Implementation of the RVE boundary conditions for the textile composites

Zahur Ullah, Łukasz Kaczmarczyk and Chris J. Pearce

School of Engineering, Rankine Building, The University of Glasgow, Glasgow, UK, G12 8LT

1 Introduction

For the textile composites, representative volume element (RVE) boundary conditions are implemented using the generalised procedure given in [1]. In matrix form these boundary conditions are given as

$$\mathbf{Cu} = \mathbf{D}\bar{\boldsymbol{\varepsilon}} = \mathbf{g}, \tag{1}$$

where $\bar{\boldsymbol{\varepsilon}} = \begin{bmatrix} \bar{\varepsilon}_{xx} & \bar{\varepsilon}_{yy} & \bar{\varepsilon}_{zz} & 2\bar{\varepsilon}_{xy} & 2\bar{\varepsilon}_{yz} & 2\bar{\varepsilon}_{zx} \end{bmatrix}^T$ is the macro-strain associated with a Gauss point, $\mathbf{g}$ is the constraints applied to enforce the boundary conditions, while matrices $\mathbf{C}$ and $\mathbf{D}$ are given as

$$\mathbf{C} = \int_\Gamma \mathbf{H}\mathbf{N}^T\mathbf{N}d\Gamma, \qquad \mathbf{D} = \int_\Gamma \mathbf{H}\mathbf{N}^T\mathbf{X}d\Gamma. \tag{2}$$

In Equation (2), $\mathbf{N}$ is a matrix of shape functions, $\mathbf{X}$ is a matrix of Gauss point coordinates and $\mathbf{H}$ is a matrix specific to the type of boundary conditions used (e.g. displacement, periodic, traction). These matrices are further explained in the following sections. Integration in Equation (2) are performed over the boundary Γ of the RVE. The RVE boundary conditions are implemented in the University of Glasgow in-house computational tool MoFEM, which is based on the hierarchic finite element formulation [2] and allows us to use arbitrary order of approximation for the RVE simulation.

2 RVE displacement boundary conditions

For the displacement boundary condition case, the final system of linear equations is arranged as shown in Figure 1. In Figure 1, $\mathbf{K}$, $\mathbf{u}$ and $\mathbf{F}$ are standard stiffness matrix,

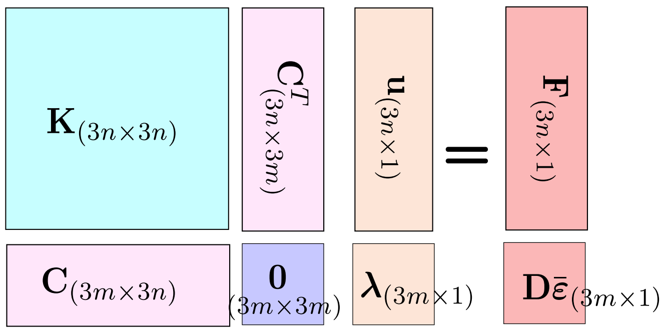

$\mathbf{K}_{(3n\times3n)}$
$\mathbf{C}^T_{(3n\times3m)}$
$\mathbf{u}_{(3n\times1)}$
=
$\mathbf{F}_{(3n\times1)}$
$\mathbf{C}_{(3m\times3n)}$
$\mathbf{0}_{(3m\times3m)}$
$\boldsymbol{\lambda}_{(3m\times1)}$
$\mathbf{D}\bar{\boldsymbol{\varepsilon}}_{(3m\times1)}$

Figure 1: Final matrix and vector arrangements for displacement boundary conditions

displacement and force vector respectively, where $3n$ are the total number of RVE degrees of freedom, including high order degrees of freedom associated with edges, faces and volumes explained below in this section or $3n$ is the total number of nodes multiplied by the number of degrees of freedom in each cartesian direction. This will depend on the order of the approximation. Additional matrices and vectors appears in Figure 1 is due to the implementation of the RVE displacement boundary conditions. Matrix $\mathbf{0}$ is zeros matrix and matrices $\mathbf{C}$ and $\mathbf{D}$ as already mentioned in Equation (2) are calculated and assembled, while integrating over the boundary Γ of the RVE. Unknown parameters $\boldsymbol{\lambda}$ are Lagrange multipliers and $3m$ used in the sizes of these additional vectors and matrices are the degrees of freedoms associated with the boundary (including degrees of freedoms associated with edges and faces) of the RVE.

A sample RVE geometry and corresponding mesh are shown in Figure 2(a) and Figure 2(b) respectively, which consist of fibres and matrix. The RVE is discretised with tetrahedral elements, which leads to triangular elements over its boundary. These triangular elements are subsequently used to perform integration in Equations (2).

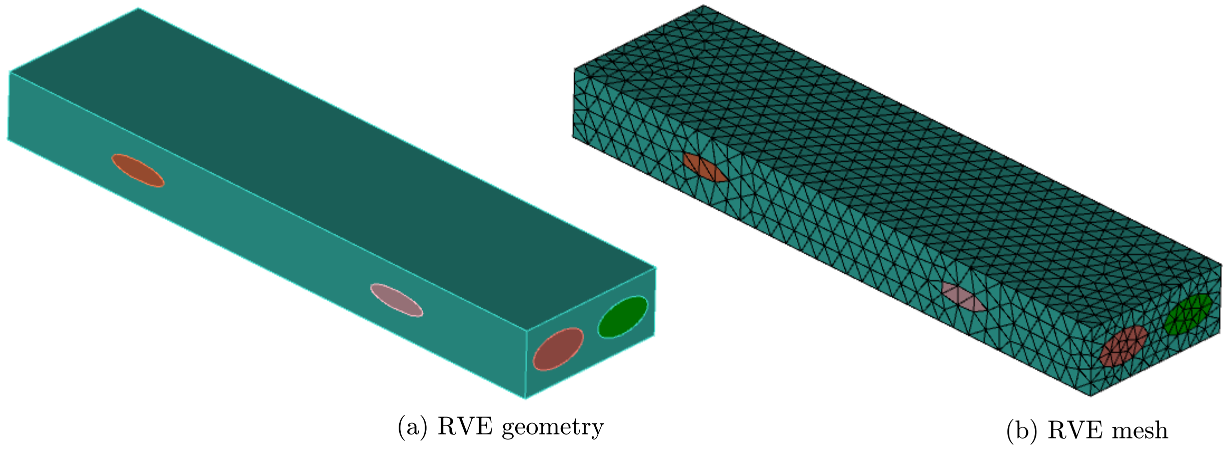

(a) RVE geometry (b) RVE mesh

Figure 2: Sampel RVE geometry and corresponding mesh

For the 1st order hierarchic finite element or $(p = 1)$ (in which approximation functions are associated only with nodes), matrix $\mathbf{C}$ is (9×9) for each triangular element on the RVE boundary, i.e.

$$\mathbf{C}_{(9\times9)} = \int_{\Gamma} \mathbf{H}_{(9\times9)} \mathbf{N}^T_{(9\times3)} \mathbf{N}_{(3\times9)} d\Gamma, \tag{3}$$

In the displacement boundary condition case, matrix $\mathbf{H}$ is an identity matrix and for every Gauss point, the shape function matrix $\mathbf{N}$ is written as

$$\mathbf{N} = \begin{bmatrix} N1 & 0 & 0 & N2 & 0 & 0 & N3 & 0 & 0 \\ 0 & N1 & 0 & 0 & N2 & 0 & 0 & N3 & 0 \\ 0 & 0 & N1 & 0 & 0 & N2 & 0 & 0 & N3 \end{bmatrix}. \tag{4}$$

Furthermore, for the 1st order or $(p = 1)$ coordinate matrix $\mathbf{D}$ is (9×6) and is written as

$$\mathbf{D}_{(9\times6)} = \int_{\Gamma} \mathbf{H}_{(9\times9)} \mathbf{N}^T_{(9\times3)} \mathbf{X}_{(3\times6)} d\Gamma, \tag{5}$$

where $\mathbf{X}$ is the Gauss point coordinate matrix and is written as

$$\mathbf{X} = \frac{1}{2}\begin{bmatrix} 2x & 0 & 0 & y & 0 & z \\ 0 & 2y & 0 & x & z & 0 \\ 0 & 0 & 2z & 0 & y & x \end{bmatrix}, \tag{6}$$

where x, y and z are the cartesian coordinates evaluated at the integration points during numerical integration they are found using the known nodal coordinates and shape functions associated with these nodes, using

$$\begin{bmatrix} x_1 & \cdots & x_{ng} \\ y_1 & \cdots & y_{ng} \\ z_1 & \cdots & z_{ng} \end{bmatrix} = \begin{bmatrix} x_1^{nd} & x_2^{nd} & x_3^{nd} \\ y_1^{nd} & y_2^{nd} & y_3^{nd} \\ z_1^{nd} & z_2^{nd} & z_3^{nd} \end{bmatrix} \begin{bmatrix} N_1 & \cdots & N_{ng} \\ N_2 & \cdots & N_{ng} \\ N_3 & \cdots & N_{ng} \end{bmatrix}, \tag{7}$$

where ng are the total number of Gauss points used in each triangular element on the boundary to perform numerical integration and x^{nd}, y^{nd} and z^{nd} are nodal coordinates.

For higher orders of approximation or $(p > 1)$, additional degrees of freedoms are associated with edges, faces and volume for a general three dimensional finite element [2]. In case of two-dimensional elements, these additional degrees of freedoms are associated with its edges and faces. The exact number of degrees of freedoms associated with each edge, face and volume for high order of approximations $(p > 1)$ is shown in Table 1, in which $p \in \mathbb{N}$ is active order of approximation and $\mathbb{N}$ is set of natural number.

Space	Vertices	Edges	Faces	Volume
H_1	1	$p-1$	$(p-2)(p-1)/2$	$(p-3)(p-2)(p-1)/6$

Table 1: Degrees of freedoms associated with hierarchic finite element

Furthermore for each edge (e.g. triangular element), for 2nd and 3rd order of approximation, i.e. $(p = 2)$ and $(p = 3)$ matrix $\mathbf{C}$ is written as

$$\mathbf{C}_{(3\times3)} = \int_\Gamma \mathbf{H}_{(3\times3)}\mathbf{N}^T_{(3\times3)}\mathbf{N}_{(3\times3)}d\Gamma, \qquad p = 2, \tag{8a}$$

$$\mathbf{C}_{(6\times6)} = \int_\Gamma \mathbf{H}_{(6\times6)}\mathbf{N}^T_{(6\times3)}\mathbf{N}_{(3\times6)}d\Gamma, \qquad p = 3, \tag{8b}$$

$$\tag{8c}$$

while matrix $\mathbf{D}$ is written as

$$\mathbf{D}_{(3\times6)} = \int_\Gamma \mathbf{H}_{(3\times3)}\mathbf{N}^T_{(3\times3)}\mathbf{X}_{(3\times6)}d\Gamma, \qquad p = 2, \tag{9a}$$

$$\mathbf{D}_{(6\times6)} = \int_\Gamma \mathbf{H}_{(6\times6)}\mathbf{N}^T_{(6\times3)}\mathbf{X}_{(3\times6)}d\Gamma, \qquad p = 3. \tag{9b}$$

$$\tag{9c}$$

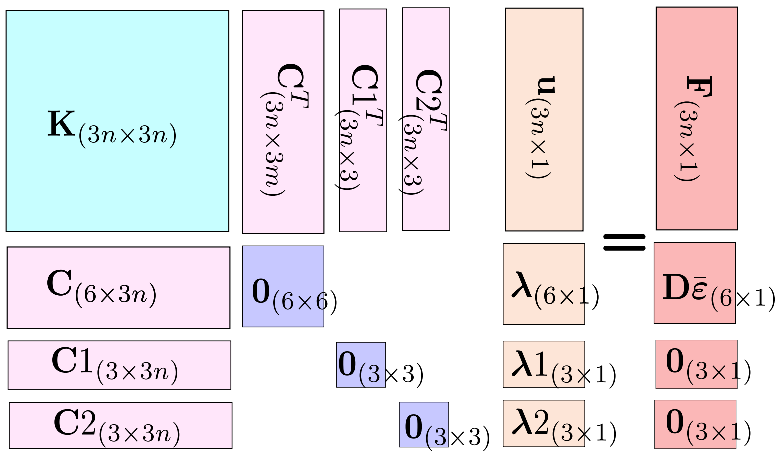
In Equations 8 and 9, matrices $\mathbf{H}_{(3\times3)}$ and $\mathbf{H}_{(6\times6)}$ are (3×3) and (6×6) identity matrices, while shape functions matrices $\mathbf{N}_{(3\times3)}$ and $\mathbf{N}_{(3\times6)}$ are written as

$$\mathbf{N}_{(3\times3)} = \begin{bmatrix} N1 & 0 & 0 \\ 0 & N1 & 0 \\ 0 & 0 & N1 \end{bmatrix} \qquad p=2, \tag{10a}$$

$$\mathbf{N}_{(3\times6)} = \begin{bmatrix} N1 & 0 & 0 & N2 & 0 & 0 \\ 0 & N1 & 0 & 0 & N2 & 0 \\ 0 & 0 & N1 & 0 & 0 & N2 \end{bmatrix} \qquad p=3, \tag{10b}$$

3 RVE traction boundary conditions

In case of traction boundary conditions, the final system of linear equations is shown in Figure 3, where additional matrices $\mathbf{C}$, $\mathbf{D}$ and additional vector of unknown Lagrange multipliers $\boldsymbol{\lambda}$ are due to the imposition of RVE boundary condition. Furthermore, matrix



$\mathbf{K}_{(3n\times3n)}$
$\mathbf{C}^T_{(3n\times3m)}$
$\mathbf{C1}^T_{(3n\times3)}$
$\mathbf{C2}^T_{(3n\times3)}$
$\mathbf{u}_{(3n\times1)}$
$\mathbf{F}_{(3n\times1)}$
$=$
$\mathbf{C}_{(6\times3n)}$
$\mathbf{0}_{(6\times6)}$
$\boldsymbol{\lambda}_{(6\times1)}$
$\mathbf{D}\bar{\boldsymbol{\varepsilon}}_{(6\times1)}$
$\mathbf{C1}_{(3\times3n)}$
$\mathbf{0}_{(3\times3)}$
$\boldsymbol{\lambda}1_{(3\times1)}$
$\mathbf{0}_{(3\times1)}$
$\mathbf{C2}_{(3\times3n)}$
$\mathbf{0}_{(3\times3)}$
$\boldsymbol{\lambda}2_{(3\times1)}$
$\mathbf{0}_{(3\times1)}$

Figure 3: Final matrix and vector arrangements for traction boundary conditions

$\mathbf{C1}$ and corresponding vector of unknown Lagrange multipliers $\boldsymbol{\lambda}1$ are used to restrict the rigid body translations in x, y and z directions, while matrix $\mathbf{C2}$ and corresponding vector of unknown Lagrange multipliers $\boldsymbol{\lambda}2$ are used to constrains the rigid body rotations about x, y and z axis. In this case, for 1^{st} order or $p=1$, matrices $\mathbf{C}$ and $\mathbf{D}$ are written as

$$\mathbf{C}_{(6\times9)} = \int_\Gamma \mathbf{H}_{(6\times9)}\mathbf{N}^T_{(9\times3)}\mathbf{N}_{(3\times9)}d\Gamma, \qquad \mathbf{D}_{(6\times6)} = \int_\Gamma \mathbf{H}_{(6\times9)}\mathbf{N}^T_{(9\times3)}\mathbf{X}_{(3\times6)}d\Gamma. \tag{11}$$

The number of row of matrix $\mathbf{C}$ in this case are always 6 independent of the degrees of freedoms used on the boundary. In traction boundary condition case all matrices appeared in Equation (11) are the same as explained in the previous sections with only exception of matrix $\mathbf{H}$, which is different for each side of the RVE. For negative x-face of the RVE matrix $\mathbf{H}$ is written as

$$\mathbf{H}_x^- = \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}, \tag{12}$$

where its each row represents admissible distribution of nodal traction, i.e. $\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \sigma_{xy} & \sigma_{xz} & \sigma_{zx} \end{bmatrix}^T$ as shown in Figure 4 and 5 and its every adjacent three columns belong to a nodes of a triangular element. For remaining negative y-face

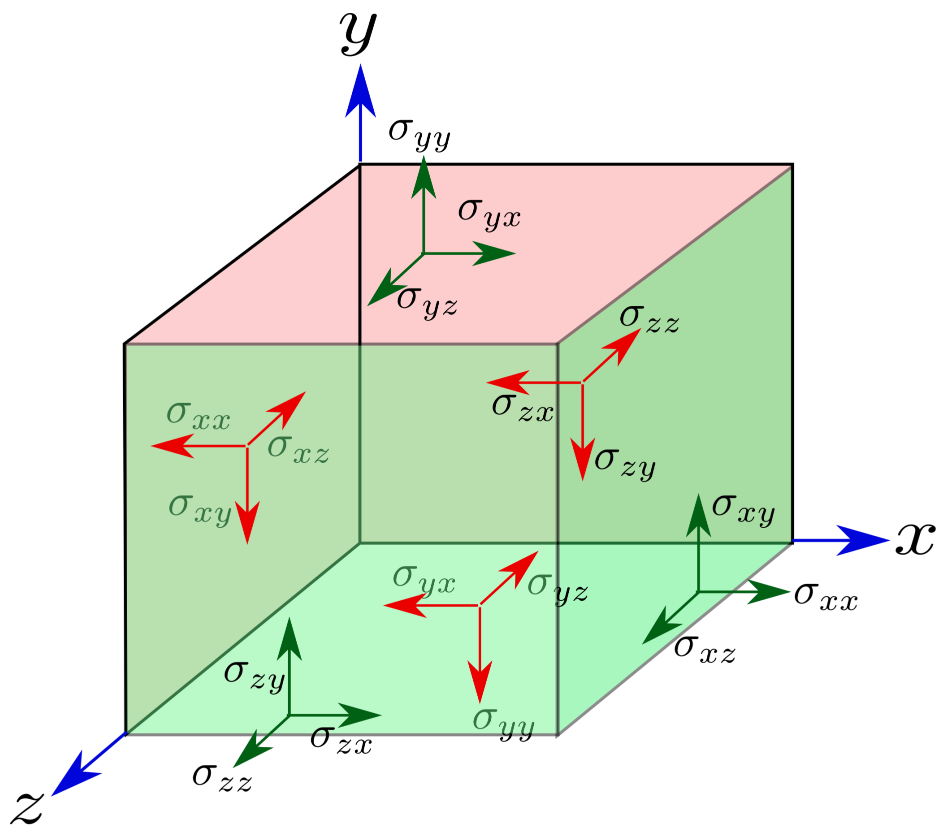


Figure 4: Three-dimensional state of stress

$$\begin{array}{c|ccc|ccc|ccc|}
 & x & y & z & x & y & z & x & y & z \\
\sigma_{xx} & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\
\sigma_{yy} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\sigma_{zz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\sigma_{xy} & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\
\sigma_{yz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\sigma_{zx} & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \\
 & \multicolumn{3}{c}{\text{Node 1}} & \multicolumn{3}{c}{\text{Node 2}} & \multicolumn{3}{c}{\text{Node 3}}
\end{array}$$

Figure 5: H-matrix for the RVE negative x-face

and negative z-face of the RVE matrix $\mathbf{H}$ is written as

$$\mathbf{H}_y^- = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{13}$$

$$\mathbf{H}_z^- = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}. \tag{14}$$

For the three positive faces, i.e. x, y and z matrix $\mathbf{H}$ is written as

$$\mathbf{H}_x^+ = -\mathbf{H}_x^-, \qquad \mathbf{H}_y^+ = -\mathbf{H}_y^-, \qquad \mathbf{H}_z^+ = -\mathbf{H}_z^- \tag{15}$$

where $\mathbf{H}_x^-$, $\mathbf{H}_y^-$ and $\mathbf{H}_z^-$ are the $\mathbf{H}$ matrices associated with the negative x, y and z faces of the RVE, while $\mathbf{H}_x^+$, $\mathbf{H}_y^+$ and $\mathbf{H}_z^+$ are the $\mathbf{H}$ matrices associated with positive x, y and z faces of the RVE.

Furthermore, in case of high order of approximation, e.g. 2^{nd} and 3^{rd} order, i.e. $p = 2$ and $p = 3$ for each edge belongs to the negative x-face, matrix $\mathbf{H}$ is written as

$$\mathbf{H}_{\mathbf{x}}^- = \begin{cases} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, & p = 2 \\ \begin{bmatrix} -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, & p = 3 \end{cases} \tag{16}$$

Matrix $\mathbf{C}1$ associated with constraining rigid body translations is shown Figures 6, which is constructed using equations

$$u_{1x} + u_{2x} + u_{3x} + \cdots u_{mx} = 0, \tag{17a}$$
$$u_{1y} + u_{2y} + u_{3y} + \cdots u_{my} = 0. \tag{17b}$$
$$u_{1z} + u_{2z} + u_{3z} + \cdots u_{mz} = 0. \tag{17c}$$

Furthermore, matrix $\mathbf{C}2$, associated with constraining rigid body rotations is shown in

$$\begin{array}{c|ccccccccccc|} & x & y & z & x & y & z & & x & y & z \\ x & 1 & 0 & 0 & 1 & 0 & 0 & \cdots & 1 & 0 & 0 \\ y & 0 & 1 & 0 & 0 & 1 & 0 & \cdots & 0 & 1 & 0 \\ z & 0 & 0 & 1 & 0 & 0 & 1 & \cdots & 0 & 0 & 1 \\ & \multicolumn{3}{c}{\text{Node 1}} & \multicolumn{3}{c}{\text{Node 2}} & & \multicolumn{3}{c}{\text{Node m}} \end{array}$$

Figure 6: Constraint matrix $\mathbf{C}1$ for rigid body translations

Figure 7, which is constructed using spin tensor [3] as

$$\text{Spin}(\mathbf{x}) = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{18}$$

$$\begin{array}{c|ccccccccccc|} & x & y & z & x & y & z & & x & y & z \\ x & 0 & -z & y & 0 & -z & y & \cdots & 0 & -z & y \\ y & z & 0 & -x & z & 0 & -x & \cdots & z & 0 & -x \\ z & -y & x & 0 & -y & x & 0 & \cdots & -y & x & 0 \\ & \multicolumn{3}{c}{\text{Node 1}} & \multicolumn{3}{c}{\text{Node 2}} & & \multicolumn{3}{c}{\text{Node m}} \end{array}$$

Figure 7: Constraint matrix **C**2 for rigid body rotations

4 RVE periodic boundary conditions

In this case the final system of linear equation is shown in Figure 8, which is the same as shown in Figure 3 for traction boundary condition case but the number of rows of matrix $\mathbf{C}$, vectors $\boldsymbol{\lambda}$ and $\mathbf{D}\bar{\boldsymbol{\varepsilon}}$ are $\frac{3m}{2}$. In this case a matching pair of nodes on opposite faces of the RVE (e.g. nodes on positive and negative x-face with the same y and z coordinates) will contribute to same row in matrix $\mathbf{C}$ and vector $\mathbf{D}\bar{\boldsymbol{\varepsilon}}$. Therefore, only one Lagrange multiplier λ is associated with such a pair of nodes. Due to this requirement of matching nodes on negative and positive faces, the negative sides of the RVE are meshed first, which is then copied to the positive faces of the RVE. Triangular prisms are then inserted between the matching triangles on opposite sides of the RVE, which facilitates easy handling of data during calculation and assembly of matrix $\mathbf{C}$ and vector $\mathbf{D}\bar{\boldsymbol{\varepsilon}}$. A sample triangular prism generated between element A on negative x-face and element B on positive x-face of the RVE is shown in Figure 9. In this case numerical integration is performed over these inserted triangular prisms and matrices $\mathbf{C}$ and $\mathbf{D}$ are calculated and assembled. The integration procedure in this case is the same as already explained in the displacement boundary conditions case in Equations 3 and 5 with the exception of matrix $\mathbf{H}$ and the additional contribution to the same rows for matching pairs of nodes belongs to positive and negative faces of the RVE. For this case matrix $\mathbf{H}$ is the (9×9) identity matrix for negative x, y and z faces of the RVE, while for its positive x, y and z faces it is given as

$$\mathbf{H}^{+} = -\mathbf{H}^{-}, \tag{19}$$

where $\mathbf{H}^{+}$ and $\mathbf{H}^{-}$ are matrix $\mathbf{H}$ for positive and negative faces of the RVE respectively.

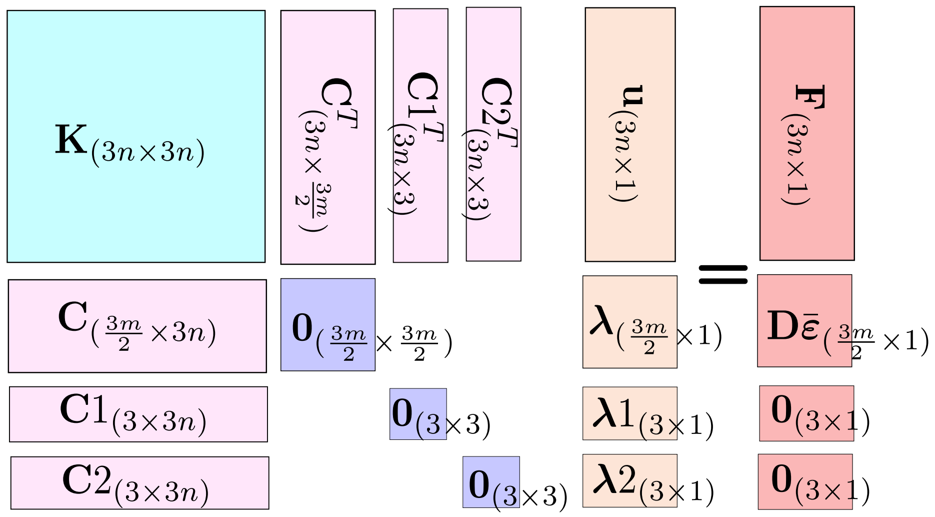

$\mathbf{K}_{(3n\times3n)}$
$\mathbf{C}^T_{(3n\times\frac{3m}{2})}$
$\mathbf{C1}^T_{(3n\times3)}$
$\mathbf{C2}^T_{(3n\times3)}$
$\mathbf{u}_{(3n\times1)}$
=
$\mathbf{F}_{(3n\times1)}$
$\mathbf{C}_{(\frac{3m}{2}\times3n)}$
$\mathbf{0}_{(\frac{3m}{2}\times\frac{3m}{2})}$
$\boldsymbol{\lambda}_{(\frac{3m}{2}\times1)}$
$\mathbf{D}\bar{\boldsymbol{\varepsilon}}_{(\frac{3m}{2}\times1)}$
$\mathbf{C1}_{(3\times3n)}$
$\mathbf{0}_{(3\times3)}$
$\boldsymbol{\lambda}1_{(3\times1)}$
$\mathbf{0}_{(3\times1)}$
$\mathbf{C2}_{(3\times3n)}$
$\mathbf{0}_{(3\times3)}$
$\boldsymbol{\lambda}2_{(3\times1)}$
$\mathbf{0}_{(3\times1)}$

Figure 8: Final matrix and vector arrangements for displacement boundary conditions

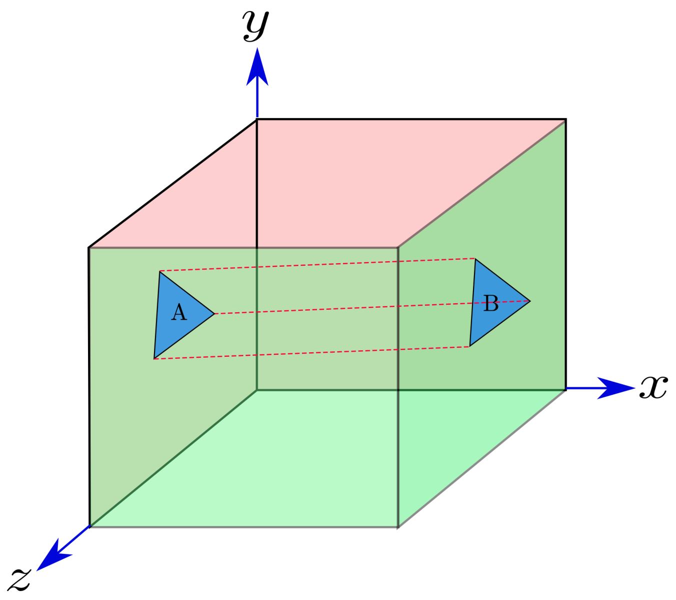


Figure 9: Inserted prism between positive and negative x-axis

5 Calculation of homogenised stress

Expression for homogenised stress $\overline{\boldsymbol{\sigma}} = \left[\begin{array}{cccccc} \overline{\sigma}_{xx} & \overline{\sigma}_{yy} & \overline{\sigma}_{zz} & \overline{\sigma}_{xy} & \overline{\sigma}_{yz} & \overline{\sigma}_{zx} \end{array} \right]^T$ is given as [1]

$$\overline{\boldsymbol{\sigma}} = \frac{1}{V}\mathbf{D}^T\boldsymbol{\lambda}, \tag{20}$$

where $\mathbf{D}$ is the same matrix, explained in the previous sections, V is the volume of the RVE and $\boldsymbol{\lambda}$ are the Lagrange multipliers obtained after solution of the final system of equations shown in Figures 1, 3 and 8 for the displacement, traction and periodic boundary condition cases. The contribution to the homogenised stress $\overline{\boldsymbol{\sigma}}$ is calculated individually for each boundary triangular element separately, which are then assembled to calculate the full RVE homogenised stress. For displacement boundary conditions and 1^{st} order or $p = 1$, for each individual triangular element of the RVE Equation (20) is written as

$$\overline{\boldsymbol{\sigma}}_{(6\times1)} = \frac{1}{V}\mathbf{D}^T_{(6\times9)}\boldsymbol{\lambda}_{(9\times1)} \tag{21}$$

for 2^{nd} and 3^{rd} order, i.e. $p = 2$ and $p = 3$ for each edge of the triangular element Equation 20 is written as

$$\begin{array}{ll} \overline{\boldsymbol{\sigma}}_{(6\times1)} = \frac{1}{V}\mathbf{D}^T_{(6\times3)}\boldsymbol{\lambda}_{(3\times1)}, & p = 2 \\ \overline{\boldsymbol{\sigma}}_{(6\times1)} = \frac{1}{V}\mathbf{D}^T_{(6\times6)}\boldsymbol{\lambda}_{(6\times1)}, & p = 3 \end{array}. \tag{22}$$

6 RVE input file for MoFEM

Geometry of the textile RVE, including fibres and matrix is generated using CUBIT which is a software package for the creation of parameterised geometries and meshes. CUBIT is linked with MoFEM, which can automatically read geometry, boundary conditions and material properties from a file created in CUBIT. A sample RVE used here, including fibres and matrix is shown in Figure 10, in which elliptical cross sections and cubic splines are used respectively to model the cross sections and paths of the fibre. Transversely isotropic materials are introduced for the fibres, while isotropic material is used for the matrix part. A self-explanatory python script is used here to create the textile RVE geometry, meshing and subsequently defining its material properties are given in Figure 11. The python script gives us a flexible and automatic way to create an input textile RVE file for MoFEM, i.e. a new textile RVE with all new material and geometry parameters can be generated without manually repeating all the time consuming and error-prone steps.

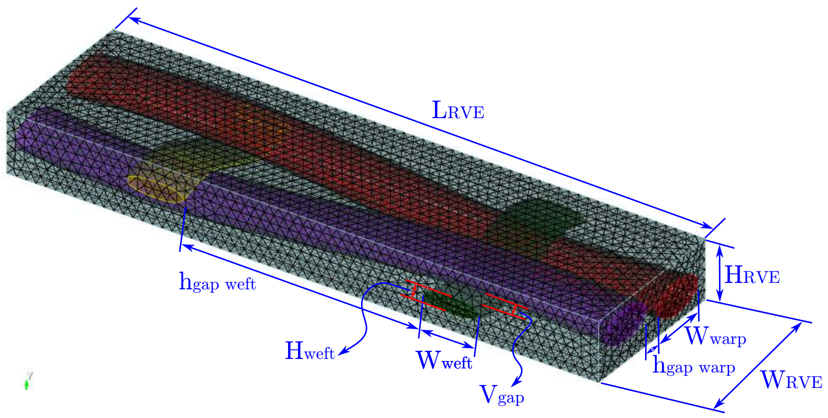


Figure 10: RVE geometry with dimensions

7 RVE material properties and stiffness matrix assembly

Stiffness matrix **K**, shown in Figures 1, 3 and 8 for displacement, traction and periodic boundary conditions respectively consists of contributions from three different parts, i.e. matrix, fibres and interfaces between fibres and matrix. Transversely isotropic material [4, 5] are used for the fibres while isotropic material is assumed for the matrix part. Five material parameters are required for the transversely isotropic material, i.e. E_p, ν_p, E_z, ν_{pz} and G_{zp} where E_p and ν_p are Young's modulus and Poisson's ratio in the transverse direction respectively, while E_z, ν_{pz} and G_{zp} are the Young's modulus, Poisson's ratio and shear modulus in the fibre directions respectively. For the matrix part, only two material parameters are required, i.e. Young's modulus E and Poisson's ratio ν. Material properties for elastic and transversely isotopic material (including plane of isotropy and principal axis) are shown in Figure 12. Stiffness matrix for the transversely isotropic material in the local coordinates is given as [6]

$$\mathbf{C} = \begin{bmatrix} \frac{1-\nu_{pz}\nu_{zp}}{E_pE_z\Delta} & \frac{\nu_p+\nu_{zp}\nu_{pz}}{E_pE_z\Delta} & \frac{\nu_{zp}+\nu_p\nu_{zp}}{E_pE_z\Delta} & 0 & 0 & 0 \\ \frac{\nu_p+\nu_{zp}\nu_{pz}}{E_pE_z\Delta} & \frac{1-\nu_{pz}\nu_{zp}}{E_pE_z\Delta} & \frac{\nu_{zp}+\nu_{zp}\nu_p}{E_pE_z\Delta} & 0 & 0 & 0 \\ \frac{\nu_{pz}+\nu_p\nu_{pz}}{E_p^2\Delta} & \frac{\nu_{pz}(1+\nu_p)}{E_p^2\Delta} & \frac{1-\nu_p^2}{E_p^2\Delta} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2G_{zp} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2G_{zp} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{E_p}{1+\nu_p} \end{bmatrix}, \tag{23}$$

where

$$\Delta = \frac{(1+\nu_p)(1-\nu_p-2\nu_{pz}\nu_{zp})}{E_p^2 E_z}, \qquad \frac{\nu_{pz}}{E_p} = \frac{\nu_{zp}}{E_z}. \tag{24}$$

The fibre directions at each Gauss point need to be determined, in order to re-orient the stiffness matrix in local coordinates given in Equation 23 into global coordinates. These fibres directions can simply be determined from the cubic splines, which were used in the construction of these fibres. However, this can lead to inaccurate fibres directions in the case of fibres with non-uniform cross-sections along their length. An alternative and unusual way is used here, in which the fibres directions are determined by solving for the potential flow field separately along each fibre. The governing equation for potential flow is given as

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0, \tag{25}$$

where ϕ is the stream function, gradient of which, i.e. $\nabla\phi = \mathbf{v}$ subsequently determines the fibre direction. The potential flow problem is also implemented in MoFEM using hierarchic finite element [2], which enables to found very accurate fibres direction s to be determined for very coarse meshes. A detailed description of how to transfer the stiffness matrix for transversely isotropic material between local and global coordinate axis is given in [4, 5]. An example of the output from a potential flow analysis for the textile is shown in Figure 13.

Detailed theoretical background and implementation of the interface elements, which is used here between the fibres and matrix part is given in [7].

References

[1] Ł. Kaczmarczyk, C. J. Pearce, and N. Bićanić. Scale transition and enforcement of RVE boundary conditions in second-order computational homogenization. *International Journal for Numerical Methods in Engineering*, 74(3):506–522, 2008.

[2] M. Ainsworth and J. Coyle. Hierarchic finite element bases on unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 58(14):2103–2130, 2003.

[3] Ł. Kaczmarczyk, T. Koziara, and C. J. Pearce. Corotational formulation for 3D solids. an analysis of geometrically nonlinear foam deformation. arXiv preprint arXiv:1110.5321, 2011.

[4] Z. Ullah, Ł. Kaczmarczyk, M. Cortis, and C. J. Pearce. Multiscale modelling of the textile composite materials. In *Proceedings of the 22nd UK Conference of the Association for Computational Mechanics in Engineering 2 - 4 April 2014, University of Exeter, Exeter*, pages 214–217, 2014.

[5] M. Cortis, Ł. Kaczmarczyk, and C. J. Pearce. Computational modelling of braided fibre for concrete reinforcement. In *Proceedings of the 22nd UK Conference of the Association for Computational Mechanics in Engineering 2 - 4 April 2014, University of Exeter, Exeter*, pages 189–192, 2014.

[6] Transversely isotropic materials. Available at: eFunda: The ultimate online reference for engineers. (Accessed date: 26 May 2014).

[7] J. Segurado and J. LLorca. A new three-dimensional interface finite element to simulate fracture in composites. *International Journal of Solids and Structures*, 41(11–12):2977 – 2993, 2004.

(a) Part 1 of 3

```
#This scrip will create RVE including f bres and matrix, which can be used for all
#three type of boundary conditions including displacement, traction and periodic.

cubit.cmd('new')
#===============================================================
#Geometry Input parameters
#===============================================================
W_wrap=0.3; H_wrap=0.1514; hgap_wrap=0.09
W_weft=0.3; H_weft=0.0757; hgap_weft=1.2
L_RVE=3.0;    W_RVE=0.8;        H_RVE=0.3;
Vgap=0.012

s_weft=hgap_weft+W_weft;    s_wraf=hgap_wrap+W_wrap
T=(H_weft/2+H_wrap/2)/2+Vgap

L_RVE=2*s_weft;    W_RVE=2*s_wraf;     H_RVE=0.3;

#===============================================================
#Create vertices for splines to create splines
#===============================================================

coordx=-11.5*s_weft;  coordy=T; coordz=0;
for i in range(0, 24):
    if i % 2 == 0:
        str1='create vertex ' + str(coordx) +' '+str(coordy)+' '+str(coordz)+' color' ; coordx=coordx+s_weft;
        cubit.cmd(str1)
    if i % 2 == 1:
        str1='create vertex ' + str(coordx) +' '+str(-coordy)+' '+str(coordz)+' color' ; coordx=coordx+s_weft;
        cubit.cmd(str1)

coordx=-11.5*s_weft;  coordy=-T; coordz=s_wraf;
for i in range(0, 24):
    if i % 2 == 0:
        str1='create vertex ' + str(coordx) +' '+str(coordy)+' '+str(coordz)+' color' ; coordx=coordx+s_weft;
        cubit.cmd(str1)
    if i % 2 == 1:
        str1='create vertex ' + str(coordx) +' '+str(-coordy)+' '+str(coordz)+' color' ; coordx=coordx+s_weft;
        cubit.cmd(str1)

coordx=-0.5*s_weft;  coordy=-T; coordz=-11*s_wraf;
for i in range(0, 24):
    if i % 2 == 0:
        str1='create vertex ' + str(coordx) +' '+str(coordy)+' '+str(coordz)+' color' ; coordz=coordz+s_wraf;
        cubit.cmd(str1)
    if i % 2 == 1:
        str1='create vertex ' + str(coordx) +' '+str(-coordy)+' '+str(coordz)+' color' ; coordz=coordz+s_wraf;
        cubit.cmd(str1)

coordx=0.5*s_weft;  coordy=T; coordz=-11*s_wraf;
for i in range(0, 24):
    if i % 2 == 0:
        str1='create vertex ' + str(coordx) +' '+str(coordy)+' '+str(coordz)+' color' ; coordz=coordz+s_wraf;
        cubit.cmd(str1)
    if i % 2 == 1:
        str1='create vertex ' + str(coordx) +' '+str(-coordy)+' '+str(coordz)+' color' ; coordz=coordz+s_wraf;
        cubit.cmd(str1)

#===============================================================
#Joint vertices to create splines
#===============================================================

Tvertices=24;
vertices=range(1,Tvertices+1); count1=0;
for i in range(0, 4):
    str1='create curve spline vertex '
    for j in range(0, 24):
        str1=str1+' '+ str(count1+vertices[j])
    cubit.cmd(str1);  count1=count1+24;

#===============================================================
#Creating f bers with elliptical cross sections
#===============================================================

cubit.cmd('create planar surface with plane normal to curve 1 distance 0 from vertex 1')
str1='create curve location vertex 1 direction curve 8 length '+ str(W_wrap/2); cubit.cmd(str1)
str1='create curve location vertex 1 direction curve 7 length  '+str(H_wrap/2); cubit.cmd(str1)
cubit.cmd('create surface ellipse vertex 102 104 1')
cubit.cmd('sweep surface 2 along curve 1')

cubit.cmd('create planar surface with plane normal to curve 2 distance 0 from vertex 25')
str1='create curve location vertex 25 direction curve 17 length '+ str(W_wrap/2); cubit.cmd(str1)
str1='create curve location vertex 25 direction curve 16 length  '+str(H_wrap/2); cubit.cmd(str1)
cubit.cmd('create surface ellipse vertex 112 114 25')
cubit.cmd('sweep surface 6 along curve 2')

cubit.cmd('create planar surface with plane normal to curve 3 distance 0 from vertex 49')
str1='create curve location vertex 49 direction curve 24 length '+ str(W_weft/2); cubit.cmd(str1)
```

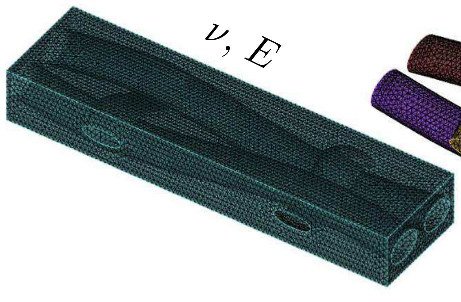
(b) Part 2 of 3

```
str1='create curve location vertex 49 direction curve 23 length  '+str(H_weft/2); cubit.cmd(str1)
cubit.cmd('create surface ellipse vertex 122 124 49')
cubit.cmd('sweep surface 10 along curve 3')

cubit.cmd('create planar surface with plane normal to curve 4 distance 0 from vertex 73')
str1='create curve location vertex 73 direction curve 33 length '+ str(W_weft/2); cubit.cmd(str1)
str1='create curve location vertex 73 direction curve 32 length  '+str(H_weft/2); cubit.cmd(str1)
cubit.cmd('create surface ellipse vertex 132 134 73')
cubit.cmd('sweep surface 14 along curve 4')

#==============================================================
#Deletion of remaining free vertices, curves and bodies
#==============================================================

cubit.cmd('delete vertex 50 20 53 52 69 36 81 65 54 67 17 22 63 70 16 42 12 93 10 2 37 84 85 43 31 6 71 \
13 55 78 95 9 90 56 57 15 94 66 83 44 3 8 79 76 39 5 19 27 35 18 47 7 92 21 62 82 4 64 88 68 89 80 \
23 14 33 41 26 38 59 75 34 74 91 77 11 60 58 86 87 32 61 30 51 40 29 28 46 45')
cubit.cmd('delete curve 1 3 36 9 19 27 10 2 4 28 37 18')
cubit.cmd('delete body 7 3 5 1')

#==============================================================
#Create matrix and imprint and merge f bres and matrix to create common surfaces
#==============================================================

str1='brick x '+str(L_RVE)+' y '+str(H_RVE)+' z '+str(W_RVE); cubit.cmd(str1)
str1='move Volume '+str(9)+' x '+str(0)+' y '+str(0)+' z '+str(s_wraf/2); cubit.cmd(str1)

cubit.cmd('intersect volume all keep')
cubit.cmd('delete volume 6 2 8 4')
cubit.cmd('subtract volume 10 11 12 13 from volume 9 keep')
cubit.cmd('delete volume 9')
cubit.cmd('imprint volume 10 11 12 13 14')
cubit.cmd('merge volume 10 11 12 13 14')

#==============================================================
#Meshing negative x, y and z face of RVE and coping the same to its positive faces
#==============================================================

cubit.cmd('group 1 add surface 42 27 24')
cubit.cmd('group "g2" add surface 44 25 28')
cubit.cmd('group "g3" add surface 39 30 33')
cubit.cmd('group "g4" add surface 31 40 34')
cubit.cmd('group "g5" add surface 43')
cubit.cmd('group "g6" add surface 41')

cubit.cmd('surface 42 27 24 size auto factor 6')
cubit.cmd('surface 42 27 24 scheme trimesh')
cubit.cmd('mesh surface 42 27 24')
cubit.cmd('surface 28 25 44 scheme mirror source surface 27 24 42 source vertex 167 target vertex 166 nosmoothing')
cubit.cmd('mesh surface 28 25 44')

cubit.cmd('surface 39 30 33 size auto factor 5')
cubit.cmd('surface 39 30 33 scheme trimesh')
cubit.cmd('mesh surface 39 30 33')
cubit.cmd('surface 31 34 40 scheme mirror source surface 30 33 39 source vertex 161 target vertex 166 nosmoothing')
cubit.cmd('mesh surface 31 34 40')

cubit.cmd('surface 43 scheme trimesh')
cubit.cmd('mesh surface 43')
cubit.cmd('surface 41 scheme mirror source surface 43 source vertex 165 target vertex 166 nosmoothing')
cubit.cmd('mesh surface 41')

#==============================================================
#Mesh volume
#==============================================================

cubit.cmd('volume all scheme Tetmesh')
cubit.cmd('mesh volume all')

#==============================================================
#Def ning blocks for elastic, transversely-isotropic and potential f ow problems
#==============================================================

vol=['14', '10,11,12,13','10', '11', '12', '13']
mat=['MAT_ELASTIC_1','MAT_TRANSISO_1','PotentialFlow1','PotentialFlow2','PotentialFlow3','PotentialFlow4']
for i in range(0, 6):
    cubit.cmd('set duplicate block elements on')
    str1='block  ' + str(i+1) +' volume '+vol[i]; cubit.cmd(str1)
    str1='block  ' + str(i+1) +' name "'+mat[i] + '"'; cubit.cmd(str1)

#==============================================================
#Material properties for matrix part
#==============================================================

cubit.cmd('block 1 attribute count 2')
Em=3.5e3; Enu=0.35;  #gig to mega as we used dimension in mm
Elastic=[str(Em), str(Enu)]
```

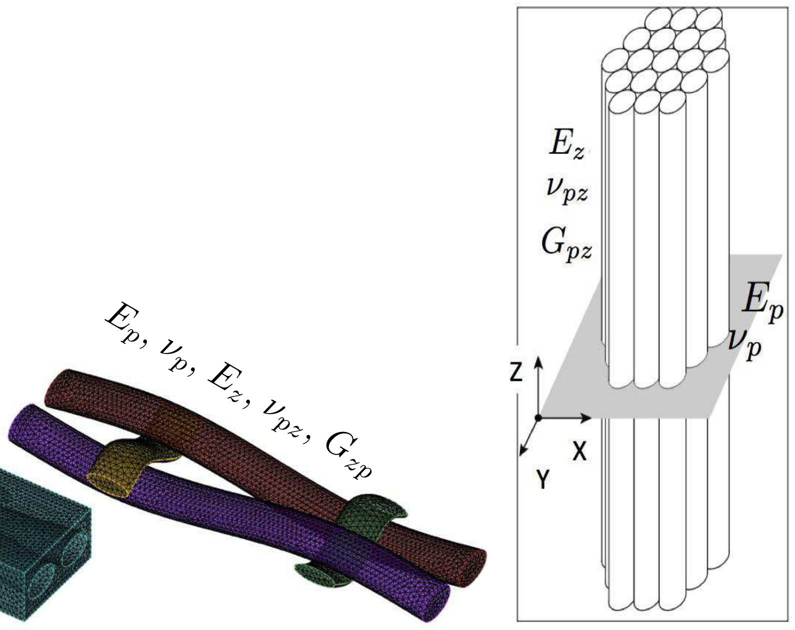
(c) Part 3 of 3

```
for i in range(0, 2):
    str1='block 1 attribute index ' + str(i+1) +' '+Elastic[i]; cubit.cmd(str1)

#==============================================================
#Material properties for f bres
#==============================================================

cubit.cmd('block 2 attribute count 5')
Ep=40e3; Ez=230e3; nup=0.26; nupz=0.26; Gzp=24e3;

TransIso=[str(Ep), str(Ez), str(nup), str(nupz), str(Gzp)]
for i in range(0, 5):
    str1='block 2 attribute index ' + str(i+1) +' '+TransIso[i]; cubit.cmd(str1)

#==============================================================
#Material properties for interface between f bres and matrix
#==============================================================

alpha_interf=500
cubit.cmd('set duplicate block elements on')
str1='block 7 surface 23 26 29 32'; cubit.cmd(str1)
str1='block 7 name "MAT_INTERF_1"'; cubit.cmd(str1)
cubit.cmd('block 7 attribute count 4')
str1='block 7 attribute index 1 '+str(alpha_interf); cubit.cmd(str1)    #now we use 4 parameters for interface
str1='block 7 attribute index 2 '+str(0.0); cubit.cmd(str1)
str1='block 7 attribute index 3 '+str(0.0); cubit.cmd(str1)
str1='block 7 attribute index 4 '+str(0.0); cubit.cmd(str1)

#==============================================================
#Def ning interfaces
#==============================================================

Interface=['23', '26','29','32']
for i in range(0, 4):
    str1='sideset ' + str(i+1) +' surface '+Interface[i]; cubit.cmd(str1)
    str1='sideset ' + str(i+1) +' name "interface'+str(i+1); cubit.cmd(str1)

#==============================================================
#Def ning pressures for potential f ow problem
#==============================================================

Pres=['24', '25','27','28','30','31','33','34']; count=0
for i in range(0, 4):
    str1='create pressure '+str(count+1)+' on surface '+str(Pres[count])+' magnitude 1';  cubit.cmd(str1)
    str1='create pressure '+str(count+2)+' on surface '+str(Pres[count+1])+' magnitude -1';  cubit.cmd(str1)
    count=count+2;

#==============================================================
#Def ning surfaces for displacement, traction and periodic boundary conditions
#==============================================================

cubit.cmd('sideset 101 surface 41 31 34 40 42 27 24')  # all -ve boundary surfaces for periodic boundary conditions
cubit.cmd('sideset 102 surface 43 30 33 39 25 44 28')  # all +ve boundary surfaces  for periodic boundary conditions
cubit.cmd('sideset 103 surface 41 31 34 40 42 27 24 43 30 33 39 25 44 28')  # all boundary surfaces

#==============================================================
#Def ning zero pressures for potential f ow problem
#==============================================================

cubit.cmd('nodeset 1 node 106, 140, 272, 279')
cubit.cmd('nodeset 1 name "ZeroPressure"')

#==============================================================
#Saving input RVE f le
#==============================================================

cubit.cmd('save as "/Users/zahur/Desktop/RVE_Meshes/Trans_RVE/RVE.cub" overwrite')
```

Figure 11: Python script for generation of RVE geometry with all input parameters



(a) Elastic



(b) Transversely isotropic

Figure 12: Material properties for elastic and transversely isotropic material

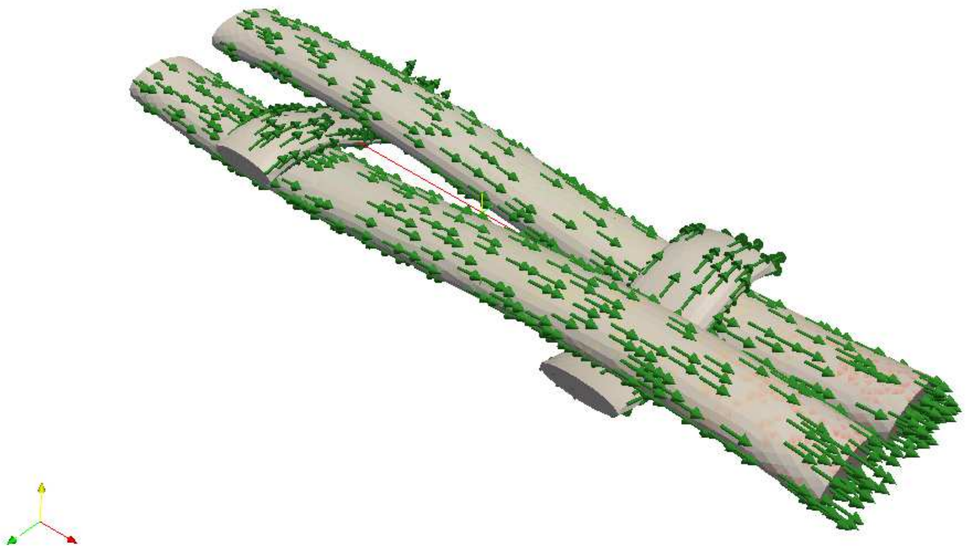


Figure 13: Fibres directions calculated from potential flow problem