing in the transverse direction means a higher matrix volume ratio, which leads to a lower transverse Young's modulus. Figure 5.21 below displays how the tranverse shear modulus G_{23} changes with the aspect ratio together with the longitudinal and transverse spacing of the fibres.



(a) Transverse shear modulus scatter plot

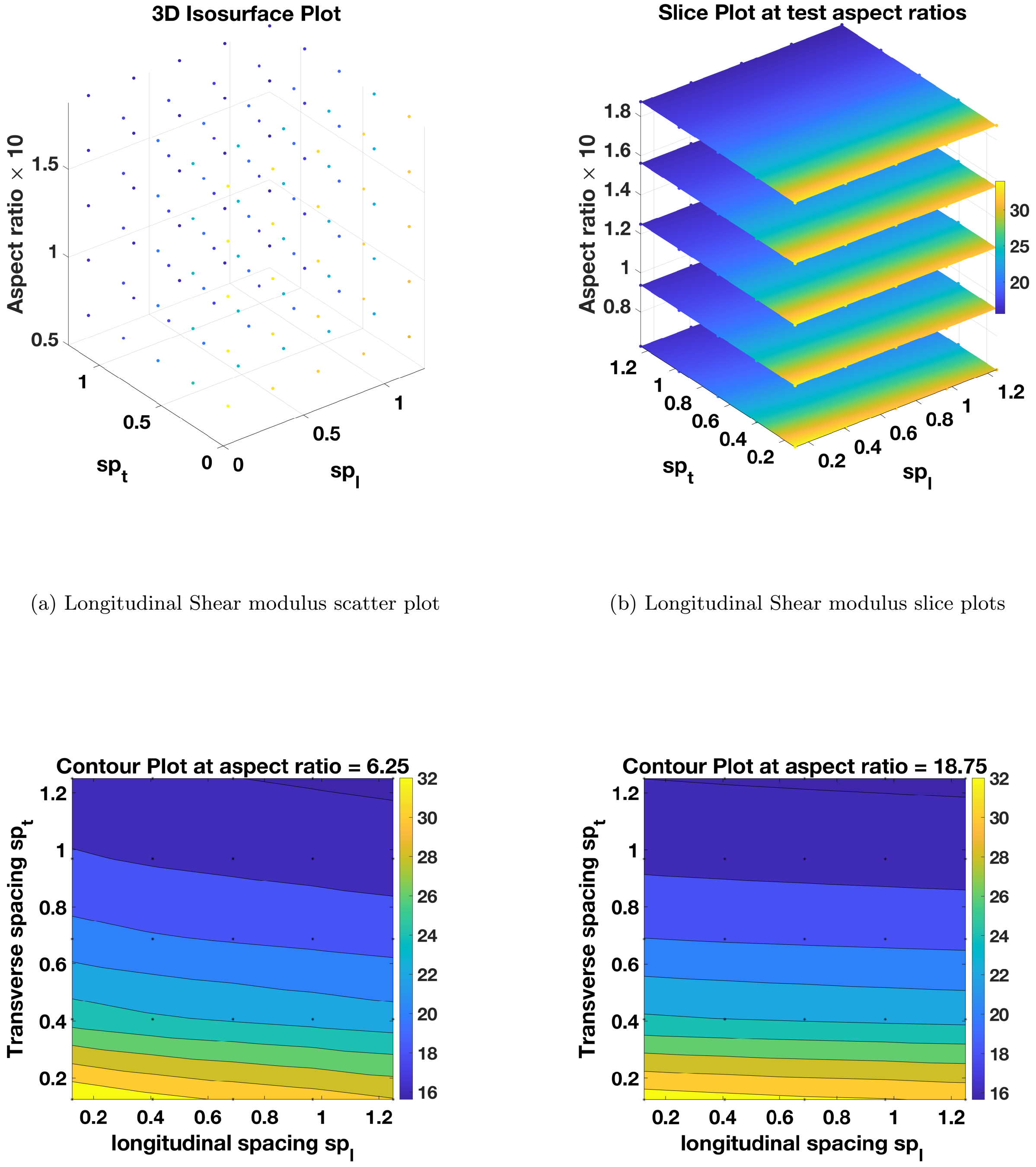
(b) Transverse Shear modulus slice plots

(c) Shear modulus at aspect ratio = 6.25

(d) shear modulus at aspect ratio = 18.75

Figure 5.21: Transverse shear Modulus

The transverse Young's modulus is affected by the longitudinal spacing at lower test aspect ratios because, in the transverse direction, most of the load is carried by the fibres under pure shearing. Therefore, larger spacing reduces fibre efficiency. However, at high aspect ratios, these changes might have minimal effects as shown by Figure 5.21d and 5.21c. Figure 5.21 below displays how the longitudinal shear modulus G_{12} changes with the aspect ratio and the spacing in longitudinal and transverse spacing of the fibres.



(a) Longitudinal Shear modulus scatter plot

(b) Longitudinal Shear modulus slice plots

(c) Shear modulus at aspect ratio = 6.25

(d) Shear modulus at aspect ratio = 18.75

Figure 5.22: Longitudinal shear Modulus

In Figure 5.22 longitudinal shear that is shear along the direction of fibres is less affected by the longitudinal spacing for all test aspect ratios. As expected the longitudinal shear modulus tends to be lower than transverse shear as shearing load in this direction is mostly carried by the matrix. As fibres spacing *sp* is reduced in any direction composite properties get better. Using the data from Figure

5.19, 5.20, 5.21, and 5.22 a neural network is trained for a given combination of aspect ratio r and spacing sp in both longitudinal and transverse direction so that the elastic properties E_1, E_2, G_{12} and G_{23}, can be approximated for other RVEs which were not tested on. The results in Figure 5.24 are of a trained neural network using, row vector of aspect ratio , the longitudinal and transverse fibre spacing as inputs. The row vectors of transverse and longitudinal shear modulus and Young's modulus were used as outputs for training. A tansig transfer function for hidden layer 1 and 3 and logsig for hidden layer two were used. Five neurons in the first layer and third layer, fifteen neurons in second layer were utilised for training. Figure 5.23 shows a neural network diagram used for this training.

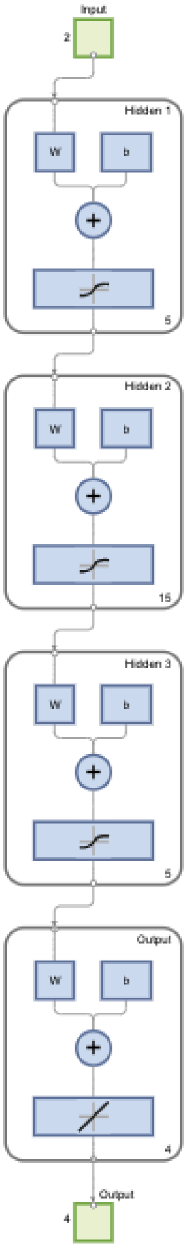


Figure 5.23: Network Diagram

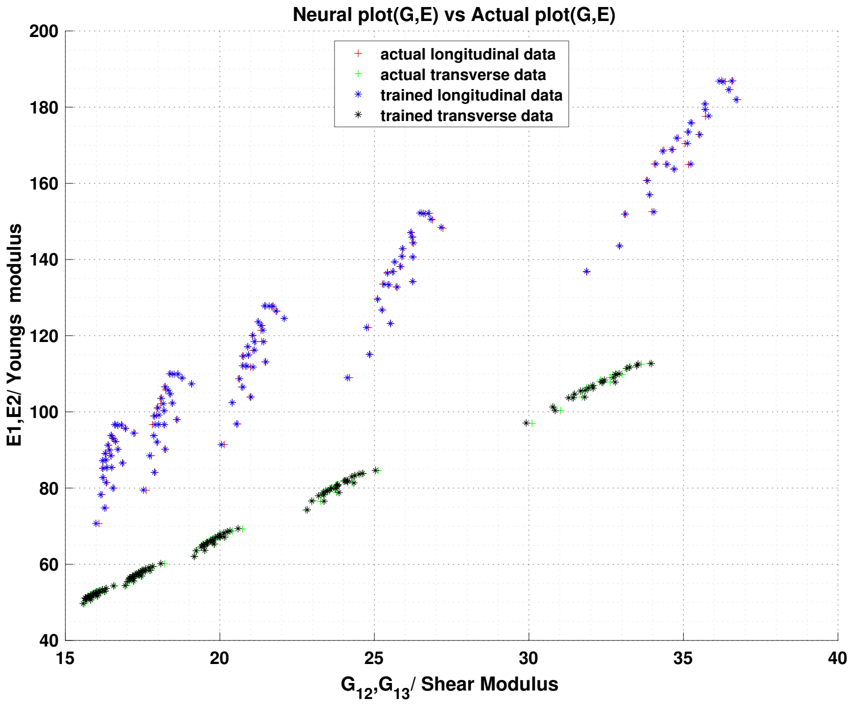


Figure 5.24: Scatter plot of Young's modulus vs shear modulus for both longitudinal and transverse directions

Figure 5.24 shows how well the neural network approximated the known data point. For both the transverse and longitudinal directions the model performed well to approximate the results where red and green points are known data point from the experiment for longitudinal and transverse directions respectively and blue and black data points are the neural network approximations for longitudinal and transverse directions respectively. A further performance analysis is carried out to see how well the data fits before saving the trained network. Figure 5.25 and 5.26 shows how well the trained network performed.

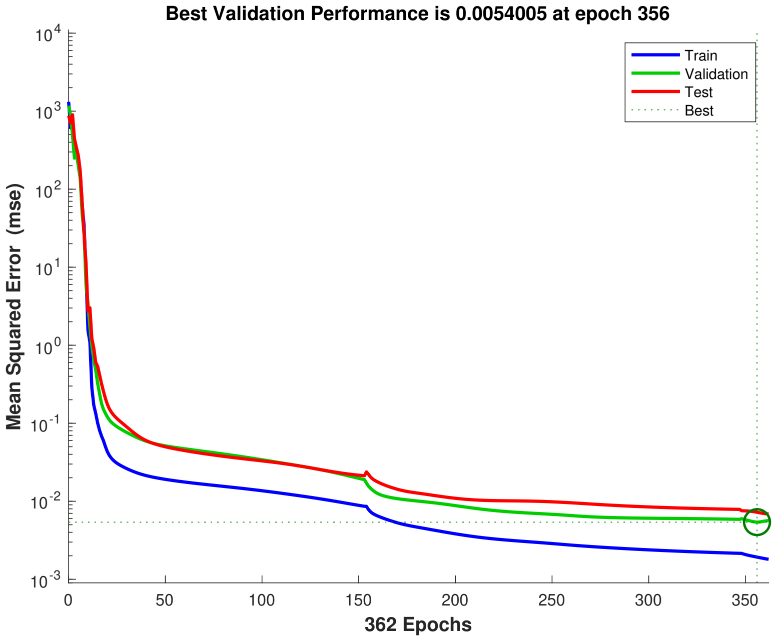


Figure 5.25: MSE with respect to epoch

Best validation performance was obtained at epoch 356 with a MSE of 0.0054 as indicated by Figure 5.25. The process then stops at epoch 362 and the best performance is used for the trained network. That it is the epoch 356 weights and biases are stored and used for a trained network and can be used to predict transverse/Longitudanal shear or Young's modulus for a given combination of aspect ratio and fibre spacing. Figure 5.26 shows how the validation, train and test fits with the targets that is $G_{12}, G_{23}, E1, E2$ from the data provided for training.

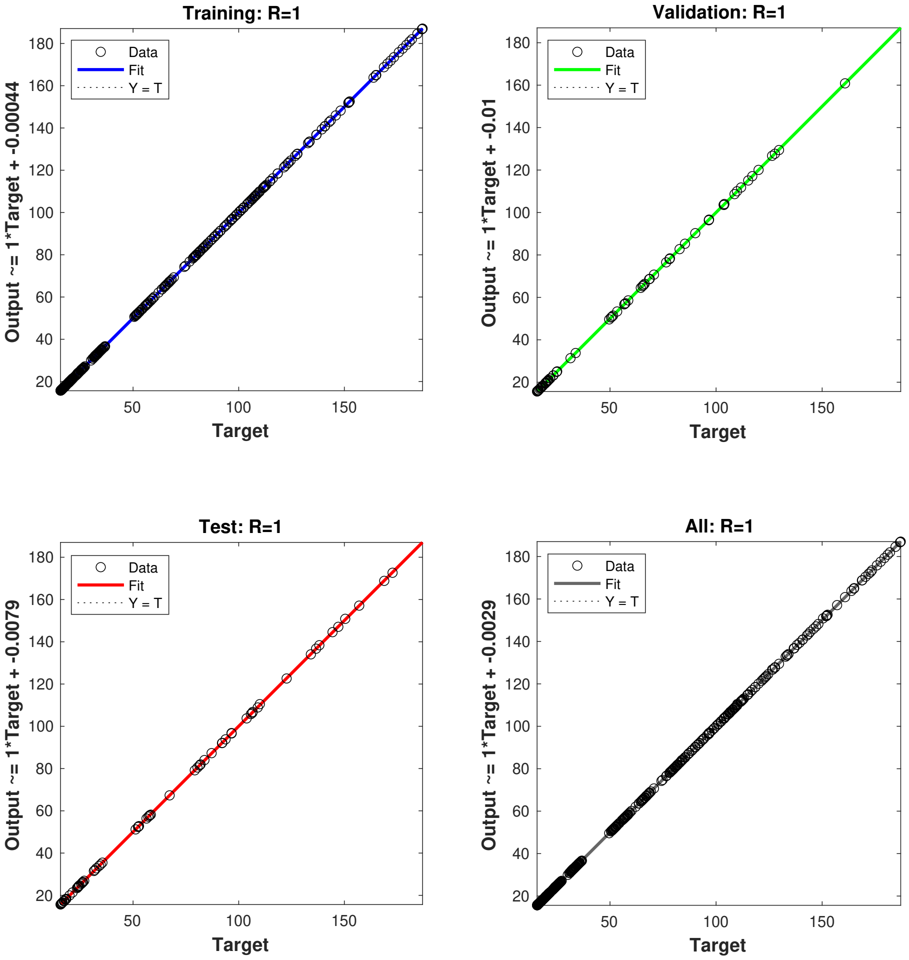


Figure 5.26: Five neurone-Three hidden layers results

R-value of all three graphs is graphs is 1 and all three graphs combined gave an R-value of 1 which indicate a very strong fit. As a results the network training is complete and it can now be saved and be used for other desired purposes. Neural network produce very accurate results in a shorter period of time compared to other natural methods. A well trained network is a powerful tool which is trained once and saved then be used for other purposes. Using more hidden layers or more neurones does not always improve the outputs as a results the number of hidden layers and number of neurone in each layer has to be chosen carefully to avoid overfitting or under-fitting. If the target/data does not fit well with neural network results the training has to be carried out again to update the weights and biases before using the saved trained network for other purposes. It should be noted that each training produces different weights and biases that is two networks with same neurones numbers in a layer and same number of layers can produce different fits. As a results training should be carried out carefully to fit the data.

Chapter 6

Optimisation

6.1 Objective and Outline of Methodology

The goal of this chapter as summarised in Figure 6.1, is to optimise the microstructure by adjusting parameters such as aspect ratio and fibre spacing to achieve the best possible elastic properties based on a defined criterion such as the cost or weight of the material. This requires the optimisation of conflicting objectives and therefore, there is no a solution that satisfy all the objectives hence their trade-offs (know as Pareto-Optimal solution) becomes the solution to the multi-objective problem. However, it's crucial to maintain the integrity of the material properties throughout this process. This means striving to achieve better material properties for all defined criterions. The method adopted for this chapter uses techniques which are fully described by [4, 54, 9, 30]. This method uses the MATLAB built in function `gamultiobj` algorithm outlined in details in MathWorks [30].

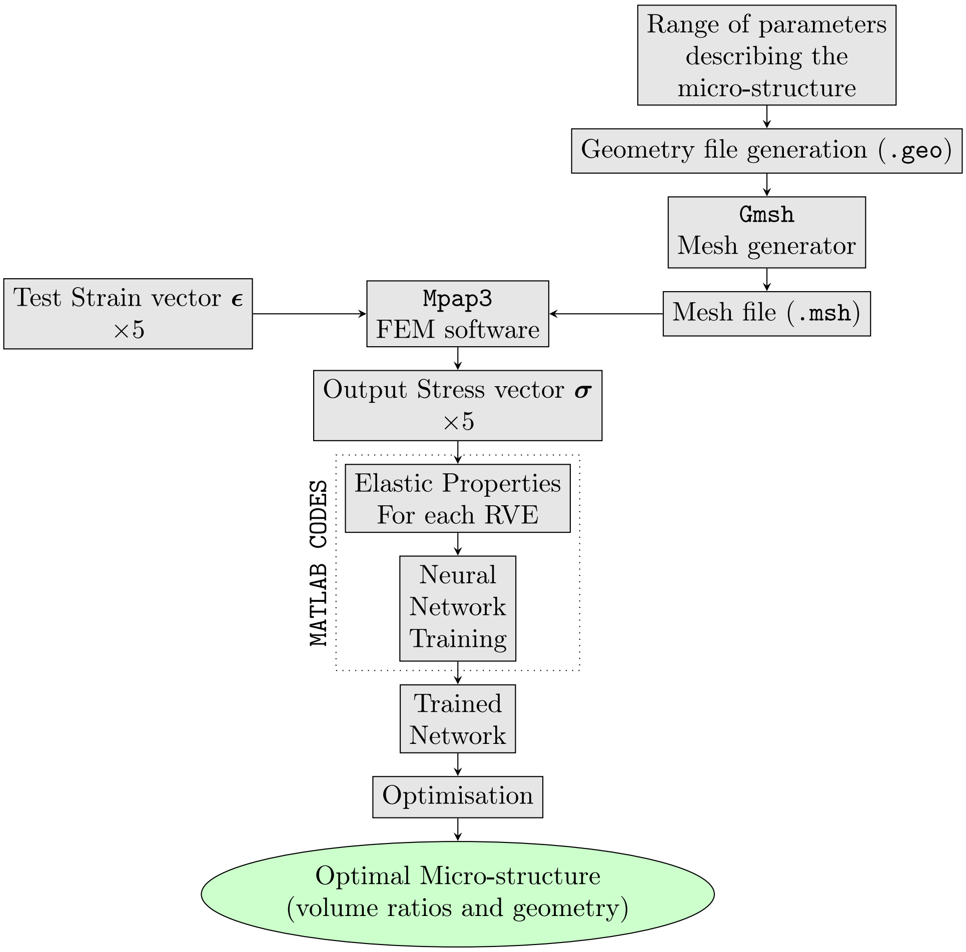

Range of parameters describing the micro-structure
Geometry file generation (.geo)
Gmsh
Mesh generator
Mesh file (.msh)
Test Strain vector $\boldsymbol{\epsilon}$ ×5
Mpap3
FEM software
Output Stress vector $\boldsymbol{\sigma}$ ×5
MATLAB CODES
Elastic Properties For each RVE
Neural Network Training
Trained Network
Optimisation
Optimal Micro-structure (volume ratios and geometry)

Figure 6.1: Procedural Methodology Utilized

6.2 Multi-objective Optimisation

A multi-objective function is defined as a vector that maps a set of m decision variables to a set of n objectives. Optimisation of this problem involves minimisation or maximisation of the objective vector $\mathbf{F}(x)$ which can be written as follows:

$$\begin{aligned} \text{min/max} \quad & \mathbf{F}(x) = [f^1(x), f^2(x), ..., f^m(x)] \\ \text{subject to } & x = (x^1, x^2, ..., x^n) \quad x \in \mathbb{R}^n \\ \text{subject to constrain} \quad & \phi_i(x) = \begin{cases} 1 & \text{when } m_e \leq i \leq m \\ 0 & \text{when } i < m_e \end{cases} \\ \text{subject to bounds} \quad & lb \leq x \leq ub \\ & y = (y^1, y^2, ..., y^m) \quad y \in \mathbb{R}^m \end{aligned} \tag{6.1}$$

Where x and y represent the decision and objective vectors within the space $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ respectively. The set of solutions of a multi-objective function consists of all decision vectors for which the objective vector cannot improve in any dimension without compromising the other dimensions. An increase in the number of objectives increases the complexity of finding or quantifying these trade-offs, hence numerically and realistically formulated design problems address natural problems. These optimal solutions are referred to as Pareto optimal solutions. This concept can be simplified and explained by a domain Γ in parameter space and x is an element of the n-dimensional real numbers that satisfy all constraints and bounds.

$$\Gamma = \{x \in \mathbb{R}^n\}. \tag{6.2}$$

From this parameter region a correlative admissible domain $\mathbb{A}$ for the objective space can be obtained as

$$\mathbb{A} = \{y \in \mathbb{R}^m\}. \tag{6.3}$$

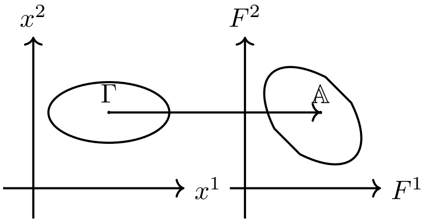


Figure 6.2: Parameter space into Objective Function Space

Non-dominant solution point x^* where $x^* \in \Gamma$ can now be defined for some neighbouring solution of x^* where there is no Δx such that $(x^* + \Delta x) \in \Gamma$ and

$$\begin{aligned} F^i(x^* + \Delta x) \leq F^i x^* \qquad & i = 1, ...m \\ F^j(x^* + \Delta x) < F^j x^* \qquad & \text{for some} \quad j. \end{aligned} \tag{6.4}$$

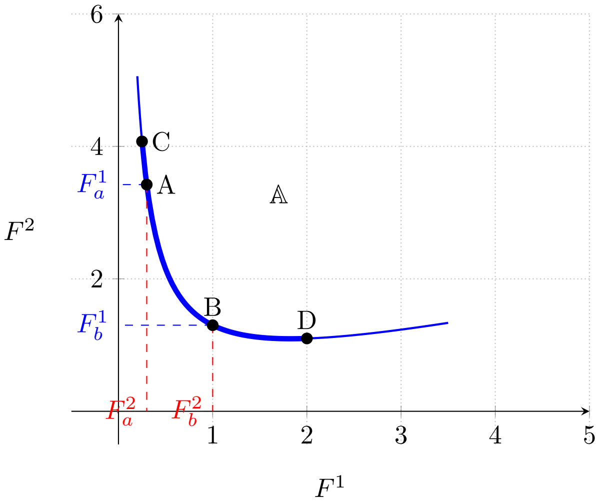


Figure 6.3: Noninferior Set of solution

Figure 6.3 shows a set of non dominant solution which lies on the curve between C and D while A and B are specific non inferior points on the curve. It can be shown that point A and B are non dominant solutions as there is no increasing one objective F^1without reducing the other objective F^2 as explained by Equation (6.2). It should be noted that, any point in the domain Γ that is not inferior implies that improvements can be made in one or all of the objectives therefore, such points are of no importance. The MATLAB built in procedure used for this non dominated solutions selections is simplified in Figure 6.4.

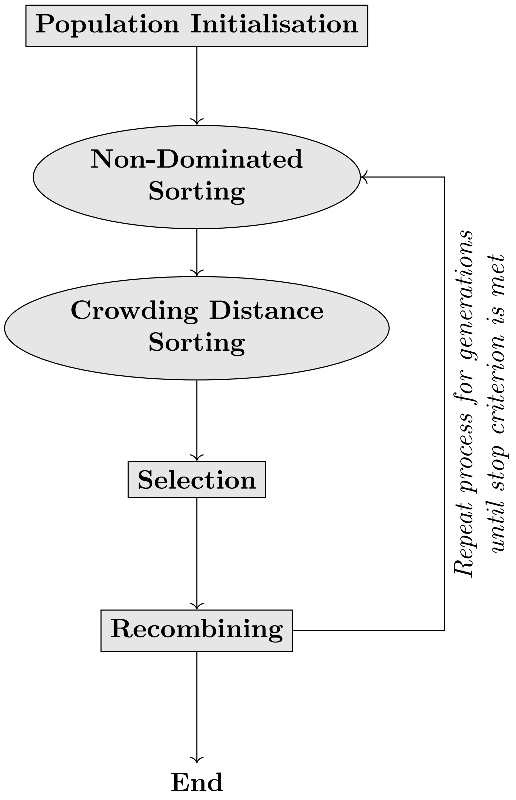

Population Initialisation
Non-Dominated Sorting
Crowding Distance Sorting
Selection
Recombining
End
Repeat process for generations until stop criterion is met

Figure 6.4: Optimisation Procedure

The population is initialized according to the problem's range and any associated constraints which can be linear or non-linear. In order for point p to be considered dominant over point q for an objective function F the following conditions given by

$$\begin{aligned} F^i(p) \leq F^i(q) \quad \forall i, \\ F^j(p) < F^j(q) \quad \forall j. \end{aligned} \tag{6.5}$$

needs to be satisfied. Therefore, a non-dominant set q among a set of points p is defined as the set in which all points in q are not dominated by any point in p. For each solution we look for the domination count /rank d_{cp} which refers to number of solutions which dominates solution p and a set of solution S_p that p dominates. The solution of non-dominant front will have a $d_{cp} = 0$ that is rank zero. For each solution p with $d_{cp} = 0$ in q of set S_p visit each member of q of set S_p and reduce dominance count by one. And if by any chance the rank of a member in q is zero we put it in a list Q and the set is used in second non-dominant front.This process is repeated until all fronts are found. In general the second or high order the non-domination count or rank can be expressed as $k - 1$ at most. The following Algorithm 3 borrowed from Deb et al. [9] is used as part of the optimisation process in MATLAB built in function `gamultiobj`.

Algorithm 3 Fast-Non-Dominated Algorithm

```
For each $p \in P$
$S_p =$                              This empty set will store all solutions dominated by $p$
$d_{cp} = 0$                          Initialisation number of solutions dominated by $p$
For each $q \in P$
if $q \succ p$ then
    $S_p = S_p \bigcup \{q\}$          Then combine $q$ and set of all solutions dominated by $p$
    if $p \succ q$ then
        $d_{cp} = d_{cp} + 1$          Increase the domination count of $p$
    end if
end if
if $d_{cp} = 0$ then
    $P_{rank} = 1$                     Solutions $p$ belong to rank 1
    $F_1 = F_1 \bigcup \{P\}$            Add $p$ to update first front sent
end if
$i = 1$                               Initialise the front counter
while $F_1 \neq$ do
    $Q \neq$                           This non empty set will store members of the next front
    for each $p \in F_1$
    for each $q \in S_p$
    $d_{cp} = d_{cp} - 1$              For each $q$ decrease the dominance count by one
    if $d_{cp} = 0$ then
        $P_{rank} = i + 1$             $q$ is member of the next front
        $Q = Q \bigcup \{q\}$            Update the set Q with $q$
    end if
    $i = i + 1$                        Update the loop counter
    $F_1 = Q$                          Set $Q$ is the next front
end while
```

Crowding Distance is computed as the average distance between two points surrounding a particular solution in a population on either side along each objective function. In short, it measures how close individuals are to each other. This is measured for members which belong to the same rank in the objective function space. Individuals who are at boundaries, where solutions with the smallest or largest function values are assigned, have an infinite crowding distance. For the rest of the individuals, Algorithm 4 by Deb et al. [9] with little changes is used for the calculation of crowding distance in MATLAB built in function `gamultiobj`.

Algorithm 4 Crowding Distance Assignment

```
$m = |I|$                                   Number of individuals or solution in $I$
For each $i$, set $I(i)_{dist} = 0$         Initialise the distance
For each objective $n$
$I = sort(I, n)$
$I(1)_{dist} = I(m)_{dist} = \infty$        Boundary values assigned infinite crowding distance
$i = 2$ to $(m - 1)$                        for all point
$I(i)_{dist} = I(i)_{dist} + (I(n, i + 1) - I(n, i - 1))$
```

The binary tournament selection is carried out using crowded comparison operator ($\prec_n$) at different stages as follows towards the spread.

1. Non dominant rank p_{rank} for all Front F_i are assigned $p_{rank} = i$

2. Crowding distance I_{dist}
   $p \prec_n q$ if $p_{rank} < q_{rank}$ or p and q belong to same front F_1 and $I_{dist}p > I_{dist}q$

In conclusion an individual with a lower rank is preferred followed by and individual with larger crowding distance. We combine the offspring population with the current population, and selection is performed. The population is sorted based on non-domination and assigned a rank equal to its non-domination level. They are then added to the subsequent population until the population size exceeds N. When this happens, the members of this front are selected based on crowding distance until the population size is N. If the size of F_i is smaller than N we choose all members to the next set. This is summarised by [9] as shown in Algorithm 5.

Algorithm 5 Main Loop

```
$R_t = P_t \bigcup Q_t$                              Parent and offspring combined
$F =$ fast non dominated sort$(R_t)$                  All dominated fronts of $R_t$
$P_{t+1} = 0$ and $i = 1$
Until $|P_{i+1}| + |F_i| \leq N$                     All population should be filled
crowding distance assignment$(F_i)$
$P_{t+1} = P_{t+1} \bigcup F_i$                      Upadate the parent population
$i = i + 1$
Sort $(F_i, \prec_n)$                                Use $\prec_n$ to sort in descending order
$P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$    choose the first $(N - |P_{t+1}|)$ of $F_i$
$Q_{i+1} =$ make new population$(P_{t+1})$           new population
$t = t + 1$
```

The spread is used as the stopping criterion for the MATLAB function `gamultiobj`. It indicates how the Pareto set is spread. This is determined by calculating the standard deviation of the crowding distance of points within a finite distance.This can be calculated as

$$S_{pr} = \frac{\bar{x} + \sigma}{\bar{x} + Qd} \tag{6.6}$$

where $\bar{x}$ is the sum over m objective indices of the norm differences between current and previous minimum value of Pareto points of that indeces. Q is the number of this points /solutions which are within the finite distance with d as the average distance between this points.

6.3 Example 1: Spherical Inclusions

This section uses the trained Neural network from Section 5.4 to optimise the microstructure by adjusting the radius of voids and inclusions. The network computes the bulk modulus K and and shear modulus G from the varying combination of void and inclusion radii of the microstructure. The radii of voids and inclusion (x^1 and x^2) also known as decision variables/vectors x are used to create objective functions. One objective function $f^1(x)$ minimises the cost while the other objective function $f^2(x)$ maximises the properties K and G as

$$F^1 = V_m C_m + V_i C_i \tag{6.7}$$

$$F^2 = a \left[\frac{max(0, K - K_{mn})}{K} + \frac{max(0, G - G_{mn})}{G} \right] \tag{6.8}$$

$$[K, G] = ann([x^1, x^2]) \tag{6.9}$$

Where V_m and V_i are the volume ratios of matrix and inclusions respectively which are functions of their respective radii, C_m and C_i are the relative costs of the matrix and the inclusion respectively. K_{mn} and G_{mn} are the minimum design requirements for bulk modulus and shear modulus respectively. a is a parameter which can be used to adjust the weight or contribution of the F^2 as compared to F^1. `ann` is the saved trained neural network which uses the decision variables to compute the elastic properties. The objective functions are saved as a MATLAB code called Cost in Appendix B.7.

Algorithm 6 outlines the MATLAB code used to determining the optimal Pareto solutions. This code uses the MATLAB built in function `gamultiobj` to evaluates the trade-offs between competing objectives to identify the most efficient solutions. Each step in the algorithm contributes to the refinement of the solution set, ensuring that the final results are Pareto optimal.

Algorithm 6 MATLAB code for Optimisation

```
Load the saved trained neural network: load('my_neural_network_model.mat')
Define the number of decision variables: $nvars = 2$
Define lower bounds for the decision variables: $LB = [lb_1, lb_2]$
Define upper bounds for the decision variables: $UB = [ub_1, ub_2]$
Set the options for the genetic algorithm:
  options = optimoptions('gamultiobj',
  'SelectionFcn', @selectiontournament, 50,
  'ConstraintTolerance', 1e-6,
  'CrossoverFraction', 0.75,
  'PopulationSize', 100,
  'MaxGenerations', 100,
  'PlotFcn', @gaplotparetodistance)
Run the multi-objective genetic algorithm:
  [b, fval, exitflag, output] = gamultiobj(@(x)cost(x, mlnet), nvars, [], [], [],
  [], lb, ub, [], options)
Display the optimized solution and objective function values:
  disp('Optimized void & incl radii Solution:')
  comb_of_ri_rv = b
  disp('Objective Function Values [C1 and C2]:')
  disp(fval)
```

For this example there are two decision variables, void radius and inclusion radius from which the volume ratios are calculated. Both decision variables have the same bounds which can be described as $0.1 \leq x \leq 0.5$. The `gamultiobj` function in Matlab's Generic Algorithm Toolbox is used for multi-objective optimisation. The goal of this function is to find a set of non-inferior (Pareto optimal) solution for trade off between conflicting objectives. The fitness function `cost` is defined by 2 objective functions in a separate file.

Each objective function takes a set of decision variables (radii of voids and inclusions) represented as x^1 and x^2, as the input. These input are then passed through a trained Neural Network which is used by the cost function to compute the corresponding objective values. The bounds constrains the decision variable to remain within the desired bounds. For this example, bounds used were $0.1 \leq x \leq 0.5$, which was the same range as the radii used for inclusions in the test data training the neural network. Neural networks tend to perform well when operating within the range of the training data.

Other options required include, population size, a parameter used to control the number of solutions in each generation. Pareto optimal solutions are chosen from these individuals, crossover rate determines the rate at which the genetic material are exchanged to produce a new offspring between two individuals. The higher the rate the higher the diversity among the solutions hence a good convergency of the solutions can be obtained, the maximum generations parameter acts as the termination criterion by setting the maximum number of iterations that the algorithm runs to produce these sets of population sizes. Visualisation plots are used to monitor the optimisation progress. The process is summarised by Figure 6.5.

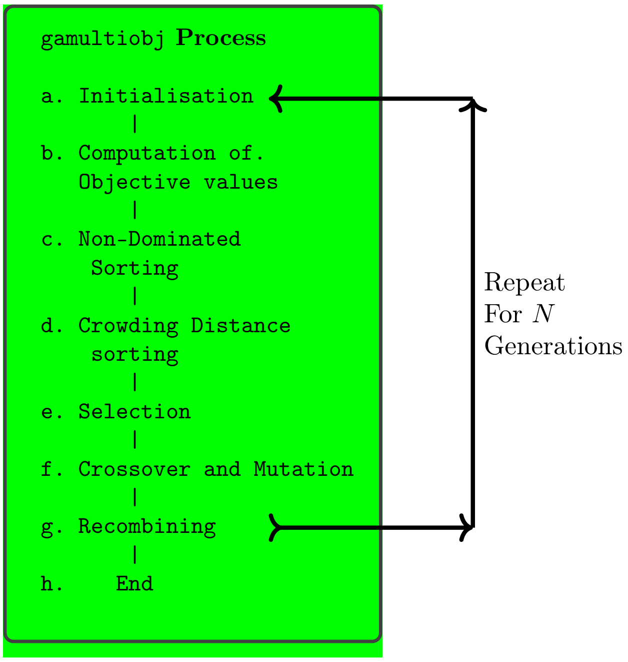

gamultiobj Process
a. Initialisation
b. Computation of. Objective values
c. Non-Dominated Sorting
d. Crowding Distance sorting
e. Selection
f. Crossover and Mutation
g. Recombining
h. End
Repeat For N Generations

Figure 6.5: Algorithm Steps `gamultiobj`

Parameters such as the population size, maximum generation and crossover rate are adjusted based on the problem and the convergency behaviour of the solutions. This algorithm uses visualisation plots to monitor the optimisation process. One of these plots is `gaplotdistance`, Figure 6.6 gives an idea on the diversity of individuals and the convergence of the solution population by showing the distance between individuals in a population. A decreasing trend indicate that the diversity of the population is improving or it is maintained. Quick decrease towards lower value is undesirable as it may indicate that the algorithm is stuck in a local optima or the solution space is not fully explored.

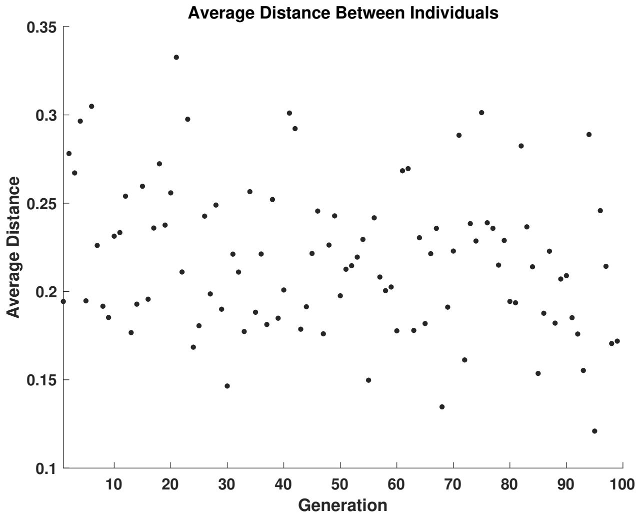


Figure 6.6: Distance between individuals

There is higher average distance indicating the diversity of the population and individuals are spread out in the decision variable space. The average distance between individual slowly decreases as number of generation increases indicating the convergence towards a possible solution. The average distance remain relatively higher which indicates that diversity is maintained. Figure 6.7 gives an idea into whether the algorithm explores and maintains sufficient diversity with the solution space or if the plot converges prematurely. Adjusting parameters such as crossover rate population size can help improve the results. The `gaplotparentodistance` in Figure 6.7 visualises the distances between Pareto solutions on the Pareto front. This help monitor how well Pareto fronts are explored as well as the diversity among non-dominated solutions. Increase or fluctuations trend may be used as a measure of diversity among non-dominated solutions. Rapid decrease towards lower values suggest that a further exploration of solution space is required.

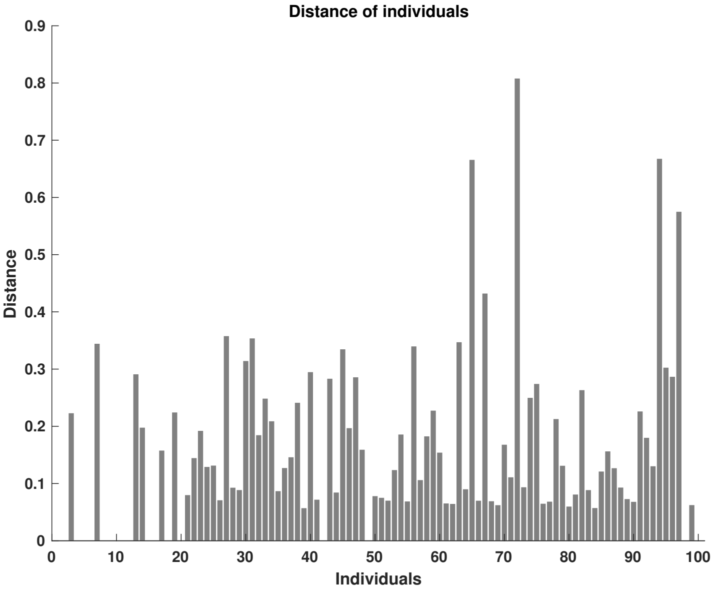


Figure 6.7: Distance between Pareto solutions

Figure 6.7 gives an idea about the diversity and convergence of individuals in the Pareto front during optimisation. The plot gives an insight into how well trade-offs are explored between the objectives. The `gaplotrankhist` in Figure 6.8 show how individuals are distributed into ranks. A descending trend from rank 1 to upper ranks is desirable as it indicates Pareto are well assigned into non-dominated subset.

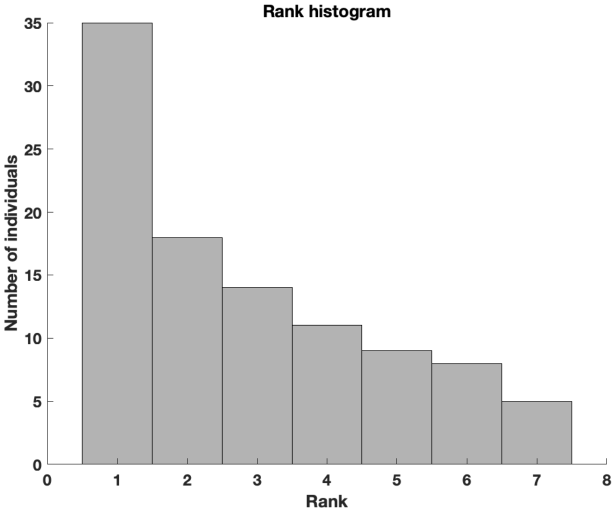


Figure 6.8: individual ranks

Figure 6.8 provides insight into selection pressure and convergence behaviuor based on fitness ranking. It illustrates how the population is distributed across different fitness levels, which reflects the efficiency of the selection process. The plot shows higher peaks at lower ranks, indicating that many individuals have similar fitness values. This suggests good convergence toward an optimal solution. A positively skewed histogram implies effective selection and propagation of high-quality solutions, further supporting the idea that the algorithm is converging. The presence of high selection pressure is also evident, which contributes to driving the population toward better solutions. The number of generations the algorithm performed before the process was terminated was 100 acted as the terminating criteria for the algorithm. Time taken by the algorithm can also be set to act as the stopping criteria. The Pareto front in Figure 6.9 presents a set of efficient non-dominated solutions that are chosen as optimal because neither objective can be improved without degrading the other. Objective 1 represent the cost and its negative so that it can be minimised while object 2 is the material property which is being maximised.

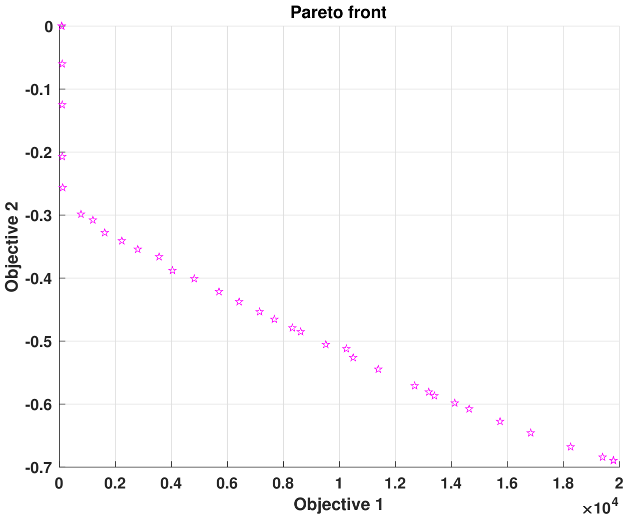


Figure 6.9: Pareto Front plot

Each point on the graph represent a solution on the Pareto front. These points are all non-dominated and represent different trade offs where improving one objective degrade the other. That is reducing the cost will reduce the material properties or improving the material properties increases the cost along this curve. The decision is made depending on the priorities and a solution is chosen from the Pareto front. The solution can be the one that balances the objectives to give design parameters. This graph is used as an aiding tool in decision making for multi objective optimisation problem. Optimised microstructure's parameters from Pareto front in Figure 6.9 are shown by figure 6.14.
The above optimal solutions in Figure 6.10 are then used to generate the optimal micro-structure of which together with the trained network the best material properties at lower cost without compromising the material's integrity can be achieved. The solutions are from the Pareto fronts and which acts as an aid in decision making. For this problem objective 1 is a function based on properties of the matrix and the fibre, while objective 2 is the cost functions. Since for this example it is assumed that fibres are more expensive than the matrix. It actually make sense that the optimised solutions suggest mostly little to no voids from Figure 6.10. Note that there are some anomalies in the plot where the cost are greatly reduced (i.e closer to no fibres at all since they are expensive) however this points results in very low shear and bulk modulus and my be undesirable for design.

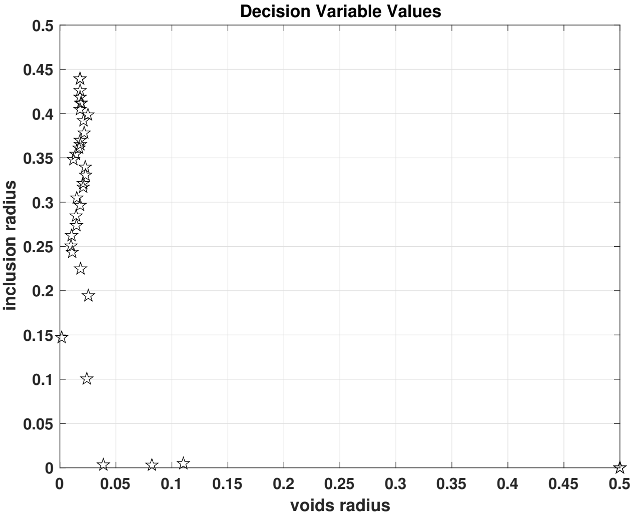
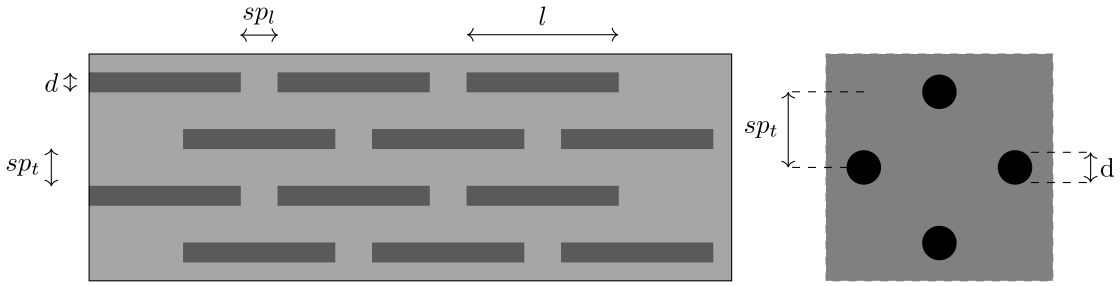


Figure 6.10: Optimised parameters if matrix is less expensive than fibers

6.4 Example 2: Discontinuous Fibres

This example involves discontinuous fibres of same diameter d with spacing between fibres sp_l, sp_t and sp_t in x, y and z directions shown by figure 6.11 below. The length of the fibre is then varied together with sp_t and sp_l to generate several RVEs which are then experimented on Using previous chapters. The diameter of the fibres remains constant throughout. After obtaining a trained network, an optimisation process is applied as described above to generate parameters an optimised microscopic structure where material properties are not compromised. That is the optimal combination of aspect ratio α and spacing sp_t and sp_l are identified and used for design. where:

$$\alpha = l/d \tag{6.10}$$



(a) Transverse view of fibre arrangement

(b) Longitudinal view of fibre arrangement

Figure 6.11: RVE parameters of interest

Transverse properties $E2$ and G_{23} and Longitudinal properties $E1$ and G_{12} are used to generate a trained network. Objective functions are generated as function of parameters sp_l, sp_t and α. That is :

$$F(sp_l, sp_t, \alpha) = [f_1(sp_l, sp_t, \alpha), f_2(sp_l, sp_t, \alpha)] \tag{6.11}$$

Where:

$$V_i = \frac{l\pi \frac{rd}{2}^2}{(sp_t + d)(l + sp_l)}, \quad V_m = 1 - V_i, \tag{6.12}$$

$$f_1(sp_l, sp_t, \alpha) = V_m C_m + V_i \times C_i \tag{6.13}$$

$$f_2(sp_l, sp_t, \alpha) = \sum_{i=1}^{4} \frac{max(0, S(i) - s(i)_{min})}{s(i)} \tag{6.14}$$

$$S = ann([sp_l, sp_t, \alpha]) \tag{6.15}$$

Function ann is a trained neural network which takes in three decision variables namely sp_l, sp_t and α to generate $E1, E2, G_{12}$ and G_{23} represented $S(i)$ where $i = 1, 2, 3, 4$ represent $E1, E2, G_{12}, G_{23}$ respectively as they appear in the neural network output. The decision variables are then set to $0.1 \leq sp \leq 0.8$ and $6 \leq \alpha \leq 25$. $C_i, C_m, s(i)_{min}$ represent density of fibre , density of matrix and minimum elastic property required for design. The average distances between individuals in the solution space are illustrated in Figure 6.12.

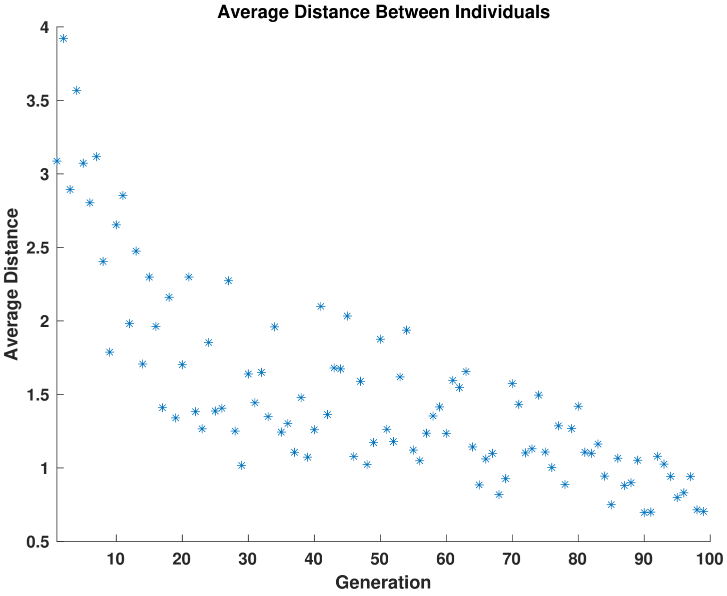


Figure 6.12: Distance between Pareto solutions

The average distance of individuals in a population shows a well scattered distribution indicating that a wide range of solutions were considered in decision making. A higher diversity was considered as this is indicated by higher average distance between individuals as indicated by figure 6.12 above. The reduction in the average distance is a an indicator that the solution is converging towards the solution. Figure 6.13 shows a rank plot of how individuals are allocated ranks:

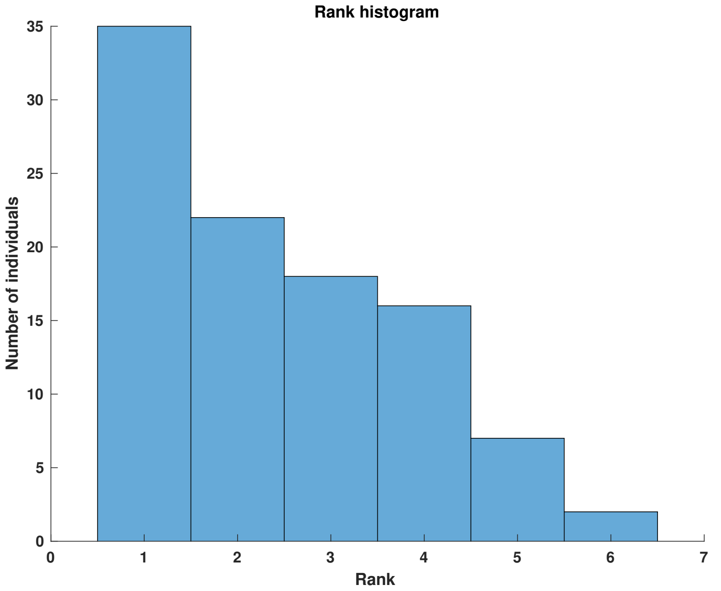


Figure 6.13: individual ranks

The solutions are ranked according to how many other solutions they dominate as explained earlier in this chapter. The histogram in figure 6.13 shows a positive skew indicating that more solutions in rank one showing more solutions that a re not not dominated by other solutions. This indicates that a better selection was carried out hence a better solution is selected. Pareto plot in figure 6.14 represents all solution that maximises elastic properties and minimises overall weight of the composite. Any improvement in one objective leads to degradation on the other. A decision is made based on the combination of sp_l, sp_t and α that results in point along this line.

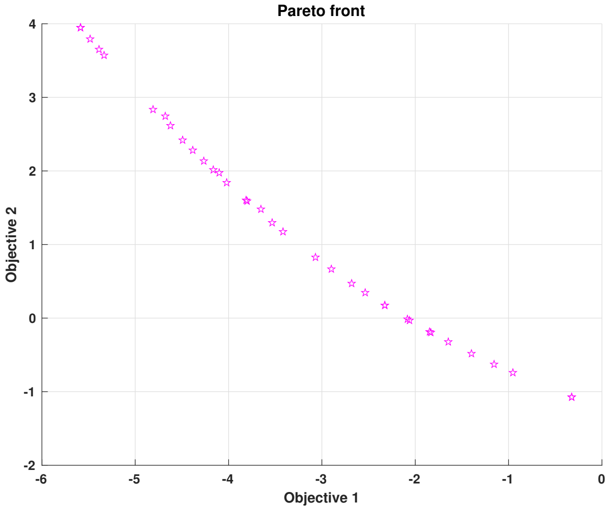


Figure 6.14: Pareto Front plot

Parento Distance plot in Figure 6.15 represents distances between all solutions of the pareto front. Diversity among the Pareto front is maintained as shown by varying distances between Pareto solutions therefore the decision variables considers a wide range of solutions within the Pareto fronts for the selection of sp_l, sp_t and α.

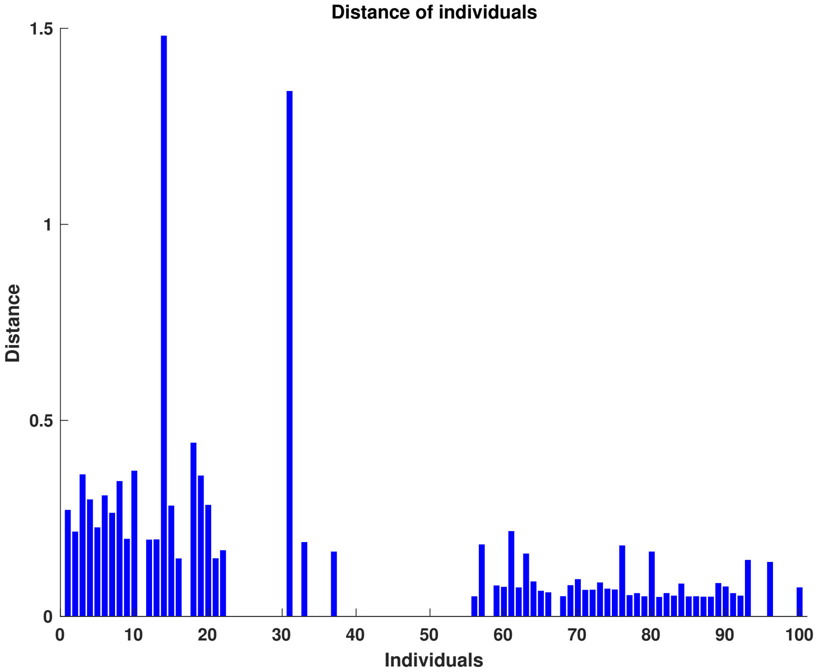


Figure 6.15: Distance between Pareto solutions

Figure 6.15 indicates that 100 generations were carried out before termination. Table 6.1 shows some of the values of sp_l, sp_t and α that resulted in the optamal solution in the pareto. This are the combinations that optimises the microstructure and can be regarded as the design parameters. Table 6.1 below shows the elastic properties which resulted from this decision variables:

E1	E2	G_{23}	G_{12}	sp_l	sp_t	aspect r	obj1	obj2
96.0162	53.1659	16.1250	16.7692	0.1513	1.2496	13.5216	-0.3632	-1.0571
90.5676	53.9872	16.5148	17.1616	0.7533	1.1171	10.1915	-2.1624	0.1520
81.1784	50.5074	15.6514	16.0871	1.4969	1.2492	12.6046	-5.6074	3.9746
89.7816	52.9845	16.2160	16.8176	1.0750	1.1453	14.0753	-3.4620	1.3284
97.5009	54.1919	16.4225	17.1107	0.3115	1.1656	13.4598	-0.7717	-0.7723
81.7494	50.5694	15.6534	16.0987	1.4788	1.2492	13.0722	-5.5130	3.8321
84.6039	51.2076	15.7720	16.2726	1.1085	1.2493	13.2349	-3.7222	1.4977
84.6051	51.1656	15.7578	16.2578	1.1342	1.2496	13.5201	-3.8378	1.6266
88.3065	51.9417	15.9151	16.4673	0.7115	1.2493	12.6992	-2.1065	-0.0003
84.1517	51.0626	15.7373	16.2295	1.1984	1.2493	13.5938	-4.1317	1.9689
85.4475	51.3546	15.7963	16.3096	1.0225	1.2496	13.3374	-3.3460	1.0974
82.7404	50.8050	15.6966	16.1638	1.3403	1.2496	13.1968	-4.8117	2.8331
83.6350	50.9591	15.7191	16.2021	1.2591	1.2493	13.5338	-4.4177	2.3201
82.2306	50.6448	15.6620	16.1166	1.4425	1.2493	13.3554	-5.3252	3.5540
86.8947	51.4128	15.7692	16.3049	0.9969	1.2492	15.1046	-3.2359	0.9867
88.0035	51.6458	15.8180	16.3679	0.8558	1.2494	14.7965	-2.6572	0.4465
86.9711	51.5256	15.8085	16.3434	0.9287	1.2494	14.2351	-2.9514	0.7118

Table 6.1: Results Table

The optimisation tool employed in this study enhances the microstructure by adjusting either the spacing or the size of the fibres within it. The tool is designed to utilise the trade-off between objective functions to identify the optimal combinations of fibre spacing and aspect ratio. In this example, the first objective is to maximise the material properties, while the second is to minimise the overall weight of the composite. The optimisation process identifies the most suitable combination of aspect ratio and spacing that achieves a reduction in weight without compromising the material properties. It is assumed that the fibres possess superior properties compared to the matrix, which is comparatively lighter. Figure 6.16 presents a heat map of five designs selected from Table 6.1, highlighting those with higher material performance.

Figure 6.17 shows that as fibres comes closer together and fibre length remain constant the elastic properties tend increases however the overall weight of the composite is compromised. Smaller fibre diameter or larger values of sp_l and sp_t means that there is more of the matrix which has lower elastic properties and a lightweight overall composite hence the overall properties of the composite are reduced. This is supported by the fact that matrix has inferior properties as compared to the fibres.

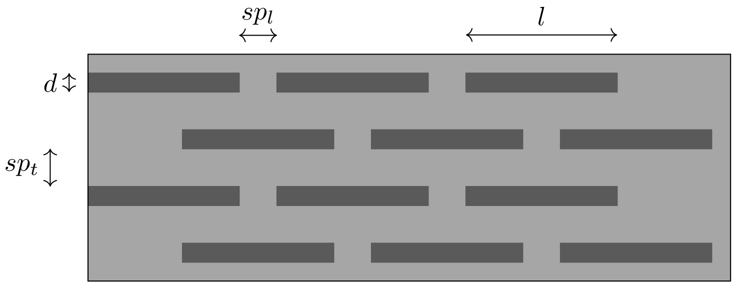
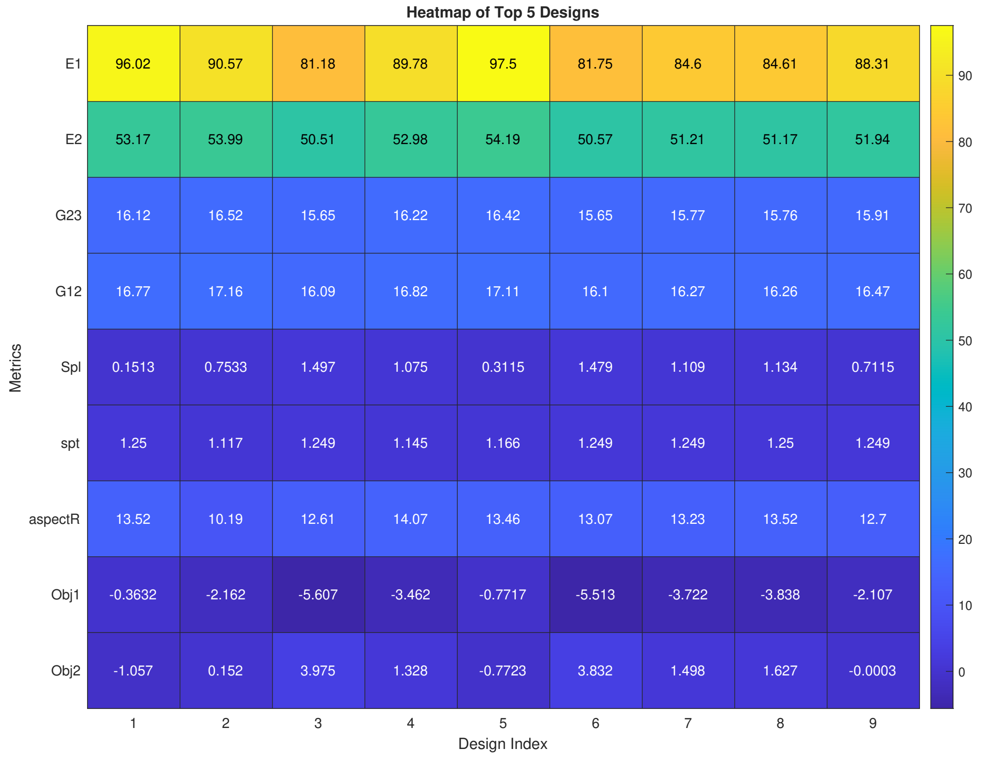


Figure 6.17: Transverse view of fibre arrangement

Figure 6.18 shows the correlation between the relevant variables. From the figure, it can be concluded that there is very low correlation between aspect ratio and either of the objectives, as the aspect ratio values are relatively small. However, there is a strong negative correlation between Objective 1 and Objective 2, as expected, improving one objective leads to a deterioration in the other.

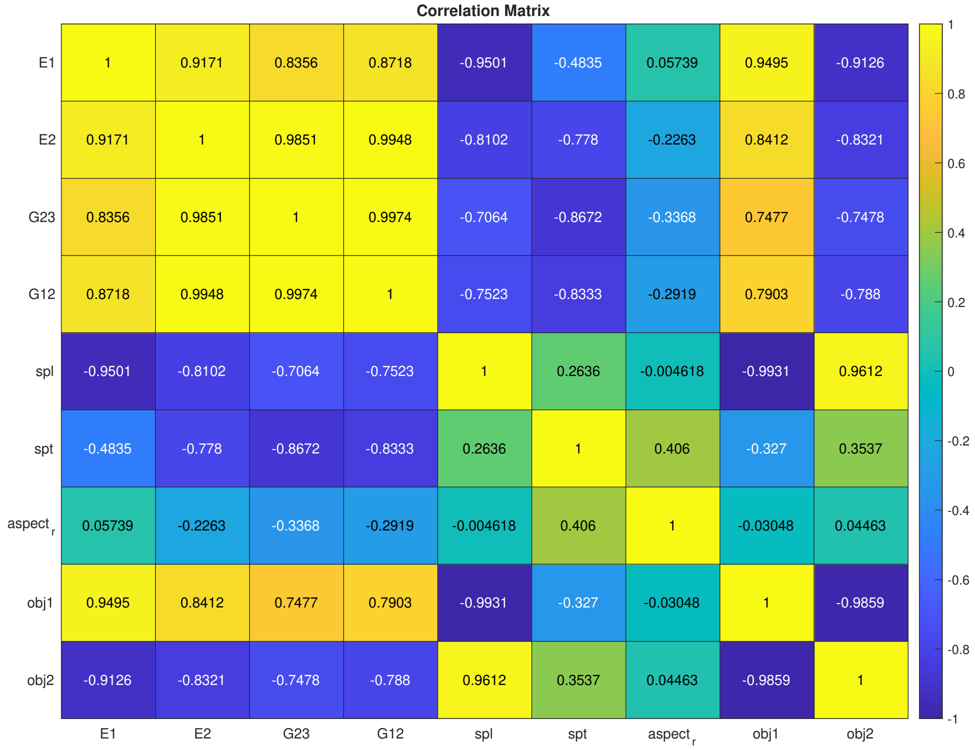


Heatmap of Top 5 Designs

Metrics	1	2	3	4	5	6	7	8	9
E1	96.02	90.57	81.18	89.78	97.5	81.75	84.6	84.61	88.31
E2	53.17	53.99	50.51	52.98	54.19	50.57	51.21	51.17	51.94
G23	16.12	16.52	15.65	16.22	16.42	15.65	15.77	15.76	15.91
G12	16.77	17.16	16.09	16.82	17.11	16.1	16.27	16.26	16.47
Spl	0.1513	0.7533	1.497	1.075	0.3115	1.479	1.109	1.134	0.7115
spt	1.25	1.117	1.249	1.145	1.166	1.249	1.249	1.25	1.249
aspectR	13.52	10.19	12.61	14.07	13.46	13.07	13.23	13.52	12.7
Obj1	-0.3632	-2.162	-5.607	-3.462	-0.7717	-5.513	-3.722	-3.838	-2.107
Obj2	-1.057	0.152	3.975	1.328	-0.7723	3.832	1.498	1.627	-0.0003

Figure 6.16: Heat Map for 9 Designs from Table 6.1

Young's modulus exhibits a strong correlation with other parameters, particularly sp_l and obj_1. Both objectives show a correlation above 0.95 with sp_l. This is because reducing the spacing (sp_l) makes the macrostructure resemble a stronger, heavier continuous fibre material, whereas increasing sp_l results in a weaker, lighter, and more discontinuous material.



Correlation Matrix

	E1	E2	G23	G12	spl	spt	aspect_r	obj1	obj2
E1	1	0.9171	0.8356	0.8718	-0.9501	-0.4835	0.05739	0.9495	-0.9126
E2	0.9171	1	0.9851	0.9948	-0.8102	-0.778	-0.2263	0.8412	-0.8321
G23	0.8356	0.9851	1	0.9974	-0.7064	-0.8672	-0.3368	0.7477	-0.7478
G12	0.8718	0.9948	0.9974	1	-0.7523	-0.8333	-0.2919	0.7903	-0.788
spl	-0.9501	-0.8102	-0.7064	-0.7523	1	0.2636	-0.004618	-0.9931	0.9612
spt	-0.4835	-0.778	-0.8672	-0.8333	0.2636	1	0.406	-0.327	0.3537
aspect_r	0.05739	-0.2263	-0.3368	-0.2919	-0.004618	0.406	1	-0.03048	0.04463
obj1	0.9495	0.8412	0.7477	0.7903	-0.9931	-0.327	-0.03048	1	-0.9859
obj2	-0.9126	-0.8321	-0.7478	-0.788	0.9612	0.3537	0.04463	-0.9859	1

Figure 6.18: Correlation Matrix Among Parameters of interest

Figures 6.19, 6.20, 6.21 and 6.22 shows Design 7, which yielded higher values of Young's modulus. The marked peaks indicates the highest Young's modulus achieved. Although this design is desirable due to its stiffness, its objective 1 value is less negative indicating that the material is somewhat heavier compared to other designs. A decision can be made by evaluating the trade-offs between objectives, using Figure 6.16 as a guiding tool.

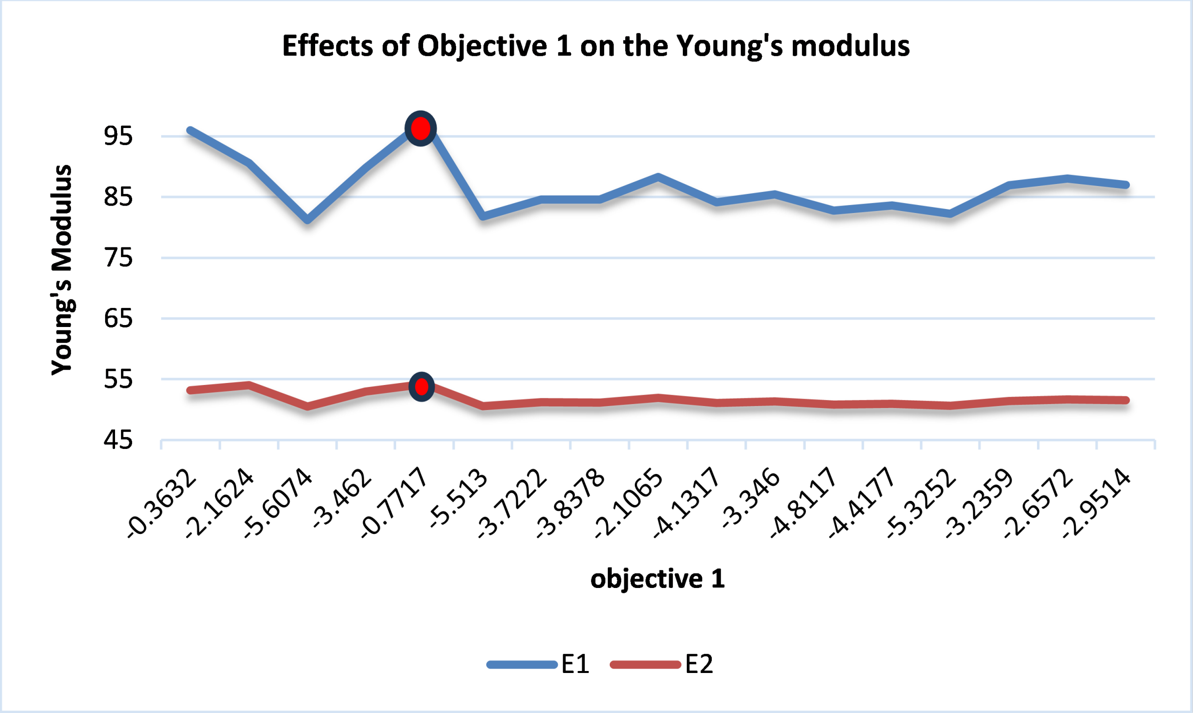


Figure 6.19: (a) Young's Modulus vs Objective 1 (material weight)

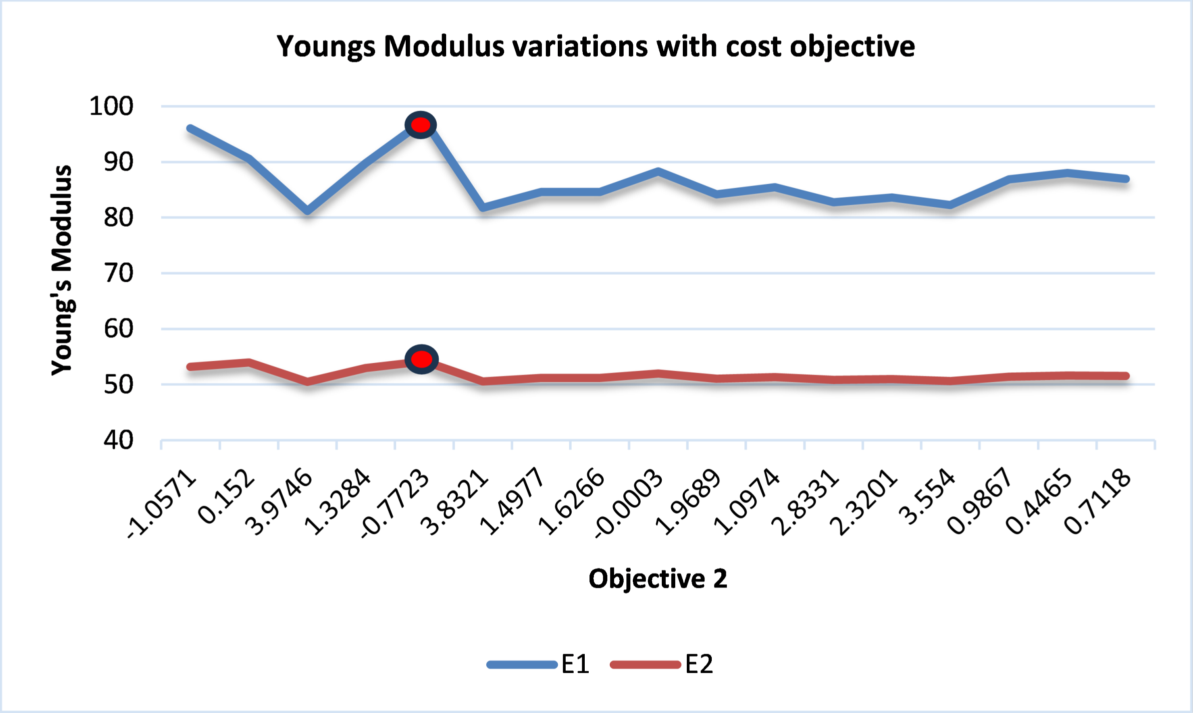


Figure 6.20: (b) Young's Modulus vs Objective 2 (material property)

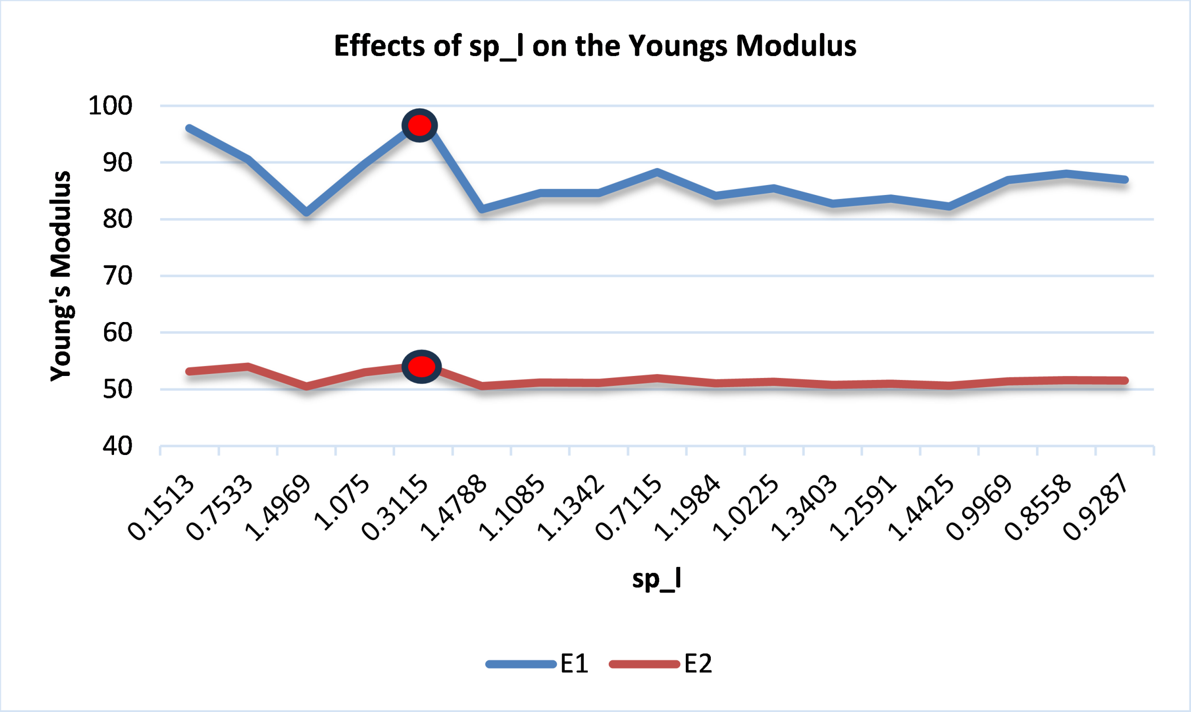


Figure 6.21: (c) Young's Modulus vs sp_l

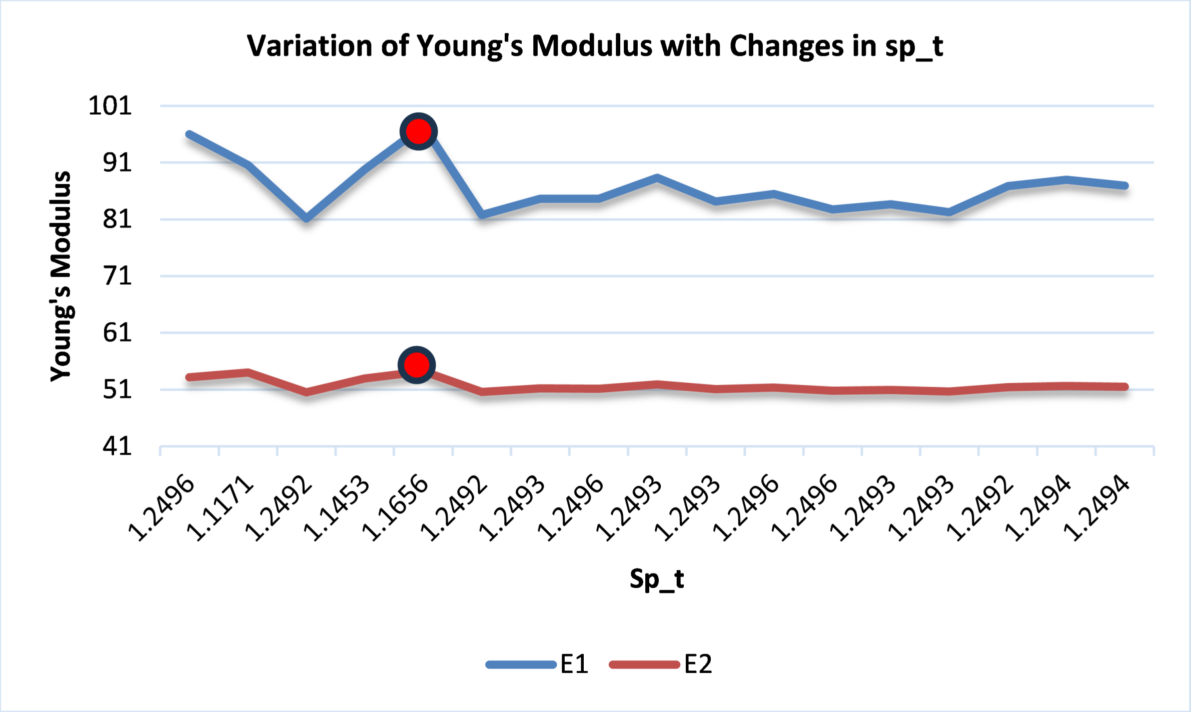


Figure 6.22: (d) Young's Modulus vs sp_t

The RVE microstructure for Design 7 is developed based on the aspect ratio and fibre spacing. This provides the size of the optimised structure, which can be used to build the new material. By translating these parameters into a representative volume element, the physical characteristics of the material can be accurately captured. This RVE can then be used for further simulations or experimental fabrication to validate performance.

The optimiser used here allows for the use of nonlinear constraints for example, a function such that the transverse modulus is kept at certain value say $E_0 = 110GPa$ is introduced and the optimiser is allowed to do best combinations of sp_l , sp_t and α that results in this transverse Young's modulus. This is done by setting the second objective to:

$$\boldsymbol{f_2} = S(2) - E_0 \tag{6.16}$$

Where, $S(2)$ is the transverse modulus obtained from the iteration process using the trained neural network. Throughout the process, the above equation remains zero, and the results in Table 6.2 represent the best designs obtained from the optimiser.

E1	E2	G_{23}	G_{12}	sp_l	sp_t	aspect r	obj1	obj2
147.0893	110.0000	33.2692	36.3303	0.6652	0.2110	13.3114	-1.2481	0.2237

Table 6.2: Results Table

Chapter 7

Conclusions

The goal of this thesis was to create a logical framework starting from modeling the composite material, homogenisation of the composite, testing, and optimisation of the microstructure for efficient design. Multi-scale analysis was used to analyse the solid when subjected to small strains to determine the elastic properties. The following provides a comprehensive account of the achieved milestones, conclusions drawn from the research, and outlines avenues for future work in this field.

Multi-scale Modelling: The described framework provided a link between micro and macro structures. Boundary conditions such as periodic and linear displacements were employed to represent the kinematic constraints within the Representative Volume Element (RVE) boundary. Linear boundary conditions were crucial for homogenisation to achieve convergence. This was verified through homogenisation examples where the RVE size was increased to obtain more accurate effective elastic properties of the composites.

Homogenisation: Dettmer's Mpap3 (Multi Physics Analysis Program) is utilised to investigate a large number of RVEs. The 3D meshes generated are generally non-periodic because unstructured meshes are used to predict material properties. Despite this, the interface allows periodic Representative Volume Elements (RVEs) to be analyzed with non-periodic meshes. Stress outputs are utilised to calculate effective elastic properties of the composite using the least squares method. These effective elastic properties are then compared to analytical approaches (Halpin Tsai and Hashin Rosen) to verify the method's accuracy. The least squares method achieves lower residuals, typically less than 10% for both longitudinal and transverse direction indicating high accuracy. Specifically, the method predicts longitudinal properties with less than 0.5% difference compared to analytical solutions, demonstrating excellent predictive capability.

Enhanced Homogenisation: A trained neural network, utilising data developed from the least squares method, is employed to predict the elastic properties of composite materials without the need for re-modelling the RVE, thus saving time and computational costs. This tool has proven to be powerful, delivering highly accurate results at a reduced cost. Moreover, a trained network can be reused multiple times, enhancing efficiency in subsequent analyses.

Optimisation: Integrating the trained neural network results with an optimiser an effectively tailored material designed to meet some criteria was developed. A material that gives the optimal balance between performance and resource utilisation in various applications is developed.

Although the procedure can be applied to numerous examples, more work is needed on meshing RVEs with irregular inclusions and RVEs with additional parameters, as most fibers have irregular shapes. Addressing these challenges would further improve the accuracy and applicability of the method. Future research should focus on refining mesh generation techniques and expanding the range of parameters considered to better represent the complexity of real composite materials. Future scope of this study entails modelling randomly oriented fibres and also considering inelastic material.

Future Research Direction: Although fibre-reinforced composites have been widely studied across various industries, several important research directions remain open and relevant for future work. In the area of multi-scale modelling and simulation, there is still a need for an integrated framework that effectively couples micro-scale fibre-matrix interactions with macro-scale structural performance. Additionally, the use of machine learning to accelerate the prediction of stiffness, strength, and damage evaluation is an emerging field that offers significant potential and remains largely unexplored.

Hybrid and multifunctional composites are also areas of growing interest. For example, hybrid composites offer opportunities to balance performance, cost, and weight more effectively. Further research can be directed toward multifunctional composites with enhanced capabilities such as energy storage, thermal management, and electrical conductivity, enabling the development of superior, next-generation materials. These composites can be tailored for specific applications, including insulation, impact resistance, and improved fatigue durability.

Quantification of uncertainty and reliability in materials science also presents an interesting avenue for study. The link between uncertainty in materials and their performance under service conditions can be further explored to help bridge the gap between laboratory results and industrial adoption.

Structural health monitoring techniques offer another avenue for advancing non-destructive evaluation methods for early damage detection. Sensors can be incorporated into composites as self-sensing fibres for in-situ monitoring. This approach enables the early detection of damage, thereby reducing the risk of injuries or structural failure. This field remains a promising area for further research.

Manufacturing and optimisation using 3D printing is also an emerging research area that can enable the production of customised geometries while reducing material waste. This field can be explored for fibre placement and resin infusion processes to improve repeatability and scalability. It offers significant potential for further research and remains one of the most promising areas in advanced manufacturing.

These emerging areas ranging from uncertainty quantification to advanced manufacturing highlight the need for continued research in materials science. Exploring these technologies further can lead to safer, more efficient, and scalable solutions for real-world applications

Appendix A

GEO and Mpap3 Output Files

A.1 Example of a Gmsh Geometry File

A geometry file generated using the c++ code. Together with `Gmash` software a mesh file is generated using this file.

```
Mesh.Binary=0;
SetFactory("OpenCASCADE");
Point(1)  = {0, 0, 0, 0.3 };
Point(2)  = {0, 1.05, 0, 0.3 };
Point(3)  = {0, 1.05, 1.05, 0.3 };
Point(4)  = {0, 0, 1.05, 0.3 };
Point(5)  = {1.05, 0, 1.05, 0.3 };
Point(6)  = {1.05, 1.05, 1.05, 0.3 };
Point(7)  = {1.05, 1.05, 0, 0.3 };
Point(8)  = {1.05, 0, 0, 0.3 };
Line(1)  = { 1, 2 };
Line(2)  = { 2, 3 };
Line(3)  = { 3, 4 };
Line(4)  = { 4, 1 };
Line(5)  = { 4, 5 };
Line(6)  = { 5, 6 };
Line(7)  = { 6, 7 };
Line(8)  = { 7, 8 };
Line(9)  = { 8, 5 };
Line(10)  = { 8, 1 };
Line(11)  = { 3, 6 };
Line(12)  = { 2, 7 };
Sphere(1) = {0.525, 0.525, 0.525, 0.145};
Curve Loop(2) = {1, 2, 3, 4};
Plane Surface(2) = { 2 };
Curve Loop(3) = {6, 7, 8, 9};
Plane Surface(3) = { 3 };
Curve Loop(4) = {3, 11, 5, 6};
Curve Loop(5) = {2, 12, 7, 11};
Curve Loop(6) = {4, 10, 5, 9};
Curve Loop(7) = {8, 10, 1, 12};
Plane Surface(8) = {4};
Plane Surface(9) = {5};
Plane Surface(10) = {6};
Plane Surface(11) = {7};
Surface Loop(12) = { 2, 3, 8 , 9, 10 , 11};
Surface Loop(13) = { 1};
Volume(2) = {12, 13};
Physical Volume("matrix") = {2};
Physical Volume("inclusions") = { 1 };
Mesh.CharacteristicLengthFromCurvature = 8;
 Mesh 3;
Mesh.Smoothing = 100;
```

Listing A.1: Cube with spherical inclusion of radius of 0.145 Geometry File

A.2 Example of FEM Output File

This file contains the stress outputs from Mpap3 software. Five test were carried out for each RVE.

```

+------------------------------+
|                              |
|          M P A P 3           |
|                              |
|    Wulf G. Dettmer (2016)    |
|                              |
+------------------------------+
| project: solid               |
+------------------------------+
| test RVE                     |
+------------------------------+

 linear b.c.

1 -> inclusions
2 -> matrix

 initialising solver for 33507 degrees of freedom ...
 creating row compressed format (unsymmetric)...
 deleting temporary data ...

 creating row compressed format (unsymmetric)...
 deleting temporary data ...

 allocating memory for matrix coefficients ...

 0. norm(residual) = 6.34218753e+03
 1. norm(residual) = 2.01269032e-11

{    0.2,     0.1,    -0.1,     0.2,     0.1,    -0.1}_6

{108.869, 84.1503, 60.8713, 21.8736, 10.0661, -10.9686}_6

 0. norm(residual) = 5.80211257e+03
 1. norm(residual) = 2.99701479e-11

{   0.03,    0.02,  -0.004,     0.2,     0.3,    -0.5}_6

{22.0758,  18.885, 16.0817, 21.8995, 30.2763, -54.7639}_6

 0. norm(residual) = 6.91051732e+03
 1. norm(residual) = 3.14948737e-11

{   0.01,    0.03,    -0.2,   -0.05,     0.2,    -0.1}_6

{-56.2017, -53.0825, -79.833, -5.47966, 20.1998, -10.9447}_6

 0. norm(residual) = 3.51719606e+03
 1. norm(residual) = 1.14377528e-11

{   0.01,    0.03,   -0.04,     0.1,    0.01,   -0.01}_6

{1.81083, 3.59946, -4.55514, 10.9367, 1.00439, -1.1029}_6

 0. norm(residual) = 6.74420067e+03
 1. norm(residual) = 2.19056688e-11

{    0.3,     0.05,    -0.01,      0.2,     0.04,   -0.005}_6

{ 177.83, 128.729, 121.745, 21.8656, 3.99714, -0.568685}_6

 vtkUnstructuredGrid:
 number of nodes = 12387
 number of cells = 8840

 file written: 'rve.vtu'
```

Listing A.2: Example of FEM output File

Appendix B

MATLAB Codes

B.1 Code 1: Isotropic Homogenisation

A transversely isotropic materiaL MATLAB code for least square method used for homogenisation.

```matlab
function [E,nu,r] = isotropic(strain,stress)

% exp is 6 by 2 by N array:
%  6 spatial coefficients
%  strains (1) and stresses (2)
%  N experiements

N = size(strain,1);

A = [];
rhs = [];

for i=1:N
    eps = strain(i,:)';
    eps(4:6) = 2*eps(4:6);

    sig = stress(i,:)';

    %
    %     S11         S12
   a = [sig(1)   (sig(2)+sig(3));    % eps11
        sig(2)   (sig(3)+sig(1));    % eps22
        sig(3)   (sig(1)+sig(2));    % eps33
      2*sig(4)    -2*sig(4)  ;    % 2 x eps12
      2*sig(5)    -2*sig(5)  ;    % 2 x eps23
      2*sig(6)    -2*sig(6)   ];  % 2 x eps31

    A = [A;a];

    rhs = [rhs;eps];
end

size(A);
size(rhs);

[S,flag,r] = lsqr(A,rhs);

size(S);

E = 1/S(1);
nu = -S(2)*E;

end
```

Listing B.1: Isotropic Data Analysis Script

B.2 Code 2: Transversely Isotropic Homogenisation

The code provides the least square method used for homogenisation of an isotropic material.

```
function [E1,E2,nu12,nu23,G12,G23,r] = transverseisotropic(strain,stress)
% exp is 6 by 2 by N array:
%  6 spatial coefficients
%  strains (1) and stresses (2)
%  N experiements

N =size(strain,1);

A = [];
rhs = [];

for i=1:N
    eps = strain(i,:)';
    eps(4:6) = 2*eps(4:6);

    sig = stress(i,:)';
    %sig(3,:)=0;

    %
    %     S11      S22      S12        S23     S55
    a = [sig(1)     0   sig(2)+sig(3)     0      0   ;    % eps11
               0    sig(2)    sig(1)     sig(3)    0   ;    % eps22
               0    sig(3)    sig(1)     sig(2)    0   ;    % eps33
               0  2*sig(4)      0     -2*sig(4)    0   ;    % 2 x eps12
               0       0        0           0    sig(5);    % 2 x eps23
               0       0        0           0    sig(6) ];  % 2 x eps31

    A = [A;a];

    rhs = [rhs;eps];
end

[S,flag,r] = lsqr(A,rhs);

E1   = 1/S(1);
E2   = 1/S(2);
nu12 = -S(3)*E1;
nu23 = -S(4)*E2;
G12  = 1/S(5);
G23  = 1/(2*(S(2)-S(4)));
end
```

Listing B.2: Transverse isotropic Data Analysis Script

B.3 Code 3: FEM Output File Stress/Strain Extraction

This MATLAB code is used to read Appendix A.2 and is used to concentrate all stress values into a larger matrix for all RVE output files for homogenisation.

```

clear all
clc
% Initialize matrices to store the numbers
matrix_line31 = [];
matrix_line33 = [];
matrix_line38 = [];
```

```
matrix_line40 = [];
matrix_line45 = [];
matrix_line47 = [];
matrix_line52 = [];
matrix_line54 = [];
matrix_line59 = [];
matrix_line61 = [];


num_exp=6;

%%============================================================================
%        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
%        +        EXTRACTION OF STRESSES AND STRAINS FROM FEM        +
%        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
%=============================================================================

% Loop through the files
for i = 1:num_exp
    disp([’Processing file ’ num2str(i)]);
    % Read the lines from the file
    lines = textread([’rve’ num2str(i) ], ’%s’, ’delimiter’, ’\n’);
    %lines = textread(’rve112’, ’%s’, ’delimiter’, ’\n’);

    % Extract numbers from line 31 and clean up the string
    line31_str = lines{31};
    line31_str = strrep(line31_str, ’{’, ’’);
    line31_str = strrep(line31_str, ’}’, ’’);
    line31_str = strrep(line31_str, ’_6’, ’’);

    % Convert the cleaned string to numbers
    line31_numbers = str2num(line31_str);

    % Append the extracted numbers to the matrix
    matrix_line31 = [matrix_line31; line31_numbers];

    % Extract numbers from line 33 and clean up the string
    line33_str = lines{33};
    line33_str = strrep(line33_str, ’{’, ’’);
    line33_str = strrep(line33_str, ’}’, ’’);
    line33_str = strrep(line33_str, ’_6’, ’’);

    % Convert the cleaned string to numbers
    line33_numbers = str2num(line33_str);

    % Append the extracted numbers to the matrix
    matrix_line33 = [matrix_line33; line33_numbers];


    % Extract numbers from line 38 and clean up the string
    line38_str = lines{38};
    line38_str = strrep(line38_str, ’{’, ’’);
    line38_str = strrep(line38_str, ’}’, ’’);
    line38_str = strrep(line38_str, ’_6’, ’’);

    % Convert the cleaned string to numbers
    line38_numbers = str2num(line38_str);

    % Append the extracted numbers to the matrix
    matrix_line38 = [matrix_line38; line38_numbers];

     % Extract numbers from line 40 and clean up the string
    line40_str = lines{40};
    line40_str = strrep(line40_str, '{', '');
    line40_str = strrep(line40_str, '}', '');
    line40_str = strrep(line40_str, '_6', '');

    % Convert the cleaned string to numbers
    line40_numbers = str2num(line40_str);

    % Append the extracted numbers to the matrix
    matrix_line40 = [matrix_line40; line40_numbers];

    % Extract numbers from line 45 and clean up the string
    line45_str = lines{45};
    line45_str = strrep(line45_str, '{', '');
    line45_str = strrep(line45_str, '}', '');
    line45_str = strrep(line45_str, '_6', '');

    % Convert the cleaned string to numbers
    line45_numbers = str2num(line45_str);

    % Append the extracted numbers to the matrix
    matrix_line45 = [matrix_line45; line45_numbers];


    % Extract numbers from line 47 and clean up the string
    line47_str = lines{47};
    line47_str = strrep(line47_str, '{', '');
    line47_str = strrep(line47_str, '}', '');
    line47_str = strrep(line47_str, '_6', '');

    % Convert the cleaned string to numbers
    line47_numbers = str2num(line47_str);

    % Append the extracted numbers to the matrix
    matrix_line47 = [matrix_line47; line47_numbers];


    % Extract numbers from line 52 and clean up the string
    line52_str = lines{52};
    line52_str = strrep(line52_str, '{', '');
    line52_str = strrep(line52_str, '}', '');
    line52_str = strrep(line52_str, '_6', '');

    % Convert the cleaned string to numbers
    line52_numbers = str2num(line52_str);

    % Append the extracted numbers to the matrix
    matrix_line52 = [matrix_line52; line52_numbers];


     % Extract numbers from line 54 and clean up the string
    line54_str = lines{54};
    line54_str = strrep(line54_str, '{', '');
    line54_str = strrep(line54_str, '}', '');
    line54_str = strrep(line54_str, '_6', '');

    % Convert the cleaned string to numbers
    line54_numbers = str2num(line54_str);

    % Append the extracted numbers to the matrix
    matrix_line54 = [matrix_line54; line54_numbers];


    % Extract numbers from line 59 and clean up the string
    line59_str = lines{59};
    line59_str = strrep(line59_str, '{', '');
    line59_str = strrep(line59_str, '}', '');
    line59_str = strrep(line59_str, '_6', '');

    % Convert the cleaned string to numbers
    line59_numbers = str2num(line59_str);

    % Append the extracted numbers to the matrix
    matrix_line59 = [matrix_line59; line59_numbers];

 % Extract numbers from line 61 and clean up the string
    line61_str = lines{61};
    line61_str = strrep(line61_str, '{', '');
    line61_str = strrep(line61_str, '}', '');
    line61_str = strrep(line61_str, '_6', '');

    % Convert the cleaned string to numbers
    line61_numbers = str2num(line61_str);

    % Append the extracted numbers to the matrix
    matrix_line61 = [matrix_line61; line61_numbers];


%%============================================================================
%                      +++++++++++++++++++++++++++++++++++++
%                      +   STRESS AND STRAIN ASSEMBLY      +
%                      +++++++++++++++++++++++++++++++++++++
%=============================================================================


    strain = [matrix_line31(i,:);matrix_line38(i,:);matrix_line45(i,:);matrix_line52(i,:);matrix_line59(i,:)];
    stress = [matrix_line33(i,:);matrix_line40(i,:);matrix_line47(i,:);matrix_line54(i,:);matrix_line61(i,:)]./3;

%%============================================================================
%                       +++++++++++++++++++++++++
%                       +   HOMOGENASATION      +
%                       +++++++++++++++++++++++++
%=============================================================================
%Isotropic Use

for j=1:5
    [E(j),nu(j),r(j)] = isotropic(strain(j,:),stress(j,:));
end

E_nu_r = [E',nu',r']

% two experiments: together

[E,nu,r] = isotropic(strain,stress);

eval(['E' num2str(i) ' =  E']);
eval(['nu' num2str(i) ' = nu']);
eval(['r' num2str(i) ' =  r']);

eval(['K' num2str(i) ' = E/(3*(1-2*nu))']);
eval(['G' num2str(i) ' = E/(2*(1+nu))']);


%For Transverse isotropic Use

```

```
for j=1:5

    [E1(j),E2(j),nu12(j),nu23(j),G12(j),G23(j),r(j)] =transverseisotropic(strain(j,:),stress(j,:));%transverseIsotropy(strain(i,:),stress(i,:));
end

[E1',E2',nu12',nu23',G12',G23',r'];

% seven experiments: together

[E1,E2,nu12,nu23,G12,G23,r] = transverseisotropic(strain,stress);

eval(['Ex' num2str(i) ' =  E1']);
eval(['Ey' num2str(i) ' =  E2']);
eval(['nuxy' num2str(i) ' = nu12']);
eval(['nuyz' num2str(i) ' = nu23']);

eval(['Gxy' num2str(i) ' = G12']);
eval(['Gyz' num2str(i) ' = G23']);
 eval(['r' num2str(i) ' =  r']);


end

%%=============================================================================
%                             +++++++++++++++++++++
%                             +         END       +
%                             +++++++++++++++++++++
%==============================================================================
```

Listing B.3: FEM output file stress/strain Extraction script

B.4 Code 4: Isotropic Assembly Script For Elastic Properties

Together with Appendix B.3 and B.1 this MATLAB code concentrated all the elastic parameters of an isotropic material to their respective matrix for further analysis.

```

clear all
clc
run('Extraction.m');

%%=========================================================================
%++++++++++++++++++
% EXTRACTION OF K  +
%++++++++++++++++++

n = num_exp;
combined_vector = [];
for i = 1:n
    var_name = ['K' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector = [combined_vector; eval([var_name '(:)'])];
    else
        warning(['Variable ' var_name ' does not exist. Skipping...']);
    end
end

%%=========================================================================
%++++++++++++++++++
% EXTRACTION OF G  +
%++++++++++++++++++

combined_vector1 = [];
```

```
for i = 1:n
    var_name = ['G' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector1 = [combined_vector1; eval([var_name '(:)'])];
    else
        warning(['Variable1 ' var_name ' does not exist. Skipping...']);
    end
end

%%========================================================================
%+++++++++++++++++++
% EXTRACTION OF E  +
%+++++++++++++++++++

combined_vector2 = [];
for i = 1:n
    var_name = ['E' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector2 = [combined_vector2; eval([var_name '(:)'])];
    else
        warning(['Variable2 ' var_name ' does not exist. Skipping...']);
    end
end
%%========================================================================
%+++++++++++++++++++
% EXTRACTION OF V  +
%+++++++++++++++++++

combined_vector3 = [];
for i = 1:n
    var_name = ['nu' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector3 = [combined_vector3; eval([var_name '(:)'])];
    else
        warning(['Variable3 ' var_name ' does not exist. Skipping...']);
    end
end

%%========================================================================
%+++++++++++++++++++++++++
% EXTRACTION OF RESIDUAL +
%+++++++++++++++++++++++++

combined_vector4 = [];
for i = 1:n
    var_name = ['r' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector4 = [combined_vector4; eval([var_name '(:)'])];
    else
        warning(['Variable4 ' var_name ' does not exist. Skipping...']);
    end
end

%%========================================================================
%+++++++++++++++++++++++++++++++++++++++
% DISPLAY ALL THE EXTRACTED PROPERTIES +
%+++++++++++++++++++++++++++++++++++++++

G =[combined_vector1]
K = [combined_vector]
E =[combined_vector2]
nu = [combined_vector3]
r = [combined_vector4]
```

```
%%=====================================================================
%               +++++++++++++++++++++
%               +        END        +
%               +++++++++++++++++++++
```

Listing B.4: Isotropic Assembly code

B.5 Code 5: Transverse Isotropic Assembly Script For Elastic Properties

Together with Appendix B.3 and B.2 this MATLAB code assemble all the elastic parameters of a transversely isotropic material to their respective matrix for further analysis.

```

clear all
clc
run('EXTRACTIONTRANS.m');
%%=====================================================================
%+++++++++++++++++++
% EXTRACTION OF E1 +
%+++++++++++++++++++

n = num_exp;
combined_vector = [];
for i = 1:n
    var_name = ['Ex' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector = [combined_vector; eval([var_name '(:)'])];
    else
        warning(['Variable ' var_name ' does not exist. Skipping...']);
    end
end

%%=====================================================================
%+++++++++++++++++++
% EXTRACTION OF E2 +
%+++++++++++++++++++


combined_vector1 = [];
for i = 1:n
    var_name = ['Ey' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector1 = [combined_vector1; eval([var_name '(:)'])];
    else
        warning(['Variable1 ' var_name ' does not exist. Skipping...']);
    end
end
%%=====================================================================
%+++++++++++++++++++++
% EXTRACTION OF V12 +
%+++++++++++++++++++++

combined_vector2 = [];
for i = 1:n
    var_name = ['nuxy' num2str(i)];
    if exist(var_name, 'var') == 1
        combined_vector2 = [combined_vector2; eval([var_name '(:)'])];
    else
        warning(['Variable2 ' var_name ' does not exist. Skipping...']);
    end
end

```

```
%%========================================================================
%+++++++++++++++++++++
% EXTRACTION OF V23 +
%+++++++++++++++++++++

combined_vector3 = [];
for i = 1:n
    var_name = [’nuyz’ num2str(i)];
    if exist(var_name, ’var’) == 1
        combined_vector3 = [combined_vector3; eval([var_name ’(:)’])];
    else
        warning([’Variable3 ’ var_name ’ does not exist. Skipping...’]);
    end
end
%%========================================================================
%+++++++++++++++++++++++++++
% EXTRACTION OF RESIDUAL r +
%+++++++++++++++++++++++++++

combined_vector4 = [];
for i = 1:n
    var_name = [’r’ num2str(i)];
    if exist(var_name, ’var’) == 1
        combined_vector4 = [combined_vector4; eval([var_name ’(:)’])];
    else
        warning([’Variable4 ’ var_name ’ does not exist. Skipping...’]);
    end
end

%%========================================================================
%++++++++++++++++++++++++++++++++++
% EXTRACTION OF SHEAR MODULUS G12 +
%++++++++++++++++++++++++++++++++++

combined_vector5 = [];
for i = 1:n
    var_name = [’Gxy’ num2str(i)];
    if exist(var_name, ’var’) == 1
        combined_vector5 = [combined_vector5; eval([var_name ’(:)’])];
    else
        warning([’Variable5 ’ var_name ’ does not exist. Skipping...’]);
    end
end

%%========================================================================
%++++++++++++++++++++++++++++++++++
% EXTRACTION OF SHEAR MODULUS G23 +
%++++++++++++++++++++++++++++++++++

combined_vector6 = [];
for i = 1:n
    var_name = [’Gyz’ num2str(i)];
    if exist(var_name, ’var’) == 1
        combined_vector6 = [combined_vector6; eval([var_name ’(:)’])];
    else
        warning([’Variable6 ’ var_name ’ does not exist. Skipping...’]);
    end
end

%%========================================================================
%+++++++++++++++++++++++++++++++++++++++
% DISPLAY ALL THE EXTRACTED PROPERTIES +
%+++++++++++++++++++++++++++++++++++++++

E1   =  [combined_vector]
E2   = [combined_vector1]
nu12 = [combined_vector2]
nu23 = [combined_vector3]
G12  = [combined_vector5]
G23  = [combined_vector6]
r    = [combined_vector4]
%%=====================================================================
%                ++++++++++++++++++++++
%                +         END         +
%                ++++++++++++++++++++++
```

Listing B.5: Transverse Isotropic Assembly script

B.6 Code 6: Neural Network Training

This code provide a neural network with three hidden layers two of which use `transig` activation fuction and one layer that uses the `logsig` activation function. This network is used for training.

```

run('BRICKrslt.m')
K=[A(:,1),A(:,2),A(:,3),A(:,4)];
G=[A(:,8),A(:,9)];

target= [K]';
input= [G]';

% Create a feedforward network with 10 neurons in one hidden layer
net304 = feedforwardnet([5,15,5]);

% Set the transfer function for the hidden layers
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'logsig';
net.layers{3}.transferFcn = 'tansig';
%net.layers{4}.transferFcn = 'logsig';
%net.trainParam.max_fail = 6; % Maximum validation failures
%net.performParam.regularization = 0.1; % L2 regularization parameter

% Configure the network
net = configure(net, input, target);

% Train the network
[net, tr] = train(net, input, target);

% Display the trained network object
disp('Trained Network:');
disp(net);

% Display the training record
disp('Training Record:');
disp(tr);

filename = 'my_neural_network_model.mat';
save(filename, 'net');
out_put=net([input])'
%%
figure;
scatter(A(:,1),A(:,3),'r+')
hold on

scatter(A(:,2),A(:,4),'g+')
hold on

```

```
scatter(out_put(:,1),out_put(:,3),'b*')
hold on
scatter(out_put(:,2),out_put(:,4),'k*')

grid on; % Turn on the grid

% Customize grid appearance
grid(gca, 'minor');
ax = gca;

% Customize minor grid lines
ax.MinorGridColor = [0.5, 0.5, 0.5]; % Minor grid lines color
ax.MinorGridLineStyle = ':'; % Minor grid lines style

% Customize major grid lines
set(gca, 'XMinorGrid', 'on'); % Turn on minor grid lines for x-axis
set(gca, 'YMinorGrid', 'on'); % Turn on minor grid lines for y-axis

% Customize grid lines further (optional)
ax.GridAlpha = 0.8; % Grid transparency (0 to 1)
ax.GridColor = [0.2, 0.2, 0.2]; % Grid color
ax.GridLineStyle = ':'; % Grid line style
xlim([31,133])
ylim([10,30])
legend('actual longitudinal data','actual transverse data','trained longitudinal data','trained transverse data')%, 'linear analysis');
title('Neural plot(G,E) vs Actual plot(G,E)');
ylabel('G_{12},G_{13}/ Shear Modulus');
xlabel('E1,E2/ Youngs  modulus');

% Adjust plot appearance (optional)
set(gca, 'FontSize', 12); % Font size for axes labels and ticks
set(gca, 'LineWidth', 0.2); % Line width for axes
set(gcf, 'Color', 'w'); % Figure background color

hold off
```

Listing B.6: Neural Network Training script

B.7 Code 7: Cost Function 1 script

This script file gives objectives in Section 6.3 as a function of its decision variables.

```
function val = cost(x,ann)

ri = x(1);
rv = x(2);

d = 4.2; side of the cube/RVE

Cm = 1;
Ci = 1000;

a = 1;
b= 100;

%        matrix    inclusion
%   K      100        1000
%   mu      10           100

% minimum stifness required (constraint)
kmn  = 95;
mumn = 9.5;

```

```
% For 3D
V2 = d^3;
n  = 4;

Vi = 0.5 * (n * n * n * 4/3 * pi * ri^3) / V2;
Vv = 0.5 * (n * n * n * 4/3 * pi * rv^3) / V2;

Vm = 1. - Vi - Vv;

s = ann([ri;rv]);

k = s(1);
mu = s(2);

C1 = (Vm * Cm + Vi * Ci)*b;
C2 = a * ((max(0,k-kmn)/k  + max(0,mu-mumn)/mu ));

val(1) = C1;
val(2) = -C2;

end
```

Listing B.7: Cost Function 1

B.8 Code 8: Cost Function 2

This script file gives objectives in Section 6.4 as a function of its decision variables.

```
function val = opt(x,ann)

%======================================================

%======================================================


ri = x(1);
rv = x(2);
rj = x(3);

Cm = 1;
Ci = 1000;

a = 1000;
b= 0.1;%1.0e+10;

%         matrix    inclusion
%    K       100         1000
%    mu       10           100

% minimum stifness required (constraint)
E1mn   = 70.71;
E2mn =   49.62;
G12mn = 15.63;
G23mn = 16.08;
fvmn   = 0.5;

s = ann([ri;rv;rj]);

E1   = s(1);
E2   = s(2);
G12  = s(3);
G23  = s(4);

rd   = 1;
```

```
l   = rj*rd;
t   = ri*rd;
lng = rv*rd;


 Fvr = (l*pi*(rd/2)^2)/((t+rd)*(l+lng));

 Mvr = 1-Fvr;


 C1 = Cm*Mvr + Ci*Fvr;

  C2 = a * ((max(0,E1-E1mn)/E1mn  + max(0,E2-E2mn)/E2mn + max(0,G12-G12mn)/G12mn ...
        + max(0,G23-G23mn)/G23mn));

 val(1) = C1;
 val(2) = -C2;


end
```

Listing B.8: Cost Function 2 script

B.9 Code 9: `gamult` optimiser code

This provide the code used for optimisation in Section 6.3.

```


% Load the saved results
load('my_neural_network_model.mat');

%%
% Define the number of decision variables
nvars = 2;

% Define lower and upper bounds for the decision variables
lb = [0.00, 0.00]; % lower bounds
ub = [0.5, 0.5]; % upper bounds

% Set the options for the genetic algorithm

   options = optimoptions('gamultiobj', 'SelectionFcn', {@selectiontournament, 50},'ConstraintTolerance', 1e-6000,...
        'CrossoverFraction',0.75, 'PopulationSize', 100, 'MaxGenerations', 100,'PlotFcn',{@gaplotparetodistance});


% Run the multi-objective genetic algorithm
[b, fval, exitflag, output] = gamultiobj(@(x)cost(x,mlnet), nvars,[], [], [], [], lb, ub, [], options);

%%

% Display the optimized solution and objective function values
disp('Optimized void & incl radii Solution:');
comb_of_ri_rv = b
disp('Objective Function Values [C1 and C2 ]:');
disp(fval);

% Calculation of the cost and design parameters K and mu with optimized ri and rv
disp('design parameters;')
```

```
K_and_mu_best = mlnet(b')'

figure(4)
plot(b(:, 2), b(:, 1), 'k*');
xlabel('voids radius');
ylabel('inclusion radius');
title('Decision Variable Values');
% Set the limits for the x-axis and y-axis
xlim([0 0.5]);
ylim([0 0.5]);
grid on


return
```

Listing B.9: Optimiser script

Appendix C

C++ Codes

C.1 Code 1: Main FEM Program

This provide the code used to Rum the Mpap3 with five different strains tests.

```
#define VTK
#define SOLID
#define MESH_TOOLS
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"

int main(int argc ,char**args)
{
        mpapInit("solid", "test RVE");
        mos.setDoubleFormat("%7g");

        FemSolidRVE rve("rve", 3);
        //rve.getMeshFromGmsh41File("rve2ex104-mod.msh", "solid3D");
        rve.getMeshFromGmsh41File(args[1], "solid3D");

        rve.mesh3D4to10noded("solid3D"); //convert 4 noded element to 10 noded

       rve.centreMesh();
        rve.x0 = rve.x;

        //Choose boundary condition
       rve.set("linearBC");
        mos << " linear b.c.\n\n";
        // rve.set("periodicBC"); mos << " periodic b.c.\n\n";

        rve.set("ngp", 5); //set Number of gauss point.
        rve.set("tol", 1.e-5); //set Tolerance.
        rve.set("maxIter", 5); // set number of iterations.

        rve.set("Hooke");
        rve.set("updateCoorFlag", false);
        rve.set("3D");

        rve.listGroups();

          //Set Material Properties

        /*rve.set("K", 100.);
        rve.set("mu", 10.);*/

        rve.setGroupData("matrix", "K", 100.);
        rve.setGroupData("matrix", "mu", 10.);
        rve.setGroupData("inclusions", "K", 1000.);
        rve.setGroupData("inclusions", "mu", 100.);

        //set strains for the experiment

        double eps[] = { 0.2,          0.1,         -0.1,          0.2,
                    0.1,         -0.1}, sig[10], cc[36],
                 F[] = {1.1, 0.1, 0.2, -0.1, 0.9, 0., 0.2, 0.1, 1.05};
        rve.stressUpdate(sig, cc, eps);
        mos << mos.array(eps, 6) << "\n\n";
        mos << mos.array(sig, 6) << "\n\n";

        double eps1[] = { 0.03,          0.02,         -0.004,          0.2,
                    0.3,         -0.5}, sig1[10];
        rve.stressUpdate(sig1, cc, eps1);
        mos << mos.array(eps1, 6) << "\n\n";
        mos << mos.array(sig1, 6) << "\n\n";

        double eps2[] = { 0.01,          0.03,         -0.2,          -0.05,
                    0.2,         -0.1}, sig2[10];
        rve.stressUpdate(sig2, cc, eps2);
        mos << mos.array(eps2, 6) << "\n\n";
        mos << mos.array(sig2, 6) << "\n\n";

        double eps3[] = { 0.01,          0.03,         -0.04,          0.1,
                    0.01,         -0.01}, sig3[10];
        rve.stressUpdate(sig3, cc, eps3);
        mos << mos.array(eps3, 6) << "\n\n";
        mos << mos.array(sig3, 6) << "\n\n";

        double eps4[] = { 0.3,          0.05,         -0.01,          0.2,
                    0.04,         -0.005}, sig4[10];
        rve.stressUpdate(sig4, cc, eps4);
        mos << mos.array(eps4, 6) << "\n\n";
        mos << mos.array(sig4, 6) << "\n\n";

 //vidualise the last strains deformations
#ifdef VTK
        rve.vtkGetUndeformedMesh();

        rve.vtkGetCellData("group");

       rve.vtkGetVectorData("displacement", 0);

        rve.vtkWriteOutputFile("rve.vtu");

#endif

}
```

Listing C.1: Program to run the Meshes

C.2 Code 2: Cube Geometry file Generator

Homogenous blocks experimented on in this study are generated using this code as a geometry generator.

```
#define VTK
#define SOLID
#define MESH_TOOLS
```

```
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"
#include <sstream>

int main()
{
  int num = 1 ,num2=1;
  std::ofstream output_file("rve2ex10423sphere_Inclusions_with_voids2.m");
  if (output_file.is_open())
   double rdv, rd;
  {
    for (double n = 1; n <= 1; n += 1)
    {
       double rd,rdv;
         rd = 0.0; // radius of inclusions
         rdv =0.0;//radius of voids
      {
        std::string filename = "output_" + std::to_string(rd) + "_" + std::to_string(n) + ".geo";
        std::ofstream file(filename);
          if (file.is_open())
        {
          std::cout << "*success..........................................*\n\n";
        }
        int j, i, r, rr, Rr, v,  S, fibre, swww1, swww;
        double d, nx, ny, nz, lc, t, a, b, z, x, y, y2, Pi, tol = 1.e-4;
        lc =0.1; //mesh size
        Pi = 3.14159265358979323846;
        double Rr1 = 5, r1 = 4, r2 = 6, Rr2 = 7, r3 = 8, Rr3 = 9;
        r = 1;
        Rr = 3;
        file << "Mesh.Binary=0;" << std::endl;
        file << "SetFactory(\"OpenCASCADE\");" << std::endl;
        double  maxr=1;          ;
        ny=maxr;
        nx = ny;
        nz = ny;
        nz = ny;
file << "Point(1)  = {" << 0.0 << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(2)  = {" << 0 << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(3)  = {" << 0 << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(4)  = {" << 0 << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(5)  = {" << nx << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(6)  = {" << nx << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(7)  = {" << nx << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(8)  = {" << nx << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Line(1)  = { 1, 2 };" << std::endl;
        file << "Line(2)  = { 2, 3 };" << std::endl;
        file << "Line(3)  = { 3, 4 };" << std::endl;
        file << "Line(4)  = { 4, 1 };" << std::endl;
        file << "Line(5)  = { 4, 5 };" << std::endl;
        file << "Line(6)  = { 5, 6 };" << std::endl;
        file << "Line(7)  = { 6, 7 };" << std::endl;
        file << "Line(8)  = { 7, 8 };" << std::endl;
        file << "Line(9)  = { 8, 5 };" << std::endl;
        file << "Line(10)  = { 8, 1 };" << std::endl;
        file << "Line(11)  = { 3, 6 };" << std::endl;
        file << "Line(12)  = { 2, 7 };" << std::endl;
        S = 13;
        v = 1;
        Double dt=0;

        swww = dt + 6;
        swww1 = dt + 2;
        file << "Curve Loop(" << dt << ") = {" << 1 << ", " << 2 << ", " << 3 <<
        ", " << 4 << "};" << std::endl;
        {
          file << "Plane Surface(" << dt << ") = { ";
          file << "" << dt << " };" << std::endl;
        }
        {
          file << endl;
        }
        file << "Curve Loop(" << dt+1 << ") = {" << 6 << ", " << 7 << ", " << 8 <<
        ", " << 9 << "};" << std::endl;
        {
          file << "Plane Surface(" << dt+1 << ") = { ";
          file << "" << (dt+1) << " };" << std::endl;
        }
        {
          file << endl;
        }
        file << "Curve Loop(" << dt + 2 << ") = {" << 3 << ", " << 11 << ", " << 5 <<
         ", " << 6 << "};" << std::endl;
        file << "Curve Loop(" << dt + 3 << ") = {" << 2 << ", " << 12 << ", " << 7 <<
        ", " << 11 << "};" << std::endl;
        file << "Curve Loop(" << dt + 4  << ") = {" << 4 << ", " << 10 << ", " << 5 <<
        ", " << 9 << "};" << std::endl;
        file << "Curve Loop(" << dt + 5  << ") = {" << 8 << ", " << 10 << ", " << 1 <<
        ", " << 12 << "};" << std::endl;
        for (j = 0; j < 4; j++)
        {
          file << "Plane Surface(" << swww << ") = {" << swww1 << "};" << std::endl;
          swww += 1;
          swww1 += 1;
        }
        int fibbrree = dt;
        file << "Surface Loop(" << dt +10  << ") = { " << dt << ", " << dt + 1 << ", "<<
        dt + 6<<" , "<< dt + 7 <<", "<< dt + 8<<" , "<< dt + 9 <<"};" << std::endl;
        {
        file << endl;
        file << "Volume(" << dt << ") = {"<< dt + 10 <<"};" << std::endl;
        file << "Physical Volume(\"matrix\") = {" << dt << "};" << std::endl;
        file << endl;

         //MESH QUALITY CONTROL
         file << " Mesh 3;" << std::endl;
         file << "Mesh.Smoothing = 100;" << std::endl;

     std::string commend = "output_" + std::to_string(rd) + "_" + std::to_string(n) + ".msh";

       //GENERATE MSH FILE
       std::string command = "gmsh " + filename + " -3 -smooth 100 > gmsh.log";
       int err = system(command.c_str());

         //CREATE A SHELL FILE
         std::ofstream Output_file("king.sh", std::ios::app);
         if (Output_file.is_open())
      Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ") working..."<<"\""<<";";
       Output_file<<" ./mpap3 " + commend + "> out/rve"+ to_string(num2) <<endl;
       Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ")  all done.."<<"\""<<endl;
     num2+=1;
      }
    }
  }

}
```

Listing C.2: Cube Geometry File

C.3 Code 3: Cube With spherical-inclusion Geometry file Generator

Geometry of an isotropic material with spherical inclusion is generated using this code. Parameters of the microstructure are varied here to generate multiple RVEs.

```
#define VTK
#define SOLID
#define MESH_TOOLS
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"
#include <sstream>

int main()
{
  int num = 1 ,num2=1;   double rdv, rd;
  {
    for (double n = 1; n <= 7; n += 1) // number of inclusions
    {
       double rd,rdv;
         rd = 0.5; // radius of inclusions
         rdv =0;//radius of voids
      {
        std::string filename = "output_" + std::to_string(rd) + "_" + std::to_string(n) + ".geo";
        std::ofstream file(filename);
          if (file.is_open())
        {
          std::cout << "*success........................................*\n\n";
        }
        int j, i, r, rr, Rr, v,  S, fibre, swww1, swww;
        double d, nx, ny, nz, lc, t, a, b, z, x, y, y2, Pi, tol = 1.e-4;
        d = 0.05; // spacing
        lc =0.5; // cube mesh size
        Pi = 3.14159265358979323846;
        double Rr1 = 5, r1 = 4, r2 = 6, Rr2 = 7, r3 = 8, Rr3 = 9;
        r = 1; //Sphere LABELS
        Rr = 3;
        file << "Mesh.Binary=0;" << std::endl;
        file << "SetFactory(\"OpenCASCADE\");" << std::endl;
        double  maxr=1;  //RVE size control parameter
        a  = ( d +  maxr ); //increment
        ny = (n) * ( d + maxr); // Side of RVE cube
        b  = (0.5 * d + 0.5 * maxr ) ; //initial starting coordinate

        nx = ny;
        nz = ny;
        nz = ny;
        double F_volume_r =  (n * n * n * 4 / 3 * Pi * rd * rd * rd) / (ny * ny * ny);
        double v_volume_r =  (n * n * n * 4 / 3 * Pi * rdv * rdv * rdv) / (ny * ny * ny);
        mos<<ny<<"\n\n";
        mos << "F_volume_r is " << F_volume_r << "\n\n";
        mos << "radius of voids is " << rdv << "\n\n";
        mos << "radius of inclusion is " << rd << "\n\n";
        mos << "spacing is " << d << "\n\n";
        output_file<< "side_of_cube"+ to_string(num) <<" = " << ny <<";"<<"\n\n"<<std::endl;
        output_file << "F_volume_r"+ to_string(num) <<" = " << F_volume_r<<";" << "\n\n"<< std::endl;
        output_file << "v_volume_r"+ to_string(num) <<" = " << v_volume_r<<";" << "\n\n"<< std::endl;
        output_file << "radius_of_voids"+ to_string(num) <<" = " << rdv<<";" << "\n\n"<< std::endl;
        output_file << "radius_of_inclusion"+ to_string(num) <<" = " << rd<<";" << "\n\n"<< std::endl;
        output_file << "spacing"+ to_string(num) <<" = " << d<<";" << "\n\n"<< std::endl;
        output_file << "%||successful.....for.......MESH"+ to_string(num)+" ..................||" << "\n\n"<< std::endl;
        num+=1;
        //CREATE A CUBE
file << "Point(1)  = {" << 0.0 << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(2)  = {" << 0 << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(3)  = {" << 0 << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(4)  = {" << 0 << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(5)  = {" << nx << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(6)  = {" << nx << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(7)  = {" << nx << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(8)  = {" << nx << ", " << 0 << ", " << 0 << ", " << lc
```

```
            << " };" << std::endl;
        file << "Line(1)  = { 1, 2 };" << std::endl;
        file << "Line(2)  = { 2, 3 };" << std::endl;
        file << "Line(3)  = { 3, 4 };" << std::endl;
        file << "Line(4)  = { 4, 1 };" << std::endl;
        file << "Line(5)  = { 4, 5 };" << std::endl;
        file << "Line(6)  = { 5, 6 };" << std::endl;
        file << "Line(7)  = { 6, 7 };" << std::endl;
        file << "Line(8)  = { 7, 8 };" << std::endl;
        file << "Line(9)  = { 8, 5 };" << std::endl;
        file << "Line(10)  = { 8, 1 };" << std::endl;
        file << "Line(11)  = { 3, 6 };" << std::endl;
        file << "Line(12)  = { 2, 7 };" << std::endl;
        //ADD SPHERICAL INCLUSION
        for (y = b; y <= ny; y += a)
          for (x = b; x < nx; x += a)
            for (z = b; z < nx; z += a)
            {
              file << "Sphere(" << r << ")"<< " = "<< "{" << x << ", "
              << y << ", " << z << ", " << rd << "};" << std::endl;
              r += 1;
            }
        double dr = 0;
        double dt = r + dr;
        S = dt+11;
        v = 1;
        swww = dt + 6;
        swww1 = dt + 2;
        file << "Curve Loop(" << dt << ") = {" << 1 << ", " << 2 << ", " << 3
            <<
        ", " << 4 << "};" << std::endl;
        {
        file << "Plane Surface(" << dt << ") = { ";
        file << "" << dt << " };" << std::endl;
        }
        {
          file << endl;
        }
        file << "Curve Loop(" << dt+1 << ") = {" << 6 << ", " << 7 << ", " << 8
            <<
        ", " << 9 << "};" << std::endl;
        {
        file << "Plane Surface(" << dt+1 << ") = { ";
        file << "" << (dt+1) << " };" << std::endl;
        }
        {
          file << endl;
        }
        file << "Curve Loop(" << dt + 2 << ") = {" << 3 << ", " << 11 << ", "
            << 5 <<
         ", " << 6 << "};" << std::endl;
        file << "Curve Loop(" << dt + 3 << ") = {" << 2 << ", " << 12 << ", "
            << 7 <<
        ", " << 11 << "};" << std::endl;
        file << "Curve Loop(" << dt + 4  << ") = {" << 4 << ", " << 10 << ", "
            << 5 <<
        ", " << 9 << "};" << std::endl;
        file << "Curve Loop(" << dt + 5  << ") = {" << 8 << ", " << 10 << ", "
            << 1 <<
        ", " << 12 << "};" << std::endl;
        for (j = 0; j < 4; j++)
        {
          file << "Plane Surface(" << swww << ") = {" << swww1 << "};" << std::
              endl;
          swww += 1;
          swww1 += 1;
        }
        int fibbrree = dt;
        file << "Surface Loop(" << dt +10  << ") = { " << dt << ", " << dt + 1 << ", "<<
        dt + 6<<" , "<< dt + 7 <<", "<< dt + 8<<" , "<< dt + 9 <<"};" << std::endl;
        {
        file << endl;
        }
        for (j = 0; j < (r - 1 ); j++)
        {
          file << "Surface Loop(" << S << ") = { " << v << "};" << std::endl;
          S += 1;
          v += 1;
        }
        int fbr =dt + 9;
        file << "Volume(" << dt << ") = {";
        while (fbr <= (2*dt +7 ))
        {
          fbr = fbr + 1;
          file << fbr << ", ";
        }
        file << "" << (2 * dt + 9 ) << "};" << std::endl;
        //LABEL THE MATRIX
        file << "Physical Volume(\"matrix\") = {" << dt << "};" << std::endl;
        //ASIGN THE INCLUSION
        {
          int inclu = 0;
          file << "Physical Volume(\"inclusions\") = { ";
          while (inclu < (dt)-2)
          {
            inclu = inclu + 1;
            file << inclu << ", ";
          }
        }

        {
        file << "" << (dt -1) << " };" << std::endl;
        }
        file << endl;
        //MESH QUALITY CONTROL
        // file<<"Mesh.Algorithm3D = 10;"<<std::endl;
         //file<<"Mesh.Algorithm = 5;"<<std::endl;
         //file << " Mesh.CharacteristicLengthExtendFromBoundary =1;" << std::endl;
         //file << "Mesh.CharacteristicLengthFromCurvature = 0.8;" << std::endl;
         //file << "Mesh.CharacteristicLengthFromPoints =0.01;" << std::endl;
       //file << "Mesh.CharacteristicLengthFromEdges = 0;" << std::endl;
       //file << "Mesh.CharacteristicLengthFromFaces = 0;" << std::endl;
     //  file << "Mesh.CharacteristicLengthFromVolumes = 0.1;" << std::endl;
       //file << "Mesh.ElementOrder = 2;" << std::endl;
       file << "Mesh.CharacteristicLengthFromCurvature = 8;" << std::endl;
       //  file << "Mesh.CharacteristicLengthMin = 0.01;" << std::endl;
         //file << "Mesh.CharacteristicLengthMax = 1;" << std::endl;
         // file<<"Mesh.MeshSize{"<<r3<<"} = 0.2;"<<std::endl;
        // file<<"Mesh.MeshSizeFactor = 0.25; "<<std::endl;
         file << " Mesh 3;" << std::endl;
         file << "Mesh.Smoothing = 100;" << std::endl;

```

```
      std::string commend = "output_" + std::to_string(rd) + "_" + std::to_string(n) + ".msh";
        //GENERATE MSH FILE
        std::string command = "gmsh " + filename + " -3 -smooth 100 > gmsh.log";
        int err = system(command.c_str());
          std::ofstream Output_file("king.sh", std::ios::app);
          if (Output_file.is_open())
       // CREATE A SHELL FILE
       Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ") working..."<<"\""<<";";
        Output_file<<" ./mpap3 " + commend + "> out/rve"+ to_string(num2) <<endl;
        Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ")  all done.."<<"\""<<endl;
      num2+=1;
       }
     }
   }

}
```

Listing C.3: Cube With Spherical-inclusion Geometry File

C.4 Code 4: Cube With spherical voids and inclusion Geometry file Generator

Geometries of an isotropic material with spherical inclusion and voids are generated using this code. Parameters of the microstructure are varied here to generate multiple RVEs.

```
#define VTK
#define SOLID
#define MESH_TOOLS
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"
#include <sstream>

int main()
{
   double rdv, rd;
 for (double rdv = 0.145; rdv <= 0.5; rdv += 0.06)//RADIUS OF VOIDS
  {
    for (double rd = 0.145; rd <= 0.5; rd += 0.06)//RADIUS OF FIBRES
    {
      {
        std::string filename = "output_" + std::to_string(rd) + "_" + std::to_string(rdv) + ".geo";
        std::ofstream file(filename);
          if (file.is_open())
        {
          std::cout << "%success.....................................\n\n";
        }

        int j, i, r, rr, Rr, v, n, S, fibre, swww1, swww;
        double d, nx, ny, nz, lc, t, a, b, z, x, y, y2, Pi, tol = 1.e-4;
        d = 0.05; // spacing
        n = 2; // number of inclusions
        lc =0.1; //cube mesh size
        Pi = 3.14159265358979323846;
        double Rr1 = 5, r1 = 4, r2 = 6, Rr2 = 7, r3 = 8, Rr3 = 9;
        r = 1;
        Rr = 3;
        file << "Mesh.Binary=0;" << std::endl;
        file << "SetFactory(\"OpenCASCADE\");" << std::endl;
        double  maxr=1;  //maximum RVE size control parameter
        a  = 2 * d + 2 * maxr ; //increment
        ny = (n) * (2 * d + 2 * maxr); // size of cube
        b  = (0.5 * d + 0.5 * maxr ) ; //initial starting coordinate
        nx = ny;
        nz = ny;
        nz = ny;
        double dr          = 2 * n;
        double F_volume_r = 0.5 *  (dr * dr * dr * 4 / 3 * Pi * rd * rd * rd) / (ny * ny * ny);
        double v_volume_r = 0.5 *  (dr * dr * dr * 4 / 3 * Pi * rdv * rdv * rdv ) / (ny * ny * ny);
        //mos << "F_volume_r  = " << F_volume_r << "\n\n";
       // mos << "v_volume_r = " << v_volume_r << "\n\n";
       // mos << "radius of voids = " << rdv << "\n\n";
        //mos << "radius of inclusion = " << rd << "\n\n";
        //mos << "spacing = " << d << "\n\n";

        //ADD CUBE
file << "Point(1)  = {" << 0.0 << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(2)  = {" << 0 << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(3)  = {" << 0 << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(4)  = {" << 0 << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(5)  = {" << nx << ", " << 0 << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(6)  = {" << nx << ", " << ny  << ", " << nz  << ", " << lc << " };" << std::endl;
        file << "Point(7)  = {" << nx << ", " << ny  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(8)  = {" << nx << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;

        file << "Line(1)  = { 1, 2 };" << std::endl;
        file << "Line(2)  = { 2, 3 };" << std::endl;
        file << "Line(3)  = { 3, 4 };" << std::endl;
        file << "Line(4)  = { 4, 1 };" << std::endl;
        file << "Line(5)  = { 4, 5 };" << std::endl;
        file << "Line(6)  = { 5, 6 };" << std::endl;
        file << "Line(7)  = { 6, 7 };" << std::endl;
        file << "Line(8)  = { 7, 8 };" << std::endl;
        file << "Line(9)  = { 8, 5 };" << std::endl;
        file << "Line(10)  = { 8, 1 };" << std::endl;
        file << "Line(11)  = { 3, 6 };" << std::endl;
        file << "Line(12)  = { 2, 7 };" << std::endl;

        //ADD INCLUSIONS
        for (y = b; y <= ny; y += a)
          for (x = b; x < nx; x += a)
            for (z = b; z < nx; z += a)
            {
             file << "Sphere(" << r << ")"<< " = "<< "{" << x << ", "
              << y << ", " << z << ", " << rdv << "};" << std::endl;
                 //
              file << "Sphere(" << r+1 << ")"<< " = "<< "{" << x << ", "
              << y << ", " << z + a / 2 << ", " << rd << "};" << std::endl ;
                  //
              file << "Sphere(" << r + 2 << ")"<< " = "<< "{" << x + a / 2 <<
              ", " << y << ", " << z + a / 2 << ", " << rdv << " };" << std::
              endl;
                  //
              file << "Sphere(" << r+3 << ")"<< " = "<< "{" << x + a / 2 <<
              ", " << y << ", " << z << ", " << rd << "};" << std::endl;
                  //
              file << "Sphere(" << r + 4 << ")"<< " = "<< "{" << x + a / 2 <<
              ", " << y + a / 2 << ", " << z << ", " << rdv << "};" << std::
              endl;
                  //
              file << "Sphere(" << r + 5 << ")"<< " = " << "{" << x + a / 2 <<
              ", " << y + a / 2 << ", " << z + a / 2 << ", " << rd << "};"<<
                  std
              ::endl;
                //
              file << "Sphere(" << r + 6 << ")"<< " =  "<< "{" << x << ", " <<
                  y +
               a / 2 << ", " << z + a / 2 << ", " << rdv << "};" << std::endl;
                //
              file << "Sphere(" << r + 7 << ")"<< " = "<< "{" << x << ", " << y
              + a / 2 << ", " << z << ", " << rd << "};"<< std::endl;
              r += 8;
            }
        double dr = (n >= 1) ? 0 : 7;
        double dt = r + dr;
        S = dt+11;
        v = 1;
        swww = dt + 6;
        swww1 = dt + 2;
        file << "Curve Loop(" << dt << ") = {" << 1 << ", " << 2 << ", " << 3
            <<
        ", " << 4 << "};" << std::endl;
        {
        file << "Plane Surface(" << dt << ") = { ";
        file << "" << dt << " };" << std::endl;
        }
        {
        file << endl;
        }
        file << "Curve Loop(" << dt+1 << ") = {" << 6 << ", " << 7 << ", " << 8
             <<
        ", " << 9 << "};" << std::endl;
        {
        file << "Plane Surface(" << dt+1 << ") = { ";
        file << "" << (dt+1) << " };" << std::endl;
        }
        {
        file << endl;
        }
        file << "Curve Loop(" << dt + 2 << ") = {" << 3 << ", " << 11 << ", "
            << 5 <<
         ", " << 6 << "};" << std::endl;
        file << "Curve Loop(" << dt + 3 << ") = {" << 2 << ", " << 12 << ", "
            << 7 <<
        ", " << 11 << "};" << std::endl;
        file << "Curve Loop(" << dt + 4  << ") = {" << 4 << ", " << 10 << ", "
            << 5 <<
        ", " << 9 << "};" << std::endl;
        file << "Curve Loop(" << dt + 5  << ") = {" << 8 << ", " << 10 << ", "
```

```
       << 1 <<
   ", " << 12 << "};" << std::endl;
   for (j = 0; j < 4; j++)
   {
   file << "Plane Surface(" << swww << ") = {" << swww1 << "};" << std::
       endl;
   swww += 1;
   swww1 += 1;
   }
   int fibbrree = dt;
   file << "Surface Loop(" << dt +10  << ") = { " << dt << ", " << dt + 1
       << ", "<<
   dt + 6<<" , "<< dt + 7 <<", "<< dt + 8<<" , "<< dt + 9 <<"};" << std::
       endl;
   {
   file << endl;
   }
   for (j = 0; j < (r - 1 ); j++)
   {
   file << "Surface Loop(" << S << ") = { " << v << "};" << std::endl;
   S += 1;
   v += 1;
   }
   int fbr =dt + 9;
   file << "Volume(" << dt << ") = {";
   while (fbr <= (2*dt +7 ))
   {
   fbr = fbr + 1;
   file << fbr << ", ";
   }
   file << "" << (2 * dt + 9 ) << "};" << std::endl;
   file << "Physical Volume(\"matrix\") = {" << dt << "};" << std::endl;
   {
   int inclu = 0;
   file << "Physical Volume(\"inclusions\") = { ";
   while (inclu < (dt)-4)
     {
     inclu = inclu + 2;
     file << inclu << ", ";
     }
   }
   {
   file << "" << (dt -1) << " };" << std::endl;
   }
   file << endl;
   //MESH QUALITY CONTROL
   // file<<"Mesh.Algorithm3D = 10;"<<std::endl;
    //file<<"Mesh.Algorithm = 5;"<<std::endl;
    //file << " Mesh.CharacteristicLengthExtendFromBoundary =1;" << std::
        endl;
    //file << "Mesh.CharacteristicLengthFromCurvature = 0.8;" << std::endl
        ;
    //file << "Mesh.CharacteristicLengthFromPoints =0.01;" << std::endl;
  //file << "Mesh.CharacteristicLengthFromEdges = 0;" << std::endl;
  //file << "Mesh.CharacteristicLengthFromFaces = 0;" << std::endl;
//  file << "Mesh.CharacteristicLengthFromVolumes = 0.1;" << std::endl;
  //file << "Mesh.ElementOrder = 2;" << std::endl;
  file << "Mesh.CharacteristicLengthFromCurvature = 8;" << std::endl;
  //  file << "Mesh.CharacteristicLengthMin = 0.01;" << std::endl;
    //file << "Mesh.CharacteristicLengthMax = 1;" << std::endl;
    // file<<"Mesh.MeshSize{"<<r3<<"} = 0.2;"<<std::endl;
   // file<<"Mesh.MeshSizeFactor = 0.25; "<<std::endl;
    file << " Mesh 3;" << std::endl;
```

```
         file << "Mesh.Smoothing = 100;" << std::endl;
     std::string commend = "output_" + std::to_string(rd) + "_" + std::to_string(rdv) + ".msh";
       //GENERATE MSH FILE
       std::string command = "gmsh " + filename + " -3 -smooth 100 > gmsh.log";
       int err = system(command.c_str());
        //GENERATE A SHELL FILE
         std::ofstream Output_file("king.sh", std::ios::app);
         if (Output_file.is_open())
      Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ") working..."<<"\""<<";";
       Output_file<<" ./mpap3 " + commend + "> out/rve"+ to_string(num2) <<endl;
       Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ")  all done.."<<"\""<<endl;
     num2+=1;



      }
    }
  }
}
```

Listing C.4: Cube With Spherical voids and inclusion Geometry File

C.5 Code 5: Cube With Cylindrical inclusion Geometry file Generator

This code generates the geometry of a transversely isotropic material with cylindrical inclusion. Parameters of the microstructure are varied here to generate multiple RVEs.

```
#define VTK
#define SOLID
#define MESH_TOOLS
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"
#include <sstream>

int main()
{
  int num=1, num2=1;
   double rdv, rd;
  {
    for (double n = 1; n <= 7; n += 1) //number of fibres
    {
       double rd,rdv;
         rd = 0.5; // radius of inclusions
         rdv =0;//radius of voids
      {
        std::string filename = "outputcyl_" + std::to_string(rd) + "_" + std::to_string(n) + ".geo";
        std::ofstream file(filename);
          if (file.is_open())
        {
          std::cout << "*success....................................*\n\n";
        }

   int j,i,p,q,r,rr,s,h,v,I,S,fibre,SS,tv,sw1,sw,sww1,sww,swww,swww1;
   double nx,ny,nz,lc,l,t,rd,d,z,x,y,y2,Pi,tol=1.e-4,vv,vvv,vvvv;
    rd = 0.5;
    d  = 0.05;
    double rdv = 0;
    lc=0.2;
    Pi= 3.14159265358979323846;
     r=2;
  sw=3*n*n+2;sw1=15;
  sww=3*n*n+n*n+3;sww1=13;
  swww=3*n*n+3;swww1=5*n*n+3;
  file<<"Mesh.Binary=0;"<<std::endl;
  file<<"SetFactory(\"OpenCASCADE\");"<<std::endl;
double maxr = 2 * rd;
double  a  = ( d +  maxr );
        ny = (n) * ( d + maxr);
double  b  = (0.5 * d + 0.5 * maxr ) ;

        nx = ny;
        nz = ny;
        nz = ny;
        double F_volume_r =  (n * n  * Pi * rd * rd) / (ny * ny);
        double v_volume_r =  (n * n * n * 4 / 3 * Pi * rdv * rdv * rdv) / (ny * ny * ny);
        mos<<ny<<"\n\n";
        mos << "F_volume_r is " << F_volume_r << "\n\n";
        mos << "radius of voids is " << rdv << "\n\n";
        mos << "radius of inclusion is " << rd << "\n\n";
        mos << "spacing is " << d << "\n\n";
        //CREATE A CUBE
    //add points
    file<<"Point(1)  = {"<< 0<<", "<<0<<", "<<0<<", "<<lc<<" };"<<std::endl;
    file<<"Point(2)  = {"<< 0<<", "<<ny<<", "<<-.000<<", "<<lc<<" };"<<std::endl;
    file<<"Point(3)  = {"<< 0<<", "<<ny<<", "<<nz<<", "<<lc<<" };"<<std::endl;
    file<<"Point(4)  = {"<< 0<<", "<<0<<", "<<nz<<", "<<lc<<" };"<<std::endl;
    file<<"Point(5)  = {"<< nx<<", "<<0<<", "<<nz<<", "<<lc<<" };"<<std::endl;
    file<<"Point(6)  = {"<< nx<<", "<<ny<<", "<<nz<<", "<<lc<<" };"<<std::endl;
    file<<"Point(7)  = {"<< nx<<", "<<ny<<", "<<0<<", "<<lc<<" };"<<std::endl;
    file<<"Point(8)  = {"<< nx<<", "<<0<<", "<<0<<", "<<lc<<" };"<<std::endl;
    //add lines
    file<<"Line(1)  = { 1, 2 };"<<std::endl;
    file<<"Line(2)  = { 2, 3 };"<<std::endl;
    file<<"Line(3)  = { 3, 4 };"<<std::endl;
    file<<"Line(4)  = { 4, 1 };"<<std::endl;
    file<<"Line(5)  = { 4, 5 };"<<std::endl;
    file<<"Line(6)  = { 5, 6 };"<<std::endl;
    file<<"Line(7)  = { 6, 7 };"<<std::endl;
    file<<"Line(8)  = { 7, 8 };"<<std::endl;
     file<<"Line(9)  = { 8, 5 };"<<std::endl;
    file<<"Line(10)  = { 8, 1 };"<<std::endl;
    file<<"Line(11)  = { 3, 6 };"<<std::endl;
    file<<"Line(12)  = { 2, 7 };"<<std::endl;

   //add cylinders/fibres
        for (y = b; y <= ny; y += a)
          for (z = b; z < nx; z += a)
   {
   file<<"Cylinder("<<r<<")"<<" = "<<"{"<<0<<", "<<y<<", "<<z<<", "<<ny<<", "<<0<<", "<<0<<", "<<rd<<", "<<"2*Pi};"<<std::endl;
     r+=1;
   }
   file<<"Curve Loop("<<3*n*n+1<<") = {"<<1<<", "<<2<<", "<<3<<", "<<4<<"};"<<std::endl;
   for (j=0;j<n*n;j++)
   {
   file<<"Curve Loop("<<sw<<") = {"<<sw1<<"};"<<std::endl;
   sw+=1;sw1+=3;
   }
   {
   int fibree =3*n*n;
   file<<"Plane Surface("<<3*n*n+1<<") = { ";
   while (fibree <=(3*(n*n)+n*n-1))
    {
    fibree=fibree+1;
    file <<fibree<<", ";
    }
   file<<""<<(3*(n*n)+n*n+1)<<" };"<<std::endl;
    }
  file<<endl;
  file<<"Curve Loop("<<3*n*n+n*n+2<<") = {"<<6<<", "<<7<<", "<<8<<", "<<9<<"};"<<std::endl;
 {
  for (j=0;j<n*n;j++)
   {
  file<<"Curve Loop("<<sww<<") = {"<<sww1<<"};"<<std::endl;
  sww+=1;sww1+=3;
  }
}
{
 int fibrree =3*n*n+n*n+1;
 file<<"Plane Surface("<<3*n*n+2<<") = { ";
 while (fibrree <=(3*(n*n)+2*n*n))
    {
   fibrree=fibrree+1;
   file <<fibrree<<", ";
    }
file<<""<<(3*(n*n)+2*n*n+2)<<" };"<<std::endl;
}
   file<<endl;
   file<<"Curve Loop("<<3*n*n+2*n*n+3<<") = {"<<3<<", "<<11<<", "<<5<<", "<<6<<"};"<<std::endl;
   file<<"Curve Loop("<<3*n*n+2*n*n+4<<") = {"<<2<<", "<<12<<", "<<7<<", "<<11<<"};"<<std::endl;
   file<<"Curve Loop("<<3*n*n+2*n*n+5<<") = {"<<4<<", "<<10<<", "<<5<<", "<<9<<"};"<<std::endl;
   file<<"Curve Loop("<<3*n*n+2*n*n+6<<") = {"<<8<<", "<<10<<", "<<1<<", "<<12<<"};"<<std::endl;
   for (j=0;j<4;j++)
   {
   file<<"Plane Surface("<<swww<<") = {"<<swww1<<"};"<<std::endl;
   swww+=1;swww1+=1;
   }
    int fibbrree =3*n*n;
    int cibre=-2;
   file<<"Surface Loop("<<3*n*n+3<<") = { ";
   while (fibbrree <=(3*(n*n)+5))
    {
    fibbrree=fibbrree+1;
    file <<fibbrree<<", ";
    }

  while (cibre <=(3*(n*n)-6))
    {
    cibre=cibre+3;
    file<<cibre<<", ";
    }
   file<<""<<((3*n*n-2))<<"};"<<std::endl;
{
   file<<endl;
}
   int fbr =3*n*n+2;
   file<<" Volume("<<2*(n*n*n+2)<<") = {";
   file<<""<<3*n*n+3<<"};"<<std::endl;
   file<<"Physical Volume(\"matrix\", "<<3*n*n+4<<") = {"<<2*(n*n*n+2)<<"};"<<std::endl;
   {
   int inclu =1;//3*n*n+3;//n*n+2;
   file<<"Physical Volume(\"inclusions\", "<<3*n*n+5<<") = { ";
   while (inclu < (n*n))
    {
    inclu=inclu+1;
    file<<inclu<<", ";
    }
   }
   {
   file<<""<<(n*n)+1<<" };"<<std::endl;
  }
   //MESH QUALITY CONTROL

  //cout<<rd<<", "<<ny<<std::endl;
 //file<<"Field[1] = Cylinder ;"<<std::endl;
 //file<<"Field[1].VIn = "<<lc<<";"<<std::endl;
 //file<<"Field[1].VOut = "<<lc<<";"<<std::endl;
 file << "Mesh.CharacteristicLengthFromCurvature = 15;" << std::endl;
//file<<"Mesh.MinimumElementsPerTwoPi = 30;"<<std::endl;
//file<<"Mesh.Algorithm3D = 10;"<<std::endl;
//file<<"Mesh.Algorithm = 2;"<<std::endl;
         file << " Mesh 3;" << std::endl;
         file << "Mesh.Smoothing = 100;" << std::endl;



     std::string commend = "outputcyl_" + std::to_string(rd) + "_" + std::to_string(n) + ".msh";

     //GENERATE A MESH FILE
       std::string command = "gmsh " + filename + " -3 -smooth 100 > gmsh.log";
       int err = system(command.c_str());

      // CREAT A SHELL FILR TO STORE THE MESH
         std::ofstream Output_file("kingcyl.sh", std::ios::app);
         if (Output_file.is_open())
      Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ") working..."<<"\""<<";";
       Output_file<<" ./mpap3 " + commend + "> out/rve"+ to_string(num2) <<endl;
       Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ")  all done.."<<"\""<<endl;
     num2+=1;
      }
    }
  }
}
```

Listing C.5: Cube With Cylindrical inclusion Geometry File

C.6 Code 6: Cube With Cylindrical Discontinuous inclusion Geometry file Generator

The code provide geometry of a transversely isotropic material with discontinuous cylindrical inclusion. Parameters of the microstructure are varied here to generate multiple RVEs.

```
#define VTK
#define SOLID
#define MESH_TOOLS
#define MA41
#include "Fem.h"
#include <iostream>
#include <fstream>
#include <vector>
#include "MpapGlobal.h"
#include "FemSolidRVE.h"
#include "MaterialHooke.h"
#include <sstream>


int main()
{
  int num = 1 ,num2=1,mshi=1;
  std::ofstream output_file("sph_IV.m");
  if (output_file.is_open())

        // std::ofstream Output_file("king.sh", std::ios::app);
       //  if (Output_file.is_open())

       // Output_file<<"#!/bin/bash"<<endl;


   double rdv, rd; //d, maxr=1;
  //for (double rdv = 0.1; rdv < maxr; rdv += 0.1)
 for (double l = 0.5; l <= 1.5; l += 0.25) //0.5>>>> 1.5>>>0.25
     //0.04---1.321//0.98====0.135  (0.5===========>1.5)
  {
    for (double dpy = 0.01; dpy <= 0.1; dpy += 0.0225)// 0.01>>>> 0.1>>>0.03
        for (double rd = 0.04; rd <= 0.985; rd += 0.135) (0.01==>0.1)
    {
       for (double dp = 0.01; dp <= 0.1; dp += 0.0225) // 0.1>>>> 0.1>>>0.03
           (0.01=====>0.1)
       //double rd=0;//,rdv;
         //rd = 0.5; // radius of inclusions
        // rdv =1;//radius of voids

      {
        //for (double dpz = 0.1; dpz <= 0.1; dpz += 0.03) // 0.01>>>>
            0.1>>>0.03 (0.01====>0.1)

        {


        std::string filename = "output_"+ std::to_string(mshi) + ".geo";
        std::ofstream file(filename);
         //mshi+=1;
          if (file.is_open())

        //std::ofstream file("rve2ex104-mod.geo");
        //if (file.is_open())
        {
          std::cout << "%success
              ......................................................................\
              n\n";
        }

        int j, i, r, rr, Rr, v, n, S, fibre, swww1, swww;
        double d, nx, ny, nz, lc, t, a, b, z, x, y, y2, Pi, tol = 1.e-4;
        double rdv = 0.08 ,rd=0;

       //  rd = 1; // radius of inclusions
       //  rdv =1;//radius of voids
        d = 0.05; // spacing
        n = 2; // number of inclusions
        lc =2*rdv;
        Pi = 3.14159265358979323846;

    //  if(rdv>=0.1)
      {
      //  lc = rd*0.8;
      }
      //else
      {
       // lc= 0.1*0.8;
      }

        double Rr1 = 5, r1 = 4, r2 = 6, Rr2 = 7, r3 = 8, Rr3 = 9;
        r = 1;
        Rr = 3;
       // S = r+ 2;
        //v = 1;

       // swww = r+ 8;
       // swww1 = r+ 4;

        file << "Mesh.Binary=0;" << std::endl;
        file << "SetFactory(\"OpenCASCADE\");" << std::endl;
        double  maxr = 1;
        /*a  = (2 * d + 2 * maxr) ;
        ny = (n) * (2 * d + 2 * maxr);
        b  = (0.5 * d + 0.5 * maxr ) ;*/
       // d = maxr;


        //b = maxr;//(0.5*d + maxr);
       // a = 4 * maxr;//(4 * maxr + 2 * d );
       // ny = (n-1)*(7*maxr);//(n - 1)*(a + d + 2 * maxr)+ d +maxr;  double dp = 2*maxr-(rd+rdv); //2rdv

       double dpz = dpy;
      double  ax, bz,by; //dpy = 0.001, dpz=0.001,l = .5; //l=0.6-1
       ax = dp + l;//dp;
       double ay = 2*(2*rdv+ dpy),az= 2*(2*rdv+dpz);
       b  = dp/2;// - 0.5*rdv;
       by = dpy/2 + rdv;
       bz = dpz/2 + rdv;
       ny = 2*(4*(2 * rdv + dpy));//-(rdv+dpy)+rdv +0.5*dpy;
       nx = 3*dp+ 3*l;//2*(4*(2 * rdv + dp))-(rdv+a); 2*4
       nz = 2*(4*(2 * rdv + dpz));//-(rdv+dpz)+rdv + 0.5 * dpz;

      //double  np = a + l;



        //nx = ny;
        //nz = ny;
        //nz = ny;
        double RVEarea          = nx*ny*nx;
        double F_volume_r = (Pi* rdv * rdv * l)*256/RVEarea;//0.5 *  (drS * drS * drS * 4 / 3 * Pi * rd * rd * rd) / (ny * ny * ny);
        double v_volume_r = 0;// *  (drS * drS * drS * 4 / 3 * Pi * rdv * rdv * rdv) / (ny * ny * ny);

        //mos<<ny<<"\n\n";
        //mos << "F_volume_r  = " << F_volume_r << "\n\n";
       // mos << "v_volume_r = " << v_volume_r << "\n\n";
       // mos << "radius of voids = " << rdv << "\n\n";
        //mos << "radius of inclusion = " << rd << "\n\n";
        //mos << "spacing = " << d << "\n\n";

        output_file<< "side_of_cube"+ to_string(num) <<" = " << ny <<";"<<"\n\n"<<std::endl;
        output_file << "F_volume_r"+ to_string(num) <<" = " << F_volume_r<<";" << "\n\n"<< std::endl;
        output_file << "v_volume_r"+ to_string(num) <<" = " << v_volume_r<<";" << "\n\n"<< std::endl;
        output_file << "radius_of_voids"+ to_string(num) <<" = " << rd<<";" << "\n\n"<< std::endl;
        output_file << "radius_of_inclusion"+ to_string(num) <<" = " << rdv<<";" << "\n\n"<< std::endl;
        output_file << "longitudinal_sp"+ to_string(num) <<" = " << dp/rdv<<";" << "\n\n"<< std::endl;
        output_file << "Transverse_sp"+ to_string(num) <<" = " << dpy/rdv<<";" << "\n\n"<< std::endl;
        output_file << "Horizontal_sp"+ to_string(num) <<" = " << dpz/rdv<<";" << "\n\n"<< std::endl;
        output_file << "Aspect_ratio"+ to_string(num) <<" = " << l/rdv<<";" << "\n\n"<< std::endl;
        output_file << "%||successful......for.......MESH"+ to_string(num)+"...................||" << "\n\n"<< std::endl;
        num+=1;
        //double l = 0.5;
        double  np =  2*a+ 4*l; //nxd //a + l;//, nt = 8*rdv + 4*d; //2*d + (3*rdv+a)  ;

double nyd=ny,nxd=nx ,nzd=nz ,cpx =0;//0.5*l ; //+ a/2 +rdv/2)
file << "Point(1)  = {" <<cpx << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(2)  = {" << cpx<< ", " << nyd  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(3)  = {" << cpx << ", " << nyd  << ", " << nzd  << ", " << lc << " };" << std::endl;
        file << "Point(4)  = {" << cpx<< ", " << 0 << ", " << nzd  << ", " << lc << " };" << std::endl;
        file << "Point(5)  = {" << nxd << ", " << 0 << ", " << nzd  << ", " << lc << " };" << std::endl;
        file << "Point(6)  = {" << nxd << ", " << nyd  << ", " << nzd  << ", " << lc << " };" << std::endl;
        file << "Point(7)  = {" << nxd << ", " << nyd  << ", " << 0 << ", " << lc << " };" << std::endl;
        file << "Point(8)  = {" << nxd << ", " << 0 << ", " << 0 << ", " << lc << " };" << std::endl;

        file << "Line(1)  = { 1, 2 };" << std::endl;
        file << "Line(2)  = { 2, 3 };" << std::endl;
        file << "Line(3)  = { 3, 4 };" << std::endl;
        file << "Line(4)  = { 4, 1 };" << std::endl;
        file << "Line(5)  = { 4, 5 };" << std::endl;
        file << "Line(6)  = { 5, 6 };" << std::endl;
        file << "Line(7)  = { 6, 7 };" << std::endl;
        file << "Line(8)  = { 7, 8 };" << std::endl;
        file << "Line(9)  = { 8, 5 };" << std::endl;
        file << "Line(10)  = { 8, 1 };" << std::endl;
        file << "Line(11)  = { 3, 6 };" << std::endl;
        file << "Line(12)  = { 2, 7 };" << std::endl;

        //double dp = 2*maxr-(rd+rdv); //2rdv

        for (y = by ; y <= ny; y += ay)
          for (x = b; x < nx+ 0.5*l; x += ax)
            for (z = bz; z < nz; z += az)
            {

            // file << "Sphere(" << r << ")"<< " = "<< "{" << x << ", "
            //  << y << ", " << z << ", " << rdv << "};" << std::endl;

             // file << "Cylinder(" << r << ")"<< " = "<< "{" << x-rdv << ", "
             // << y-rdv << ", " << z-rdv << ", " << l << ", "<<0<<", "<<0<<",
                 "<<rdv<<", 2*Pi};" << std::endl;

            file << "Cylinder(" << r << ")"<< " = "<< "{" << x-0.5*l << ", "
              << y << ", " << z << ", " << l << ", "<<0<<", "<<0<<", "<<rdv<<",
                   2*Pi};" << std::endl;




             //file << "Cylinder(" << r + 1 << ")"<< " = "<< "{" << x+rdv << ",
                  "
             // << y-rdv+a/2  << ", " << z-rdv+ a/2 << ", " << l << ", "<<0<<",
                  "<<0<<", "<<rdv<<", 2*Pi};" << std::endl;
             // here uknow -
             file << "Cylinder(" << r + 1 << ")"<< " = "<< "{" << x + dp/2 << "
                 , "
              << y << ", " << z+ az/2 << ", " << l << ", "<<0<<", "<<0<<", "<<
                  rdv<<", 2*Pi};" << std::endl;
              //
                  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                   x-rdv-dp + dpz/2 //x+rdv+dp/2

              file << "Cylinder(" << r +2<< ")"<< " = "<< "{" << x-0.5*l << ",
                  "
              << y +ay/2<< ", " << z +az/2<< ", " << l << ", "<<0<<", "<<0<<",
                  "<<rdv<<", 2*Pi};" << std::endl;

              //unknown -
              file << "Cylinder(" << r + 3 << ")"<< " = "<< "{" << x + dp/2 <<
                  ", "
              << y+ay/2  << ", " << z<< ", " << l << ", "<<0<<", "<<0<<", "<<
                  rdv<<", 2*Pi};" << std::endl;

                //
             // file << "Sphere(" << r + 7 << ")"<< " = "<< "{" << x << ", " <<
                  y
             // + a / 2 << ", " << z << ", " << rd << "};"<< std::endl;

            //  file << "Cylinder(" << r + 7 << ")"<< " = "<< "{" << x-rd << ",
                 "
             // << y-rd + a / 2 << ", " << z-rd << ", " << rd << ", "<<0<<",
                 "<<0<<", "<<rd<<", 2*Pi};" << std::endl;


              r += 4;

            }
        double dr = (n >= 1) ? 0 : 7;
        double dt = r*3 ;// + dr;
        S = dt+11;
        v = 1;

        swww = dt + 6;
        swww1 = dt + 2;

        file << "Curve Loop(" << dt << ") = {" << 1 << ", " << 2 << ", " << 3
            <<
        ", " << 4 << "};" << std::endl;
        {
          file << "Plane Surface(" << dt << ") = { ";
          file << "" << dt << " };" << std::endl;
        }
        {
          file << endl;
        }
        file << "Curve Loop(" << dt+1 << ") = {" << 6 << ", " << 7 << ", " << 8
             <<
        ", " << 9 << "};" << std::endl;

        {
          //int fibrree = 3 * n * n + n * n + 1;

          file << "Plane Surface(" << dt+1 << ") = { ";
          file << "" << (dt+1) << " };" << std::endl;
        }

        {
          file << endl;
        }
        file << "Curve Loop(" << dt + 2 << ") = {" << 3 << ", " << 11 << ", "
            << 5 <<
         ", " << 6 << "};" << std::endl;
        file << "Curve Loop(" << dt + 3 << ") = {" << 2 << ", " << 12 << ", "
            << 7 <<
        ", " << 11 << "};" << std::endl;
        file << "Curve Loop(" << dt + 4  << ") = {" << 4 << ", " << 10 << ", "
            << 5 <<
        ", " << 9 << "};" << std::endl;
        file << "Curve Loop(" << dt + 5  << ") = {" << 8 << ", " << 10 << ", "
            << 1 <<
        ", " << 12 << "};" << std::endl;

        for (j = 0; j < 4; j++)
        {
          file << "Plane Surface(" << swww << ") = {" << swww1 << "};" << std::
              endl;
          swww += 1;
          swww1 += 1;
        }
        int fibbrree = dt;
        file << "Surface Loop(" << dt +10  << ") = { " << dt << ", " << dt + 1
            << ", "<<
        dt + 6<<" , "<< dt + 7 <<", "<< dt + 8<<" , "<< dt + 9 <<"};" << std::
            endl;


        {
          file << endl;
```

```
       }

for (j = 0; j < ((r - 1) * 3 ); j++)
       {
         file << "Surface Loop(" << S << ") = { " << v << "};" << std::endl;
         S += 1;
         v += 1;
       }

       /*int fbr =dt + 9;
       file << "Volume(" << dt << ") = {";
       while (fbr <= (dt + 56 ))
       {
         fbr = fbr + 1;
         file << fbr << ", ";
       }
       file << "" << ( dt + 58 ) << "};" << std::endl;
       file << "Physical Volume(\"matrix\") = {" << dt << "};" << std::endl;*/

       /*{
         //
         int inclu = 0;
         file << "Physical Volume(\"inclusions\") = { ";
         while (inclu < (dt)-4)
         {
           inclu = inclu + 1;
           file << inclu << ", ";
         }
       }

       {
         file << "" << (dt -1) << " };" << std::endl;
       }*/
       file << endl;


         //file << "Surface Loop(124) = {51, 52, 57 , 58, 59 , 60};"<< std::endl;

       file << "Surface Loop(" << S << ") = { " << dt << ", " << dt + 1 << ", "<<
       dt + 6<<" , "<< dt + 7 <<", "<< dt + 8<<" , "<< dt + 9 <<"};" << std::endl;


         file << "Volume("<<r<<") = {"<<S<<"};"<< std::endl;

         //file << "Physical Volume(\"matrix\") = {64};" << std::endl;

      /** file << " BooleanIntersection{ Volume{64};} ";
       file <<" { Volume{1}; Volume{2}; Volume{3}; Volume{4}; Volume{5}; Volume{6};";
        file <<" Volume{7}; Volume{8}; Volume{9}; Volume{10}; Volume{11}; Volume{12}; Volume{13}; ";
         file << "Volume{15}; Volume{14}; Volume{16};Delete;}"<< std::endl;*/


       file << " BooleanIntersection{ Volume{"<<r<<"};} { ";
        double sp =1;
       for (j = 0; j < ((r - 2)); j++)
       {
         file << "Volume{" << sp << "};" ;
         sp += 1;
       }
       file << "Volume{" << r-1 << "};Delete;}"<< std::endl;

         //file << "Physical Volume(\"matrix\") = {61,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};"<< std::endl;
         //file<<"Volume(100) = {61,62,63,64,65, 66, 67, 68, 69, 70, 71, 72,73,74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109};"<<std::endl;
         //file << "Physical Volume(\"matrix\") = {100};"<< std::endl;


        // file << "Physical Volume(\"inclusions\")= {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};"<< std::endl;

         int inclu = 0;
         file << "Physical Volume(\"inclusions\") = { ";
         while (inclu < r-2)
         {
           inclu = inclu + 1;
           file << inclu << ", ";
         }


       {
         file << "" << (r-1) << " };" << std::endl;
       }


        file << endl;

        //file<<" Physical Volume(\"matrix\") =  BooleanDifference{ Volume{64};Delete;}  { Volume{1};";
         //file<<"Volume{2}; Volume{3}; Volume{4}; Volume{5}; Volume{6}; Volume{7}; Volume{8}; Volume{9}; Volume{10};";
         //file<<" Volume{11}; Volume{12}; Volume{13}; Volume{15}; Volume{14}; Volume{16};};"<<std::endl;


       file << " Physical Volume(\"matrix\") =  BooleanDifference{ Volume{"<<r <<"};Delete;} { ";
        double spi =1;
       for (j = 0; j < ((r - 2)); j++)
       {
       // Check if spi is within the specified ranges and is a multiple of 2
       if ((spi >= 49 && spi <= 65 && static_cast<int>(spi) % 2 == 0) ||
           (spi >= 113 && spi <= 129 && static_cast<int>(spi) % 2 == 0) ||
           (spi >= 177 && spi <= 192 && static_cast<int>(spi) % 2 == 0) ||
           (spi >= 242 && spi <= 256 && static_cast<int>(spi) % 2 == 0)) {
           spi += 1;
           continue; // Skip this iteration
           }
         file << "Volume{" << spi << "};" ;
         spi += 1;
       }
       file << "Volume{" << r-2 << "};};"<< std::endl;




       file << endl;
       // file<<"Mesh.Algorithm3D = 10;"<<std::endl;
        //file<<"Mesh.Algorithm = 5;"<<std::endl;
        file << " Mesh.CharacteristicLengthExtendFromBoundary =0;" << std::endl;
        //file << "Mesh.CharacteristicLengthFromCurvature = 0.8;" << std::endl;
        //file << "Mesh.CharacteristicLengthFromPoints =0.01;" << std::endl;
      //file << "Mesh.CharacteristicLengthFromEdges = 0;" << std::endl;
      //file << "Mesh.CharacteristicLengthFromFaces = 0;" << std::endl;
      //file << "Mesh.CharacteristicLengthFromVolumes = 0.1;" << std::endl;
      //file << "Mesh.ElementOrder = 2;" << std::endl;
      file << "Mesh.CharacteristicLengthFromCurvature = 15;" << std::endl; // 15
      //  file << "Mesh.CharacteristicLengthMin = 0.01;" << std::endl;
        //file << "Mesh.CharacteristicLengthMax = 1;" << std::endl;
        // file<<"Mesh.MeshSize{"<<r3<<"} = 0.2;"<<std::endl;
       // file<<"Mesh.MeshSizeFactor = 0.25; "<<std::endl;

       file<<" Field[1] = Box;" << std::endl;
         file<<" Field[1].VIn = " << 0.08<< ";" << std::endl; // mesh size inside the cylinder"
         file<<" Field[1].VOut =" << 0.08 << ";" << std::endl; // mesh size outside the cylinder
         file<<" Field[1].XMin = 0;" << std::endl;
         file<<" Field[1].XMax = " << nx << ";"<< std::endl;
         file<<" Field[1].YMin = 0;" << std::endl;
         file<<" Field[1].YMax = " << ny << ";"<< std::endl;
         file<<" Field[1].ZMin = 0;" << std::endl;
         file<<"  Field[1].ZMax = " << nz << ";" << std::endl;


        file<<" Background Field = 1;" << std::endl;



        file << " Mesh 3;" << std::endl;
        file << "Mesh.Smoothing = 100;" << std::endl;





    std::string commend = "output_"  + std::to_string(mshi) + ".msh";//.msh
    std::string cend = "gmsh output_" + std::to_string(mshi) + ".geo";//.msh

      //system("gmsh rve2ex104-mod.geo");

     // system(cend.c_str());




      std::string command = "gmsh " + filename + " -3 -smooth 100 > gmsh.log";
      int err = system(command.c_str());

        std::ofstream Output_file("king.sh", std::ios::app);
        if (Output_file.is_open())

     // Output_file<<"#!/bin/bash"<<endl;
     Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ") working..."<<"\""<<";";
      Output_file<<" ./mpap3" + commend + "> out/rve"+ to_string(num2) <<endl;
      Output_file<< " echo "<<"\"(" + to_string(num2); Output_file<< ")  all done.."<<"\""<<endl;
    num2+=1;
```

```


         std::ofstream youtput_file("kinghpc.sh", std::ios::app);
         if (Output_file.is_open())

      // Output_file<<"#!/bin/bash"<<endl;
       youtput_file<<"mpiexec ./mpap3 ~/fem2/mpap3/proj/rve/trial/"+ commend +
          " > out/test"+ to_string(mshi) <<endl;

     //num2+=1;



     mshi+=1;




    /*   mpapInit("solid", "test RVE");

        mos.setDoubleFormat("%7g");

        FemSolidRVE rve("rve", 3);
        rve.getMeshFromGmsh41File("output_1.msh", "solid3D");

       // rve.getMeshFromGmsh41File(args[1], "solid3D");

        //rve.mesh3D4to10noded("solid3D");

       rve.centreMesh();
        rve.x0 = rve.x;

#ifdef VTK

        rve.vtkGetUndeformedMesh();

        rve.vtkGetCellData("group");

       rve.vtkGetVectorData("displacement", 0);

        rve.vtkWriteOutputFile("rve.vtu");

#endif


     /* rve.set("linearBC");
        mos << " linear b.c.\n\n";
        // rve.set("periodicBC"); mos << " periodic b.c.\n\n";

        rve.set("ngp", 5);
        rve.set("tol", 1.e-5);
        rve.set("maxIter", 5);

        rve.set("Hooke");
        rve.set("updateCoorFlag", false);
        rve.set("3D");

        rve.listGroups();

        /*rve.set("K", 100.);
        rve.set("mu", 10.);

        rve.setGroupData("matrix", "K", 100.);
        rve.setGroupData("matrix", "mu", 10.);
        rve.setGroupData("inclusions", "K", 1000.);
        rve.setGroupData("inclusions", "mu", 100.);

        double eps[] = { 0.2,           0.1,           -0.1,           0.2,
                    0.1,           -0.1}, sig[10], cc[36],
                 F[] = {1.1, 0.1, 0.2, -0.1, 0.9, 0., 0.2, 0.1, 1.05};
        rve.stressUpdate(sig, cc, eps);
        mos << mos.array(eps, 6) << "\n\n";
        mos << mos.array(sig, 6) << "\n\n";

        #ifdef VTK

        rve.vtkGetUndeformedMesh();

        rve.vtkGetCellData("group");

       rve.vtkGetVectorData("displacement", 0);

        rve.vtkWriteOutputFile("rve.vtu1");

#endif

        double eps1[] = { 0.03,           0.02,           -0.004,           0.2,
                    0.3,           -0.5}, sig1[10];
        rve.stressUpdate(sig1, cc, eps1);
        mos << mos.array(eps1, 6) << "\n\n";
        mos << mos.array(sig1, 6) << "\n\n";

        double eps2[] = { 0.01,           0.03,           -0.2,           -0.05,
                    0.2,           -0.1}, sig2[10];
        rve.stressUpdate(sig2, cc, eps2);
        mos << mos.array(eps2, 6) << "\n\n";
        mos << mos.array(sig2, 6) << "\n\n";

        double eps3[] = { 0.01,           0.03,           -0.04,           0.1,
                    0.01,           -0.01}, sig3[10];
        rve.stressUpdate(sig3, cc, eps3);
        mos << mos.array(eps3, 6) << "\n\n";
        mos << mos.array(sig3, 6) << "\n\n";

        double eps4[] = { 0.3,           0.05,           -0.01,           0.2,
                    0.04,           -0.005}, sig4[10];
        rve.stressUpdate(sig4, cc, eps4);
        mos << mos.array(eps4, 6) << "\n\n";
        mos << mos.array(sig4, 6) << "\n\n"; */


 /*
#ifdef VTK

        rve.vtkGetUndeformedMesh();

        rve.vtkGetCellData("group");

       rve.vtkGetVectorData("displacement", 0);

        rve.vtkWriteOutputFile("rve.vtu");

#endif */
```

```



        }
      }
    }
    }



 output_file<<" n ="<< num -1 <<";" << std::endl;

 output_file<<"combined_vector = [];" << std::endl;
 output_file<<"combined_vector1 = [];" << std::endl;
 output_file<<"combined_vector2 = [];" << std::endl;
 output_file<<"combined_vector3 = [];" << std::endl;
 output_file<<"combined_vector4 = [];" << std::endl;

 output_file<<"for i = 1:n" << std::endl;
 output_file<<" var_name = ['spacing' num2str(i)];" << std::endl;
 output_file<<" var_name1 = ['F_volume_r' num2str(i)];" << std::endl;
 output_file<<" var_name2 = ['v_volume_r' num2str(i)];" << std::endl;
 output_file<<" var_name3 = ['radius_of_voids' num2str(i)];" << std::endl;
 output_file<<" var_name4 = ['radius_of_inclusion' num2str(i)];" << std::endl;

 /*output_file<<"    exist(var_name, 'var') == 1" << std::endl;
 output_file<<"    exist(var_name1, 'var') == 1" << std::endl;
 output_file<<"    exist(var_name2, 'var') == 1" << std::endl;
 output_file<<"    exist(var_name3, 'var') == 1" << std::endl;
 output_file<<"    exist(var_name4, 'var') == 1" << std::endl;*/

 output_file<<"combined_vector = [combined_vector; eval([var_name '(:)'])];" << std::endl;
 output_file<<"combined_vector1 = [combined_vector1; eval([var_name1 '(:)'])];" << std::endl;
 output_file<<"combined_vector2 = [combined_vector2; eval([var_name2 '(:)'])];" << std::endl;
 output_file<<"combined_vector3 = [combined_vector3; eval([var_name3 '(:)'])];" << std::endl;
 output_file<<"combined_vector4 = [combined_vector4; eval([var_name4 '(:)'])];" << std::endl;

 /*output_file<<"  else" << std::endl;
 output_file<<"  warning(['Variable ' var_name ' does not exist. Skipping...']);" << std::endl;
 output_file<<"  warning(['Variable1 ' var_name1 ' does not exist. Skipping...']);" << std::endl;
 output_file<<"  warning(['Variable2' var_name2 ' does not exist. Skipping...']);" << std::endl;
 output_file<<"  warning(['Variable3 ' var_name3 ' does not exist. Skipping...']);" << std::endl;
 output_file<<"  warning(['Variable4' var_name4 ' does not exist. Skipping...']);" << std::endl;*/

 //output_file<<"  end" << std::endl;
 output_file<<"end" << std::endl;
 output_file<<"spacing =[combined_vector];" << std::endl;
 output_file<<"F_volume_r =[combined_vector1];" << std::endl;
 output_file<<"v_volume_r =[combined_vector2];" << std::endl;
 output_file<<"radius_of_voids =[combined_vector3];" << std::endl;
 output_file<<"radius_of_inclusion =[combined_vector4];" << std::endl;

}
```

Listing C.6: Cube With Cylindrical Discontinuous inclusion Geometry File

Bibliography

[1] Ahmad Al-Maharma, Sandeep Patil, and Bernd Markert. Effects of porosity on the mechanical properties of additively manufactured components: a critical review. *Materials Research Express*, 7:122001, 12 2020.

[2] R. Barrett, M. Berry, T. F. Chan, et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1994.

[3] Mark Hudson Beale, Martin T. Hagan, and Howard B. Demuth. *Neural Network Toolbox^{TM} Getting Started Guide*. The MathWorks, Inc., Natick, MA, r2018a edition, 2018.

[4] Paolo Campigotto, Andrea Passerini, and Roberto Battiti. Active learning of pareto fronts. Technical Report DISI-13-001, DISI - Via Sommarive, 5 - 38123 POVO, Trento - Italy, 2013.

[5] Erasmo Carrera, Marco Petrolo, M. H. Nagaraj, and Michele Delicata. Evaluation of the influence of voids on 3d representative volume elements of fiber-reinforced polymer composites using cuf micromechanics. *Composite Structures*, 254:112833, 2020.

[6] CompositesLab. History of composites. URL `http://compositeslab.com/composites-101/history-of-composites/`. Accessed: 15-May-2024.

[7] Isaac M. Daniel and Ori Ishai. *Engineering Mechanics of Composite Materials*. Oxford University Press, New York, 2nd edition, 2006.

[8] E. A. de Souza Neto and R. A. Feijóo. Variational foundations of multi-scale constitutive models of solid: Small and large strain kinematical formulation. Internal Research and Development Report 16/2006, National Laboratory for Scientific Computing (LNCC), Brazil, 2006.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2), April 2002.

[10] Y. Dong, editor. *Manufacturing, Characterisation and Properties of Advanced Nanocomposites*. MDPI, 2018.

[11] Marzia Dulal, Md Rashedul Islam, Saptarshi Maiti, Mohammad Hamidul Islam, Iftikhar Ali, Amr M. Abdelkader, Kostya S. Novoselov, Shaila Afroj, and Nazmul Karim. Smart and multifunctional fiber-reinforced composites of 2d heterostructure-based textiles. *Advanced Functional Materials*, 33(40): 2305901, 2023.

[12] K. Essaadaoui, M. Ait El Fqih, M. Idiri, and B. Boubeker. Experimental investigation on reinforced concrete beams by honeycomb sandwich panel structures: mechanical properties study. In *International Conference on Advances in Energy Technologies, Environment*, volume 948 of *IOP Conference Series: Materials Science and Engineering*, page 012027, 2020.

[13] Christophe Geuzaine and Jean-François Remacle. *Gmsh Reference Manual*. The documentation for Gmsh 4.12.2 A finite element mesh generator with built-in pre- and post-processing facilities, January 2024.

[14] N. Ghadarah. A review on acoustic emission testing for structural health monitoring of composite materials. *Materials Today Communications*, 31:103724, 2023. doi: 10.1016/j.mtcomm.2022.103724. URL `https://doi.org/10.1016/j.mtcomm.2022.103724`. Additional authors not provided.

[15] Morteza Sasani (Ed.) Ghamsari. *Nanorods and Nanocomposites*. IntechOpen, London, England, 2020.

[16] 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.

[17] J. C. Halpin and J. L. Kardos. The halpin-tsai equations: A review. *Polymer Engineering and Science*, 16(5):344–352, May 1976.

[18] Abdulhammed K. Hamzat, Md Shafinur Murad, Ibrahim A. Adediran, Eylem Asmatulu, and Ramazan Asmatulu. Fiber-reinforced composites for aerospace, energy, and marine applications: an insight into failure mechanisms under chemical, thermal, oxidative, and mechanical load conditions. *Advanced Composites and Hybrid Materials*, 8(1):152, 2025. doi: 10.1007/s42114-024-01192-y. URL `https://doi.org/10.1007/s42114-024-01192-y`.

[19] Sahar Hassani, Mohsen Mousavi, and Amir H. Gandomi. Structural health monitoring in composite structures: A comprehensive review. *Sensors*, 22(1):153, 2022. doi: 10.3390/s22010153. URL `https://doi.org/10.3390/s22010153`. Published: 27 December 2021.

[20] Haowei Huang, S. Ali Hadigheh, and Keyvan Aghabalaei Baghaei. Influences of fibre shape on the transverse modulus of unidirectional fibre reinforced composites using finite element and machine learning methods. *Composite Structures*, 312:116872, 2023.

[21] David Hutton. *Fundamentals of Finite Element Analysis*. McGraw-Hill Education, New York, NY, 1st edition, 2004.

[22] M. Jawaid, H.P.S. Abdul Khalil, A. Abu Bakar, and P. Noorunnisa Khanam. Chemical resistance, void content and tensile properties of oil palm/jute fibre reinforced polymer hybrid composites. Technical report, School of Industrial Technology, Universiti Sains Malaysia, Penang, Malaysia, 2010.

[23] Wei Jiang, Zhigao Huang, Yunming Wang, Bing Zheng, and Huamin Zhou. Voids formation and their effects on mechanical properties in thermoformed carbon fiber fabric-reinforced composites. *Polymer Composites*, 2018.

[24] T. Johnson. History of composites: The evolution of lightweight composite materials, 2018. Accessed: 15-May-2024.

[25] Frank Leone, Bao Mosinyi, John Bakuckas, Jonathan Awerbuch, Alan Lau, and Tein-Min Tan. Structural testing and evaluation of honeycomb sandwich composite fuselage panels. pages 1–16, 04 2008.

[26] Yan Li, Qian Li, and Hao Ma. The voids formation mechanisms and their effects on the mechanical properties of flax fiber reinforced epoxy composites. *Applied Science and Manufacturing*, 76:1–9, 2015.

[27] C. Lv, Y. Xing, J. Zhang, X. Na, Y. Li, T. Liu, D. Cao, and F.-Y. Wang. Levenberg-marquardt backpropagation training of multilayer neural networks for state estimation of a safety critical cyber-physical system. *IEEE Transactions on Industrial Informatics*, 14(8):3–5, August 2018.

[28] Saptarshi Maiti, Md Rashedul Islam, Mohammad Abbas Uddin, Shaila Afroj, Stephen J. Eichhorn, and Nazmul Karim. Sustainable fiber-reinforced composites: A review. *Advanced Sustainable Systems*, 2022. doi: 10.1002/adsu.202200258. URL `https://doi.org/10.1002/adsu.202200258`. First published: 19 September 2022, Citations: 261.

[29] Inc. Mar-Bal. History of composite materials, n.d. URL `https://www.mar-bal.com/language/en/applications/history-of-composites/`. Accessed: 15-May-2024.

[30] MathWorks. gamultiobj function - matlab & simulink - mathworks, 2023a. URL `https://www.mathworks.com/help/gads/gamultiobj.html`.

[31] MathWorks. Lsqr: Solve system of linear equations — least-squares method. `https://uk.mathworks.com/help/matlab/ref/lsqr.html`, 2024.

[32] Mahoor Mehdikhani, Larissa Gorbatikh, Ignaas Verpoest, and Stepan V. Lomov. Voids in fiber-reinforced polymer composites: A review on their formation, characteristics, and effects on mechanical performance. *Journal of Composite Materials*, 53(12):1579–1669, 2019.

[33] A. de Souza Neto, D. Peric, and D. R. J. Owen. *Computational Methods for Plasticity: Theory and Applications*. John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 2008.

[34] Tri-Dung Ngo. Introduction to composite materials. *IntechOpen*, 2020.

[35] I Ocampo, RR López, S Camacho-León, V Nerguizian, and I Stiharu. Comparative evaluation of artificial neural networks and data analysis in predicting liposome size in a periodic disturbance micromixer. *Micromachines*, 12(10):1164, 2021.

[36] C. C. Paige and M. A. Saunders. LSQR: An Algorithm for Sparse Linear Equations And Sparse Least Squares. *ACM Transactions on Mathematical Software*, 8:43–71, 1982.

[37] Anand Patel and Gülenay Kilic. *A Review on Advanced Manufacturing Techniques and Their Applications*. 07 2021.

[38] D Peric, EA de Souza Neto, RA Feijóo, M Partovi, and AJ Carneiro Molina. On micro-to-macro transitions for multi-scale analysis of non-linear heterogeneous materials: unified variational basis and finite element implementation. *International Journal for Numerical Methods in Engineering*, 87:149–170, 2011.

[39] Zoran Petrovic and Ivan Lazarevic. Design and analysis of the flat honeycomb sandwich structures. *Scientific Technical Review*, 65:50–56, 01 2015.

[40] B. A. Praveena, N. Santhosh, Abdulrajak Buradi, H. V. Srikanth, G. Shankar, K. Ramesha, N. Manjunath, S. N. Karthik, M. Rudra Naik, and S. Praveen Kumar. Experimental investigation on density and volume fraction of void, and mechanical characteristics of areca nut leaf sheath fiber-reinforced polymer composites. *International Journal of Polymer Science*, 2022:13, 2022.

[41] Ramzyzan Ramly, Wahyu Kuntjoro, and Amir Radzi Ab Ghani. Honeycomb carbon sandwich composite panel under compressive load flat-wise: Experimental analysis. *Journal of Mechanical Engineering*, 17(3):13–26, 2020.

[42] R. Sathya and Annamma Abraham. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2 (2):35–36, 2013.

[43] Abdullah Sayam, A. N. M. Masudur Rahman, Md Sakibur Rahman, Shamima Akter Smriti, Faisal Ahmed, Md Fogla Rabbi, Mohammad Hossain, and Md Omar Faruque. A review on carbon fiber-reinforced hierarchical composites: mechanical performance, manufacturing process, structural applications and allied challenges. *Carbon Letters*, 32(5):1173–1205, 2022. ISSN 1976-4251. doi: 10.1007/s42823-022-00358-2. PMID: 40477668, PMCID: PMC9172091.

[44] Haykin Simon. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, Nj, 2009.

[45] Antonella Sola and Alireza Nouri. Microstructural porosity in additive manufacturing: The formation and detection of pores in metal parts fabricated by powder bed fusion. *Journal of Advanced Manufacturing and Processing*, 1(3):e10021.

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

[47] Patrick Terriault and Vladimir Brailovski. Modeling and simulation of large, conformal, porosity-graded and lightweight lattice structures made by additive manufacturing. *Finite Elements in Analysis and Design*, 138:1–11, 2018.

[48] D. Therriault and R.D. Farahani. Design of multifunctional composites and their advanced additive manufacturing. In *ADDFABCOMP– Additive Fabrication of Composite n°2*, Online conference, November 2021. Polytechnique Montreal, Laboratory for Multiscale Mechanics (LM2), Mechanical Engineering Department, Polytechnique Montreal.

[49] S. Venkatarajan, C. Subbu, A. Athijayamani, and R. Muthuraja. Mechanical properties of natural cellulose fibers reinforced polymer composites – 2015–2020: A review. *Composites Part B: Engineering*, 207:108613, 2021.

[50] M Vinod and K S Shivakumar Aradhya. *validation of halpin-tsai and nielson empirical relations for composite materials using finite element method*. 2005.

[51] Zhenyu Wang, Zhen Fang, Zhihong Xie, and Douglas E. Smith. A review on microstructural formations of discontinuous fiber-reinforced polymer composites prepared via material extrusion additive manufacturing: Fiber orientation, fiber attrition, and micro-voids distribution. *Polymers*, 14:4941, 2022.

[52] Dongmin Yang, Haoqi Zhang, Jiang Wu, and Edward D. McCarthy. Fibre flow and void formation in 3d printing of short-fibre reinforced thermoplastic composites: An experimental benchmark exercise. *Additive Manufacturing*, 36:101686, 2020.

[53] O. C. Zienkiewicz and R. L. Taylor. *The Finite Element Method*. McGraw-Hill, 1991.

[54] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4): 257, November 1999.