AUTOMATED GENERATION OF EQUATION-BASED BOUNDARY CONDITIONS FOR MULTISCALE MODELLING OF UNIT CELLS

L.F.C. Jeanmeure[1]*, S. Li.[1]
[1] Department of Mechanical, Materials and Manufacturing Engineering,
The University of Nottingham, Nottingham, England
* Corresponding author (Laurent.Jeanmeure@nottingham.ac.uk)

Keywords: *unit-cell, composites, boundary conditions, ABAQUS scripting*

1 Background

Due to their nature, composites are multi-scale materials, the material properties of interest are those seen at the engineering part scale, yet their structure in turn spans several other scales. This is especially true of textile composites [1]. Assuming the engineering part level to represent the macro-scale; the scale at which the internal structure of that part, such as fibre orientation and volume fraction of the component yarns, is defined as the meso-scale. Further modelling refinements lead to the concept of a micro-scale where the fibre arrangement comprising the yarns is of interest. It should be noted that in this paper we assume that the concept of material continuum applies in the three scales introduced above. However, fibre mechanical properties used at the micro-scale depend on an even more refined scale at the limit of molecular assembly, which precludes the standard material continuum assumption. This nano-scale is not considered in this study.

2 Theory

2.1 Unit-cell and translational symmetries

To bridge the various scales mentioned above, the concept of Representative Volume Element (RVE) in the form of unit-cells can be introduced. The definition of the term unit-cell is based on the use of geometric symmetries from which boundary conditions (BCs) can be derived rationally. References [2-9] provide an overview of the importance of boundary conditions set-up for the field of unit-cell theory. In this paper the terms RVE and unit-cell will be used interchangeably, although the definition of a unit-cell calls for the use of all possible symmetries so as to limit its size. The term RVE tends to be used at the meso-scale to denote the basic repeating pattern of a textile composite for instance. Examples of unit-cell modelling focusing on the implementation of theoretical boundary conditions can be found in [3, 4, 6, 8]; in these references, as in this paper, a commercial FEM development suite has been used: Abaqus/CAE [11]. The choice of a commercial solution versus a custom made FEM code is twofold. Firstly, benchmarking of the code is not necessary and secondly the approach follows preferred industry standards. The drawbacks of a commercial product are its limitations and in the present case the absence of an option for the set-up of periodic boundary conditions. However, thanks to the modularity of Abaqus, it is possible to create Python scripts to impose these BCs as additional constraint equations. The main breakthrough presented in this paper is the automated set-up of the theoretical periodic BCs irrespective of geometry or mesh density. Prior to this, the coding had to be done manually [2-5], thus representing a serious overhead for modelling purposes especially for studies of variable volume fractions – thus changes in geometry calling for newly defined mesh-dependent BCs. The process still requires user attention, namely in the choice of unit-cell and proper understanding of symmetries both geometrical and load related, references [2-4, 6, 8] provide the adequate guidance.
In order to strike a balance between the use of the smallest possible modelling sample and the complexity of the implementation of boundary conditions, only translational symmetries are considered in the work presented here. Reflexional and rotational symmetries are therefore not taken into account.

2.2 Displacement fields for RVEs

Assuming material continuum (RVE small compared to the scale of the engineering part) and the use of translational symmetries alone, stresses

and strains are transformed identically from one unit-cell to another anywhere in the infinite replication in all the coordinate system directions of the model. The system of equations (1) represents the relationship between macroscopic strains and relative displacements at a point P of a reference unit-cell and *P'* its image in another cell.

$$\begin{aligned} u' - u &= (x' - x)\varepsilon_x^0 + (y' - y)\gamma_{xy}^0 + (z' - z)\gamma_{zx}^0 \\ v' - v &= (y' - y)\varepsilon_y^0 + (z' - z)\gamma_{yz}^0 \\ u' - u &= (z' - z)\varepsilon_z^0 \end{aligned} \quad (1)$$

Where x, y, z and u, v, w are respectively the coordinates of and the displacements at *P*; the prime equivalent being associated with *P'*. $\varepsilon_x^0, \varepsilon_y^0, \varepsilon_z^0$ and $\gamma_{xy}^0, \gamma_{yz}^0, \gamma_{zx}^0$ are the macroscopic strains and shear strains. Constraining $u = v = w = 0$, $\frac{\partial w}{\partial x} = \frac{\partial v}{\partial x} = \frac{\partial w}{\partial y} = 0$ at *P* taken as the origin, $x = y = z = 0$, eliminates the rigid body motion.

2.3 Boundary condition set-up

The nomenclature of the unit-cell is defined as a collection of mutually exclusive sets of vertices, edges and faces. Each of these sets is tied with a set of equations derived from (1) to account for the translational symmetries between neighbouring cells. These boundary conditions (BCs) equations are listed in [4] for various types of Voronoi cells. The reason why mutually exclusive sets are considered comes from the geometric limitations of a standard unit-cell representation: edges are associated with two faces and vertices in turn are associated with three edges – or more than three, depending on the sectioning of the model for mesh quality purposes. Therefore redundant constraint may arise and may result in, as is the case with Abaqus, the inclusion of redundant BCs in turn leading either to erroneous results or even fatal errors. In practice, and as is the case here, this problem is avoided by defining faces excluding edges and edges excluding vertices. Separate sets of BCs are then applied to edges and vertices [4].

3 Model implementation

The unit-cell theory presented above introduces equation-based boundary-conditions that are not part of the usual toolset of commercial Finite Element Method (FEM) packages. Their implementation is made all the more difficult by concerns related to mesh quality and more specifically node-to-node correspondence when it comes to translational symmetries. Models based on this approach to implement these BCs have already been published [2-5] but they rely on painstaking manual handling of geometry dependent node sets. The application presented here automates the model geometry generation, material allocation (isotropic, transverse isotropic or orthotropic) as well as the implementation of the equation-based BCs for standard unit-cell types: uni-directional (UD) square and hexagon shape, or simple cubic packing as shown on Figure 1. The automated constraint setting relies on node tagging and equation generation. The program is written in Python for Abaqus/CAE. Also in development is the extension of the paradigm to RVEs (meso-scale) for complex textile composites using voxel meshes generated in TexGen. No post-processing is necessary as material properties, such as the full set of Lamé constants and coefficient of thermal expansion (CTE), are given as standard output as well as various stress distributions corresponding to all typical case loads. Figure 2 displays the inner partitioning for a cubic centred unit-cell that ensures good mesh quality. Figure 3 is an example of an RVE of a complex textile composite created with TexGen [10]. TexGen can export a voxel meshing representation that in turn is used as input for a modified code to handle voxel mesh RVE modelling at the meso-scale.

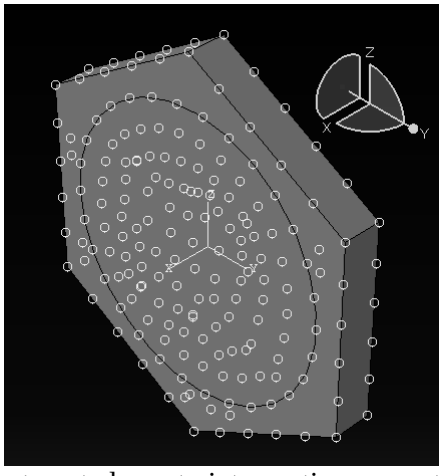

Fig.1. Automated constraint equation generation at all boundary nodes in a UD hexagonal unit-cell. Exact node-to-node connections are enforced.

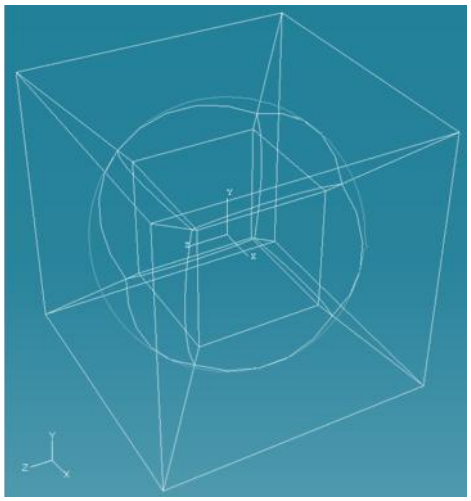

Fig.2. Partitioning of simple cubic packing unit-cell to enhance mesh quality and ensure node-to-node connection on sides and edges.

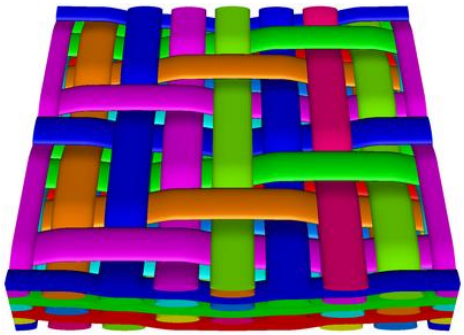

Fig.3. TexGen representation of complex textile composite RVE. Meso-scale modelling.

3.1 Mesh quality and BCs set-up

Once the geometry has been generated, the edges are seeded according to user input. Due to the design of Abaqus, seeding is only performed on edges (edges can be straight lines, circles or other closed curves). In the present case where the FEM discretisation is tightly linked with the imposition of the node-to-node BC set-up, the node-to-node correspondence must be imposed from one face to its opposite. Note that to avoid local interpolation effects within elements, the tessellation of the mesh on opposing faces must also be the same. In the simple cases presented here, and also thanks to the use of simple brick elements, these problems do not arise. To force Abaqus to generate a well proportioned mesh, the Python script forces a seed density on the fibre component commensurate to that of the external edges. Figure 4 shows the effect of this meshing constraint in the hexagonal case. This become particularly important when high fibre volume fractions are concerned as the mesh-engine capabilities related to element shape quality could impair the desired node-to-node correspondence and element tessellation.

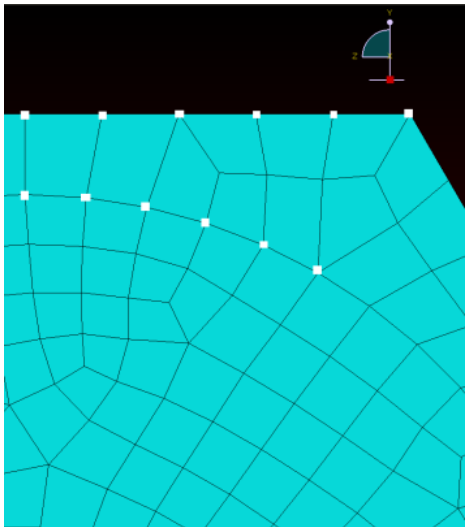

Fig.4. White dots show the imposed seeding for well proportioned mesh enforcement.

It should be noted that in the case shown on Figure 3 of meso-scale RVEs, the seeding issue is not considered because the application currently makes use of the TexGen voxel mesh export option. Although more limited than a free mesh approach, voxel meshes automatically enforce by default the node-to-node correspondence and tessellation requirements.

3.2 Constraint-equation handling in Abaqus/CAE

Abaqus solves an equation constraint by eliminating the first term on the left hand-side of the considered equation. The BC equation set-up must therefore be done in such a way that no degree of freedom appears as the first left hand-side term in any two equations. In the Python code implementation this translates to making sure that for a pair of corresponding node sets another node set has to be eliminated. Reference [4] shows this approach in details for various geometries. This approach is taken for all – mutually exclusive – faces, edges and vertices. A subtle difference of implementation must

be noted when considering the use of Abaqus/CAE or Abaqus "Keyword edition". Abaqus/CAE tends to handle additional constraint equations based on its Multiple Point Constraint (MPC) paradigm; i.e. it allows only one node as a reference point. This leads to a "many to one" approach for setting constraints between node sets. This is extremely limiting when employing a Python script to generate the model as only node pairs can be considered. In effect the current Python script must tag each external node (all of them in the simple one layer UD fibre composites presented here) for their inclusion in an equation. In the case of voxel-mesh TexGen export, the expression "external node" encompasses all the mesh nodes on the outside faces as well as edges and vertices. In the case of Abaqus input files (.INP in the keyword edition) it is possible to have a "many to many" approach as long as the tags (such as "Face A", "Face B", etc) correspond to ordered sets. This is at the same time an advantage (fewer equation set-up operation) and a drawback (the user must check manually the node ordering). At present, the programming effort concentrates on the Abaqus/CAE approach because of the ease of programming offered by Python scripting and the object-oriented approach of the Abaqus/CAE model tree. Direct .INP (keyword) file programming is currently being used for TexGen voxel-mesh export models because a voxel-mesh provides by default ordered node sets. As far as FEM algorithm complexity is concerned, either approach (CAE vs. "Keyword") gives the same FEM implementation (matrix), therefore asks for the same computer time requirements. The node-to-node correspondence and tagging is checked by looping through opposite node sets (face to face, edge to edge, etc) and testing for position within Abaqus geometrical tolerance (1e-6 of unit length). It must be noted that Abaqus does not handle "exact" coordinate values but always random ones within its geometric tolerance (based on the ACIS system [11]), hence the need to check for location prior to name-tagging. Tag names for the external nodes are automatically generated according to their node set type (face, edge, vertex) and their index in their set. Once these actions have been taken, the BC equation sets can be generated.

3.3 Simulation and result exploitation

Taking advantage of the Python scripting approach, the application also generates automatically the remaining components of an Abaqus model. Namely, the generation of steps, output requests, job definition and job posting are all handled without the need of further user interaction. Output database files (.ODB) are generated then in turn exploited by the script to produce a list of readily available material properties such as: macroscopic unit-cell equivalent Young Moduli, Poisson ratios, shear moduli and also coefficients of thermal expansions (CTEs). Figure 5 provides a flowchart of the Abaqus/CAE Python scripting application.

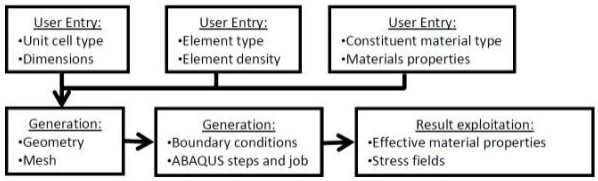

Fig.5. Abaqus/CAE Python script flowchart

4 Results

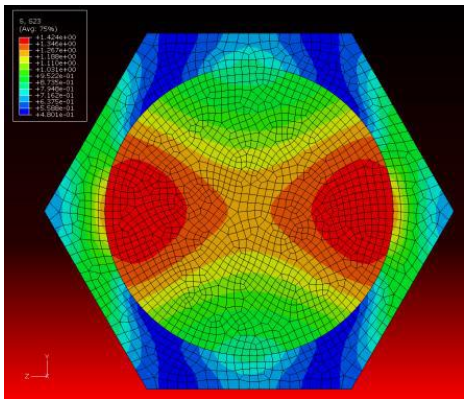

Fig.6. Stress distribution for a y-z shear load. Unit-cell at 60% fibre volume-fraction.

As expressed in the introductory section the novelty of this work is the automated implementation of what is commonly known as periodic boundary conditions. The approach itself has been demonstrated by time-consuming manual handling of Abaqus .INP files [2-6], which in turn was an approach therefore limited to fixed geometries. Figure 6 shows a simulation result with a relatively high density mesh (made possible by automation) corresponding to similar already published results [2-6]. The example given for a shear stress load proves the robustness of the proposed approach of using a "true" theoretical BC set-up. Shear loading is

usually the load-case where an "intuitive" boundary condition set-up leads to non-symmetrical stress distributions or other "simulation artifact".

5 Conclusions

The authors believe that the use of industry standard commercial applications has more advantages than drawbacks; thus they take the approach of adapting existing tools to more complex modelling operations that the tool is not design to handle but can nonetheless handle by selective use of capabilities and associated scripting efforts. The work presented here corresponds to very simple, and somewhat well known, unit-cell geometries. Current and future developments [12] are extending towards accommodating the more complex model-design requirements associated with textile composites. At present a halfway solution has been applied with the use of voxel-mesh exports and dedicated .INP file handling (within TexGen). The aim of the current development is to integrate TexGen within Abaqus/CAE Python scripts so as to exploit fully the implementation of theoretical BCs at the meso-scale (textile composite RVEs). The development of the application presented here has highlighted several requirements for further design as well as desirable developments for the evolution of commercial FEM solutions that the authors would like to share:

- Automated handling of redundant equation constraints in Abaqus/Standard (already present in Abaqus/Explicit to simplify the imposition of BC set-up.
- Identical mesh tessellation is required on opposed faces in a unit-cell model.
- Generalised "many-to-many" assignment for ordered sets.
- Extension to anticlastic problem in generalised plane strain (UD-composites) to allow the use of 2D elements instead of one layer of 3D bricks as in the current case.

Acknowledgements

The authors would like to acknowledge the support from EPSRC through platform grant EP/F02911X/1)

References

[1] S.V. Lomov, et al. "Meso-FE modelling of textile composites: Road map, data flow and algorithms". *Composite Science and Technology*, Vol. 67, pp 1870-1891, 2007.

[2] S. Li "On the unit cell for micromechanical analysis of fibre-reinforced composites". *Proc. Roy. Soc. Lond. A*, Vol 455, pp 815-838, 1999.

[3] S. Li "General unit cells for micromechanical analyses of unidirectional composites". *Composites A*, Vol 32, pp 815-826, 2001.

[4] S. Li, A. Wongsto "Unit-cells for micro-mechanical analyses of particle-reinforced composites". *Mech.Mater.*, Vol. 36, pp 543-572, 2004.

[5] A. Wongsto and S. Li "Micromechanical FE analysis of UD fibre-reinforced composites". *Composites A*, Vol 36, pp 1246-1266, 2005.

[6] S. Li "Boundary conditions for unit-cells from periodic micro-structures and their implications". *Composite Science and Technology*, Vol. 68, pp 1962-1974, 2008.

[7] S. Li, C. Zhou, H. Yu and L. Li "Formulation of a unit cell of a reduced size for plain weave textile composites". *Comp. Mater. Sci.*, Vol. 50, pp 1770-1780, 2011

[8] S. Li, N. Warrior, Z. Zou amd F. Almaskari "A unit-cell for FE analysis of materials with the microstructure of a staggered pattern". *Composites A*, Vol. 42, pp 801-811, 2011

[9] S. Li "On the nature of periodic traction boundary conditions in micromechanical FE analyses of unit-cells". *IMA J. Appl. Maths*. Accepted for publication.

[10] TexGen webpage: www.texgen.sourceforge.net

[11] SIMULIA, Abaqus CAE 6.10 User's Manual, 2010

[12] L.F.C. Jeanmeure and S. Li "Automated micro-scale unit-cell modeling in Abaqus/CAE". *Proceedings of Deformation and Fracture Composites (DFC11) & Structural Integrity and Multi-Scale Modelling (SI-5)*, Cambridge, England, April 12th-15th, 2011