Chapter title: TexGen

Author: Louise P. Brown[1]

[1] Composites Research Group, Faculty of Engineering, University of Nottingham, Nottingham, UK

louise.brown@nottingham.ac.uk

Abstract

This chapter gives an overview and introduction to the use of TexGen, open source software developed at the University of Nottingham as a pre-processor for 3D geometric modelling of textile structures. An overview is given of the modelling theory used in the software. There is a guide to creating automatically generated textile models using the built-in weave wizards and a detailed example of the method for creating a textile model using the functionality within the graphical user interface (GUI). An overview of the Python application programming interface (API) is given, illustrated by an example script, as well as information on how to use the Python functions to edit existing textiles. Finally there is an overview of the options for different meshing and export options used to prepare models as input for simulations.

Keywords: TexGen software, textile modelling, Python scripting, meshing

2.9.1. Introduction

TexGen (L.P.Brown and Sherburn, 2019) is open source software developed at the University of Nottingham for 3D modelling of textiles and textile composites. It is used as a modelling pre-processor for simulation of a variety of material properties including textile mechanical properties and permeability (Zeng et al., 2014).

This chapter aims to give a practical introduction to the use of TexGen for creating textile models and a guide to exporting them in a form which can be used for further analysis by finite element (FE) or computational fluid dynamics (CFD) packages. It starts with a brief overview of the modelling theory used in the software which will give a better understanding of the functions available both via the user interface and the Python scripting API. Utilising this functionality it is possible to create a range of textile models as shown in Fig 2.9.1.

The software can be downloaded either as executables to run on Windows computers from https://sourceforge.net/projects/texgen/, or as source code for building on either Windows or Linux operating systems from https://github.com/louisepb/TexGen.

2.9.2. TexGen Modelling Theory

A good understanding of the TexGen modelling theory will enable the user to make the most effective use of its capabilities and give the knowledge required to enable modelling of a wide range of textile structures. This section gives a brief overview of the theory; a more comprehensive description is given in (Sherburn, 2007).

Using TexGen, textiles are built up from an assembly of yarns which are modelled as solid volumes. These are defined by a yarn centreline, specifying the smallest repeating section of the yarn, and cross-sections along that yarn. A domain can be specified which defines the area of the textile to be investigated. This is typically a unit cell for prediction of, for example, mechanical properties or permeability. For other simulations, for example impact, a larger domain which covers more repeats of the weave pattern may be required.

2.9.2.1 Yarn Path

Yarn paths are specified by a set of master nodes, given by a set of x,y,z coordinates, which define the centreline of the yarn. For a yarn with a repeating pattern the smallest repeating section of the yarn is specified. The exact path between those nodes is determined by an interpolation function: Bezier, natural cubic or linear spline. When the yarn is built (automatically by TexGen) the interpolation function calculates the yarn path at a set of intermediate positions, slave nodes, the number of which is determined by the yarn resolution parameter. Fig 2.9.2a shows a yarn with three master nodes and indicates the positions of the calculated slave nodes. If the yarn is to be repeated then the 'periodic' option should be selected so that the cross-sections at either end of the yarns are continuous if the yarn is repeated, as shown in Fig 2.9.2b.

2.9.2.2 Cross-sections

Cross-sections are specified along the length of the yarn. These are given as 2D sections which are then oriented in a plane perpendicular to the yarn tangent. By default, the cross-section will be constant along the length of a yarn but cross-sections may also be set at each master node or at positions along the yarn, selected by the user.

Several cross-sectional shapes are available: ellipse, lenticular, power ellipse, rectangle and polygon. There is also a hybrid section which can be specified using a combination of different shapes. The polygon section is not available in the user interface and can only be generated from a Python script. The hybrid and polygon shapes give the most flexibility which may be useful when creating yarns which are in contact with each other and therefore deform to non-standard cross-sections.

The specified 2D cross-sections are then oriented in a plane orthogonal to the yarn tangent at their specified positions along the yarn. An interpolation function generates intermediate cross-sections at each slave node and then a surface mesh is generated by joining points around the edge of each cross-section with the corresponding points on the section at the adjacent slave nodes, shown in Fig 2.9.3.

2.9.2.3 Domain and Yarn Repeats

Defining the master nodes for the smallest repeating section of a yarn will allow that specific length of the yarn to be built. In order to generate a larger section of the textile comprising several repeats of the weave pattern a set of repeat vectors need to be specified. These are a set of x,y,z vectors, one for each direction in which the yarn is to be repeated. Typically this would be one vector for a repeat in the x direction and another one for the y. If the material is sheared then it might include a vector with both x and y components.

At this point it is possible (in theory) to define an infinite textile. A domain is therefore specified to bound the region of the textile which is to be generated. This may include a number of repeats and will be the area of the model which is exported for further analysis. In many cases the domain will correspond to a unit cell, typically for analysis to predict mechanical properties or permeability. In some cases, for example ballistic simulations, a larger area of the textile is needed and a larger domain can be selected accordingly.

2.9.2.4 Yarn Orientations and Volume Fractions

When outputting textile models for simulation it is necessary to know material orientations and, in the case of composite materials, local yarn volume fractions. In order to calculate these TexGen generates a mesh for each slave node section as shown in Fig 2.9.4. Orientations are generated by finding the vector between centre points of corresponding elements on meshes at adjacent slave nodes.

Volume fractions are calculated automatically within TexGen using equation 1. Yarn properties are used to calculate the yarn fibre area and the area of the local yarn section is calculated by extracting the yarn cross-section at the required point on the yarn and then calculating its area.

$$Volume\ Fraction = \frac{Fibre\ Area}{Section\ Area} \tag{1}$$

2.9.3 Automatically Generating Textile Models using the TexGen Weave Wizards

The simplest way to start using TexGen is to use one of the wizards to automatically generate textiles. These all use the CTextile class as a base class which contains yarn information defined in the format described in section 2.9.2. Additional functionality is implemented in the inherited classes which automates the process of generating the master nodes and cross-sections. The class hierarchy is shown in Fig 2.9.5.

2.9.3.1 2D Weave Wizard

The 2D wizard defines a grid which specifies whether the warp or weft is up at a given crossover point of the warp and weft yarns. In the weave wizard this information is input using the Pattern Dialog (Fig 2.9.6). Master nodes are defined at each of the crossover points and their x, y coordinates are defined by the yarn widths and spacing defined in the initial dialog (Fig 2.9.7). The z positions are defined by the value in the grid and the thickness of the yarns specified. The yarns are assumed to have constant, elliptical cross-sections. Once the weave pattern (and hence the master node positions) is specified the textile is built in the way described in section 2.9.2.

There is an option to create a sheared textile. The additional shear angle parameter is used for calculation of the master node positions and the option is given to create a sheared domain. In this case a parallelogram shaped domain is created with sides at the shear angle specified.

For both straight and sheared textiles there are refine options which interrogate the model for intersections and apply adjustments to the cross-sections to remove the intersections. In this case extra cross-sections are added to the model which use the polygon section. These are described more fully by Sherburn (2007) and (Zeng et al., 2015) .

Exercise 2.9.1: Create a 4x4 satin weave textile using the 2D weave wizard and the parameters in Table 1.

	Yarn width / mm	Yarn height / mm	Yarns / cm
Warp	0.9	0.3	10
Weft	0.7	0.2	13

Table 1: Yarn data for Exercise 2.9.1

2.9.3.2 3D Weave Wizard

The 3D weave wizard builds a 3D grid of crossover positions as illustrated in Fig 2.9.8. At each of these positions (i, j) it stores information as to whether it is a warp, weft or no yarn in a cell vector (see Fig 2.9.8). The 3D weave dialogs allow entry of warp, weft and binder yarn information including numbers of yarns and their dimensions. The weave pattern dialog populates the grid and then uses the information to build the yarns for the textile.

The orthogonal, angle interlock and layer-to-layer options impose constraints in the pattern dialog using information about the known yarn configurations for each of these textiles. For example in an orthogonal weave the binder yarns must pass through all weft layers and cross over the weft yarns either above the top weft layer or below the bottom layer.

A refine option is available in the orthogonal weave which implements observed geometric deformations to give a more realistic textile model, adjusted to a given thickness. Yarn properties are required for this option as local volume fractions are calculated during the process to ensure that the compaction being modelled is realistic. An example of a refined orthogonal weave can be seen in Fig 2.9.1b. A more detailed description of these refinements is given in (Zeng et al., 2014)

Exercise 2.9.2: Create a 3D orthogonal weave textile with a total of 6 warp yarns, with a ratio of 2 warp stuffers to 1 binder yarn, and 4 weft yarn stacks. There should be 3 yarns in each weft stack and 2 in each warp stack. Textile thickness is 1.4mm. Use the yarn parameters in Table 2.

	Yarn width /mm	Yarn height /mm	Yarn spacing /mm	Yarn fibre diameter /mm	Fibres per yarn
Warp	3.6	0.35	3.8	0.007	5000
Weft	2.58	0.25	2.8	0.007	8000
Binder	1.375	0.16	1.4	0.007	3500

Table 2: Yarn data for Exercise 2.9.2

2.9.4 Creating Custom Textiles and Editing Textiles using the Graphical User Interface

A User Guide is available on the TexGen webpage (TexGen 2019) (TexGen, 2019) which gives information on the different features in the GUI. This section illustrates how to use these to create a TexGen model and highlights some of the features available. It is recommended that the reader opens the TexGen GUI and follows the steps given in order to gain practical experience in its use. Also, if the Python Output tab is selected in the Log window then it is possible to see the Python commands which have been executed in order to carry out the option selected. These can be related to both the theory described in section 2.9.2 and the Python commands described in section 2.9.5 and will give a better understanding of how TexGen works. User interface commands are given in *italics*. Also note that there is no Undo option in TexGen so it is advisable to save the model at frequent stages using the *File->Save TexGen File* option.

First, ensure that the Controls, Outliner and Log windows are all open as shown in Fig 2.9.9: Select the *Window* option from the main menu and ensure that all three window options are checked. The windows can be dragged and dropped into the preferred locations.

2.9.4.1 Create a Simple Textile

1. Create an empty textile: Either select *Create: Empty* from the Controls window or *Textiles->Create Empty…* from the main menu. Enter a name for the textile to be created in the *Textile name* dialog box.
   The Render window in the centre of the GUI becomes black and shows a set of axes and a tab containing the name of the textile which was input. At this point a textile has been created and added to the textile list.Select *Modeller* from the *Controls* drop-down menu. This changes the options in the Controls window to reflect those in the *Modeller* menu option.
2. Create a yarn: Either select *Create: Yarn* from the Controls window or *Modeller->Create Yarn…* A dialog is displayed which allows for entry of the start and end points of the yarn and the number of nodes to be selected. For this example use the default settings.
   A single yarn is displayed in the Render window and a tree is displayed in the Outliner window showing that Yarn 0 has been created with Nodes 0 and 1.
3. Select the Yarn by clicking on the yarn in the Render window or by selecting *Yarn(0)* in the Outliner window. The yarn will be displayed in grey when selected.
4. Create a second yarn: Make a copy of the yarn, either by using the *Duplicate Yarn* button at the bottom of the Outliner window or by pressing *Ctrl-d* on the keyboard.
   A second yarn has been created which can be seen in the Outliner window. The Render window still apparently shows only one yarn, although the colour has changed. This is

because the second yarn created is identical to the first and so only the second one to be rendered is visible. This is shown in Fig 2.9.10.

5. Move second yarn: Select *Yarn 1* in the Outliner window. A set of axis handles will be displayed on the yarn. Click and hold the green y-axis arrow using the left mouse button. Keeping the button pressed, the yarn can now be dragged in the y direction. Drag the yarn until the *Position: Y* text box in the Controls Dialog has the value 5. The yarn can also be moved by selecting the yarn and then typing the value '5' into the *Position: Y* text box.
6. Duplicate Yarn 0: Select Yarn 0 and duplicate as described in steps 4 and 5.
7. Move node: Select *Node1* of *Yarn2* by selecting in the Outliner window, shown as greyed. Set values x = 0 and y = 10 using the *Position* text box in the Controls window.
8. Duplicate Yarn 2: Select Yarn 2 and duplicate as described in steps 4 and 5.
9. Move Yarn 3: As describe in step 6, either drag Yarn 3 type in position to move Yarn 3 to x = 5.
   At this point the textile should look as shown in Fig 2.9.11.
10. Insert nodes: Select node 1 in each yarn using the Outliner window. Multiple nodes can be selected using *Ctrl-Left Click.* Use the *Insert Node* button in the Outliner window to insert a node before the selected nodes. Note that inserted nodes will be positioned at the mid-point between the selected node and the previous node (ie will always be inserted before the selected node).
11. Move nodes: In order to create a woven pattern in the yarns the z coordinate of some of the nodes needs to be changed. Using the Outliner window select *Nodes 0 and 2 of Yarns 0 and 3 and Node 1 of Yarns 1 and 2.* Using the *Position:Z* text box in the Controls window change the z coordinate to *'2'.*
    The resulting yarn configuration is shown in Figure 2.9.12.
12. Select all yarns either by selecting the yarns in the Render window using *Shift-Left Click* to select multiple yarns or by selecting *Yarn 0* in the Outliner window and then *Shift-Left Click Yarn 3* to select all yarns between the two yarns. Note that in the Outliner window *Ctrl-Left Click* is used to make multiple discrete selections and *Shift-Left Click* is used to select a block between the two selections.
13. Assign yarn cross-sections: Either select *Assign: Section* from the Controls window or *Modeller->Assign Section…* The *Select Yarn Section dialog* is displayed. In this case a constant cross-section is assigned for each yarn. Select the *Edit Section* button.
    The *Select Cross-Section Shape dialog* is displayed as shown in Fig 2.9.13a. Select *Lenticular* from the drop down menu. Enter *Width, 4 and Height, 2,* then select *OK* and also on the *Select Yarn Section dialog.*
    The lenticular sections will now have been applied to all yarns as shown in Fig 2.9.13b
14. Assign domain: It can be seen that, at the moment just the single section of the yarns have been specified. The way in which the yarns are to repeat and the boundaries of the unit cell need to be specified. In this case the domain will be specified first using the *Domain-<Create Box…* option. In the *Box Domain dialog enter Maximum X and Y values, '10' and Minimum Z value, '-1' and Maximum Z value, '3'.*
    If the *Rendering->Trim To Domain* option is selected the model will now look as shown in Fig 2.9.14.
15. Add repeats: All of the yarns created have a length of 10. To create a continuous textile all of the yarns need to have repeat vectors of 10 specified in both the x and y directions. Select all of the yarns as described in step 13 and then either select *Assign: Repeats* from the Controls window or *Modeller->Assign Repeats…* The *Repeats dialog* will be displayed and two vectors should be specified as shown in Fig 2.9.15. Two repeat vectors are defined here; row 1 gives the repeat in the x direction and row 2 the repeat in the y direction.
    A complete, repeating unit cell has now been created as shown in Figure 2.9.16.
16. Save the model: Select the *File->Save TexGen File* option and then select the *Standard* option in the subsequent *Save dialog.*

The file for the final textile produced, GUISample.tg3, can be found in ?

2.9.4.2 Editing Textiles

Once a textile model has been created, either by using the GUI or a Python script, it can be edited using the same methods that were used to create the textile. The yarns and nodes can be selected in the same way and then either dragged using the arrow widgets or by entering values into the *Position* text box in the Controls window.

One example of where changes may need to be made to a model is if there are intersections between the yarns which are both unrealistic and may prevent FE simulations from running. The *Rendering->Render Textile Interference* or *Rendering->Render Textile Interference Depth options* can be used to give a visualisation of where intersections between yarns occur. The intersection depth is calculated at each surface mesh point. In the case of the latter option the depth of the intersection is reflected by the size of the points rendered as shown in Fig 2.9.17 which shows the interference for the model created in Section 2.9.4.1. A message is given in the TexGen Output tab of the Log window stating the depth of the maximum interference in the model.

In this case the following steps could be taken to edit the textile and reduce interference:

1. It can be seen that there is interference where the two central yarns (yarns 1 and 3) cross in the centre of the model shown in Figure 2.9.17. In this case the interference could be reduced by changing the cross-section at node 1. The model was created with constant cross-sections along the yarn so first it is necessary to change the interpolation to interpolate between nodes. This allows a different cross-section to be specified at each node point.
   Select *Yarn 1* either using the Outliner window or by *left-clicking* on the yarn in the Render window.
2. Select *Modeller->Assign Section* from the main menu or the *Section button* in the *Modeller tab* of the Controls window.
3. Select *Interpolate between nodes* from the drop down menu in the *Select Yarn Section dialog.* The dialog will then display a list of the nodes for the yarn as shown in Fig 2.9.18.
4. Click on *Section at Node 1* to select the node and then click on the *Edit Section button.*
5. The Select Cross-Section Shape dialog is displayed, showing the information for the current cross-section at the selected node. In this case the first change that will be tried in order to reduce the intersections will be to change the Distortion of the lenticular section. This changes the vertical position of the maximum width of the section.
   Enter *Distortion, 0.1* and then select *OK* on this and the Select Yarn Section dialogs.
6. Select the *Refresh View* option in the Rendering menu. On inspection of the interference markers on the central section of Yarn 1 it can be seen that the intersection towards the edge is reduced but there is still some intersection towards the middle. A better solution might be a hybrid section which can be used to change the lower half of the yarn section.
7. Repeat steps 1-4 to again allow the section at Node 1 to be edited. In the *Select Cross-section Shape dialog* select *Hybrid* from the drop-down menu.
8. Click outside the shape shown in the centre of the dialog box twice to create two points to split the shape. The circular tags can then be used to drag the points to the desired points. In this case use them to create a horizontal line as shown in Fig 2.9.19.
9. The top section will be set to be the same as the original section: *Click on the upper section,* then *Select Lenticular, Width = 4.0, Height = 2.0* and then click *OK*. To change the lower section *Click on the lower section.* In the subsequent Cross-Section Shape dialog *Select Lenticular, Width=4.0, Height=1.8* and then click *OK.*
10. Select the *Refresh View* option in the Rendering menu. It will be seen that there are now no intersections in the central region.

This edited version of the textile is available in CorrectInterferenceYarn1.tg3. The domain can be edited using the *Domain->Edit Domain* option from the main menu or the *Edit button* in the *Domain tab* of the *Controls window.* The *Domain Editor dialog* is displayed as shown in Fig 2.9.20. It can be seen that six planes are shown, one for each face if a box domain was created. In order to change the domain planes *Click on the row of one of the planes, enter the required values in the x, y, z and d boxes and then select Replace Plane.* Planes can be added or deleted using the *Add Plane* and *Delete Plane* buttons.

Exercise 2.9.3: Edit the domain of the textile so that two repeats of the weave are displayed with the edges of the domain running between the yarns.

2.9.4.3 Adding Material Properties

Selected yarn properties can be set for the whole textile. If the *Modeller->Yarn Properties* option is selected without selecting any yarns the *Properties dialog* allows the yarn linear density, fibre density, total fibre area, fibre diameter, fibres per yarn and areal density to be set for the textile. In the dialog, shown in Figure 2.9.21, both the Value and the Units can be entered for any of the properties. Standard metric units can be used as well as imperial length units and the textile-related units of tex and denier.

If one or more yarns are selected then the *Properties dialog* will allow input of material properties, Young's Modulus, Shear Modulus, Poisson's Ratio and Alpha (coefficient of thermal expansion) which will be applied to the selected yarns.

For composite materials it is assumed that the space in the domain which is not occupied by yarns is matrix whose properties can be set using the *Modeller->Matrix Properties* option. The Young's Modulus, Shear Modulus, Poisson's Ratio and Alpha can be set using the *Properties dialog.*

The yarn densities and areas are used in TexGen to calculate fibre volume fractions. These must be set if element fibre volume fractions are required in the exported files. The mechanical properties are used for populating the materials sections of the export files.

2.9.4.4 Loading and Saving TexGen Files

TexGen models can be saved using the *File->Save TexGen File* option from the main menu. Models are saved with a .tg3 file extension. These files are in XML format and can be viewed and edited using a text editor. Examining one of these files in a text editor will show that the structure of the file follows the modelling format described in previous sections. The data in the file can be edited in a text editor and these changes would be reflected when the file is next loaded into TexGen.

There are three options when saving TexGen models:

- Minimal – textile data only (only use if model was created using one of the weave classes)
- Standard – textile and yarn data. This is the default setting
- Full – textile, yarn and mesh data. This saves all of the surface mesh information.

To load TexGen models in .tg3 format use the *File->Open TexGen File* option.

2.9.5 Python Scripting API

TexGen software is written using the C++ programming language and is split into several modules as shown in Fig 2.9.22. Use of the GUI has been described in earlier sections which allows easy generation of standard textile weaves but is more time consuming for non-standard textiles. For more complex textiles use of the Python scripting interface gives greater flexibility. The Python interface is automatically generated from the C++ code using the Simplified Wrapper and Interface Generator (SWIG) (Beazley and Matus, 2007) and allows functions from the Core, Renderer and Export modules to be called from a Python script. All menu items in the GUI execute one or more of these Python commands and much can be learned by looking at the Python Output window which echoes the commands executed by the GUI.

There is a comprehensive scripting guide available on GitHub (Brown et al., 2019) and this section highlights a selection of the Python scripting capability rather than giving an in-depth guide.

TexGen Python scripts can be used either as standalone scripts or can be run using the *Python->Run Script* menu item in the TexGen GUI. In the latter case the textile will be visualised in the GUI after being added to the textile list within the script. Python commands can also be used to edit a textile already loaded into the GUI.

2.9.5.1 Creating a Textile Model using a Python Script

To create a Python script a text file is created in an editor with a .py extension. In Windows Notepad++ (Notepad++, 2019) is a good option for doing this. Python code will be displayed as shown below:

```
# Python code

# Lines beginning with a # are comments
```

Using Python scripting a textile is created in the order described in the previous sections. First a textile is defined:

```
# Create a textile

Textile = CTextile()
```

Yarns are then created which are added to the textile. If more than one yarn is to be created these can be created as a Python list:

```
# Create a Python list containing 4 yarns

Yarns = [CYarn(), CYarn(), CYarn(), CYarn()]
```

Yarns contains 4 empty yarns which, as yet, are not associated with the textile which has been created. Data which defines the geometry of the yarns will be added to each yarn and then these yarns will be assigned to the textile. First, master nodes are added to each yarn to define the yarn path:

```
# Add nodes to the yarns to describe their paths
Yarns[0].AddNode(CNode(XYZ(0, 0, 0)))
Yarns[0].AddNode(CNode(XYZ(0.22, 0, 0.05)))
Yarns[0].AddNode(CNode(XYZ(0.44, 0, 0)))

Yarns[1].AddNode(CNode(XYZ(0, 0.22, 0.05)))
Yarns[1].AddNode(CNode(XYZ(0.22, 0.22, 0)))
Yarns[1].AddNode(CNode(XYZ(0.44, 0.22, 0.05)))

Yarns[2].AddNode(CNode(XYZ(0, 0, 0.05)))
Yarns[2].AddNode(CNode(XYZ(0, 0.22, 0)))
Yarns[2].AddNode(CNode(XYZ(0, 0.44, 0.05)))

Yarns[3].AddNode(CNode(XYZ(0.22, 0, 0)))
Yarns[3].AddNode(CNode(XYZ(0.22, 0.22, 0.05)))
Yarns[3].AddNode(CNode(XYZ(0.22, 0.44, 0)))
```

Nodes are specified using the CNode class which is constructed using an instance of the XYZ class giving the position of the node. The XYZ class is constructed using 3 parameters representing the x,y,z coordinates of the point. In the Python code above the number in square brackets after Yarns refers to a particular yarn in the list and then the Yarn class function AddNode is called with the nodes created using CNode.

So far the nodal positions have been specified for each yarn but there is no information about cross-sectional shape or interpolation between the nodes.

If the same information is to be assigned to each yarn then a loop can be used to iterate through each yarn in turn and assign the properties to each in turn. To loop through each yarn in the list:

```
# Loop over all the yarns in the list
for Yarn in Yarns:
```

When inside the loop the current yarn being changed is referred to using the `Yarn` variable. In Python the instructions within the `for` loop are defined by indentation.

First the interpolation between the nodes is defined by creating an instance of one of the classes derived from `CInterpolation`. In this case periodic cubic spline interpolation is used, defined by the class `CInterpolationCubic`:

```
    # Set the interpolation function
    Yarn.AssignInterpolation(CInterpolationCubic())
```

Next, cross-sections are defined using the `AssignSection` function. This takes an instance of a class derived from `CYarnSection`, in this case a constant cross-section is created using `CYarnSectionConstant`. This, in turn, takes an instance of a class derived from `CSection` as a parameter, in this case an elliptical cross-section is defined using `CSectionEllipse` using numerical parameters defining the width and height:

```
    # Assign a constant cross-section along the yarn
    # with elliptical shape

    Yarn.AssignSection(CYarnSectionConstant(CSectionEllipse(0.18,0.0
    4)))
```

Further information on creating different cross-sectional shapes and varying the cross-section along the length of the yarn can be found in the TexGen scripting guide (Brown et al., 2019).

The resolution of the surface and volume meshes created are defined using the `SetResolution` function. If one parameter is used then this gives the number of points around a cross-section and the number of points along the length of the yarn (defining the slave nodes) is calculated automatically so that the distance along the length is similar to that around the cross-section. If two parameters are provided the first specifies the number of slave nodes along the yarn and the second the number of points around the yarn. In this case one parameter is given:

```
    # Set the resolution of the mesh created
    Yarn.SetResolution(20)
```

The repeat vectors are defined as described in section 2.6.2 using the function `AddRepeat` which takes an instance of the `XYZ` class as a parameter to define the x,y,z components of each repeat:

```
    # Add repeat vectors to the yarn
    Yarn.AddRepeat(XYZ(0.44, 0, 0))
    Yarn.AddRepeat(XYZ(0, 0.44, 0))
```

This completes the definition of the yarn which may now be added to the textile:

```
    # Add the yarn to the textile
    Textile.AddYarn(Yarn)
```

This is the end of the `for` loop, marked by the end of indentation in the following code.

The last requirement for creation of the textile is specification of a domain to define the area of the textile being observed. The function `AssignDomain` takes an instance of a class derived from `CDomain`, in this case `CDomainPlanes` using the constructor which takes two `XYZ` points defining the lower left and upper right corners for a box shaped domain:

```
Textile.AssignDomain(CDomainPlanes(XYZ(0, 0, -0.02), XYZ(0.44, 0.44, 
0.07)))
```

Lastly `Textile` is added to the list of textiles so that it can be rendered in the TexGen GUI. Each textile is given a name which is either passed as a parameter to the `AddTextile` function or will be generated automatically if no parameter is given. This name will be shown at the top of the render window in the GUI:

```
# Add the textile with the name "polyester"
AddTextile("polyester", Textile)
```

The completed script is shown here:

```
# Create a textile
Textile = CTextile()

# Create a python list containing 4 yarns
Yarns = [CYarn(), CYarn(), CYarn(), CYarn()]

# Add nodes to the yarns to describe their paths
Yarns[0].AddNode(CNode(XYZ(0, 0, 0)))
Yarns[0].AddNode(CNode(XYZ(0.22, 0, 0.05)))
Yarns[0].AddNode(CNode(XYZ(0.44, 0, 0)))

Yarns[1].AddNode(CNode(XYZ(0, 0.22, 0.05)))
Yarns[1].AddNode(CNode(XYZ(0.22, 0.22, 0)))
Yarns[1].AddNode(CNode(XYZ(0.44, 0.22, 0.05)))

Yarns[2].AddNode(CNode(XYZ(0, 0, 0.05)))
Yarns[2].AddNode(CNode(XYZ(0, 0.22, 0)))
Yarns[2].AddNode(CNode(XYZ(0, 0.44, 0.05)))

Yarns[3].AddNode(CNode(XYZ(0.22, 0, 0)))
Yarns[3].AddNode(CNode(XYZ(0.22, 0.22, 0.05)))
Yarns[3].AddNode(CNode(XYZ(0.22, 0.44, 0)))

# Loop over all the yarns in the list
for Yarn in Yarns:

    # Set the interpolation function
    Yarn.AssignInterpolation(CInterpolationCubic())

    # Assign a constant cross-section all along the yarn of 
elliptical shape
    Yarn.AssignSection(CYarnSectionConstant(CSectionEllipse(0.18, 
0.04)))

    # Set the resolution of the surface mesh created
    Yarn.SetResolution(20)

    # Add repeat vectors to the yarn
    Yarn.AddRepeat(XYZ(0.44, 0, 0))
    Yarn.AddRepeat(XYZ(0, 0.44, 0))

    # Add the yarn to our textile
    Textile.AddYarn(Yarn)

# Create a domain and assign it to the textile
Textile.AssignDomain(CDomainPlanes(XYZ(0, 0, -0.02), XYZ(0.44, 0.44,
0.07)))

# Add the textile with the name "polyester"
AddTextile("polyester", Textile)
```

The resulting textile is shown in Figure 2.9.23. The complete script Polyester.py can be found in ? This example shows the general process for creating a textile script. Much more complex textiles can be created by making use of the functions available with the TexGen Python API. A range of sample scripts for a variety of textiles are available at https://github.com/louisepb/TexGenScripts.

Exercise 2.9.4: Make the following changes to the Polyester.py script:

1. Change the cross-sections to lenticular.
2. Change the cross-section dimensions: Width = 0.2, height = 0,048
3. Change the height of the domain to suit the amended yarn dimensions so that it does not cut off any of the yarns.
4. Change the yarn resolution to 40.

2.9.5.2 Editing Textiles using the Python Console

Textiles which are displayed in the TexGen render window, whether created using the GUI, by loading an existing TexGen model from a .tg3 file or by running a Python script are all created using the same set of classes. Because of this it is then possible to access the textile and yarn information via the Python console and use the same set of Python commands to edit the textile. Some examples of this are given below.

The most recent textile loaded can be retrieved using

```
textile = GetTextile()
```

If more than one textile is loaded in the GUI then the required textile can be selected by passing a string parameter to the `GetTextile` function containing the name of the textile (as displayed on the render window tab).

A specific yarn can be accessed using the `GetYarn` function using the required yarn number as a parameter:

```
yarn = textile.GetYarn(1)
```

The API functions can then be used to edit the textile. For example the resolution could be set using:

```
yarn.SetResolution(20,40)
```

which would change the resolution of yarn 1 to have 20 slave nodes and 40 points around the outline of each yarn.

To change the position of a node in a yarn the `ReplaceNode` function can be used, again using the `CNode` class to create an instance of a node which is passed to the function:

```
yarn.ReplaceNode(1, CNode(XYZ(5,5,2.4))
```

The first parameter of `ReplaceNode` gives the index of the node to be changed.

2.9.6 Pre-processing Textile Models for Simulation

TexGen is mainly used as a pre-processor for simulations, typically finite element (FE) simulations for prediction of mechanical properties or computational fluid dynamics (CFD) simulations for permeability predictions. In order to generate input files it is necessary to create a mesh of the textile model, exported in a format which can be imported into third party software for analysis. Additional information is exported which includes yarn orientations and volume fractions for each mesh element.

ABAQUS® input files can be generated which include periodic boundary conditions (Li and Wongsto, 2004) and steps for prediction of thermo-mechanical properties of a unit cell. Alternatively an input file can be created which just contains the mesh information, leaving the user to set up the rest of the input according to the analysis to be performed.

There are several options for exporting models within TexGen and the option selected may depend on the type of analysis to be carried out, the type of textile and whether a dry textile or a textile composite is to be simulated. The following sections describe the different export options available within TexGen and highlight advantages and limitations of these methods.

2.9.6.1 Dry Fibre Volume Mesh Export

This method can be used to create a conformal mesh of just the yarns. The mesh is created in two stages: first a two dimensional mesh is created at each slave node using a rectangular/triangular mesh generation technique as illustrated in Fig 2.9.4 and then the mesh points between subsequent sections are joined to form hexahedral and wedge elements as illustrated in Fig 2.9.24.

The mesh is exported as an ABAQUS® input (.inp) file as node and element sets. Options are available to specify deformations in the x, y and z directions and to apply compression plates. In dry fibre simulations contact between the yarns must be taken into account and either upper and lower yarn contact surfaces (suitable for woven fabrics) or whole yarn surfaces (more suitable for textiles such as knits) can be selected. The input file generated creates boundary conditions, contacts and steps for applying the specified deformations. Supplementary files are also created; a .ori file contains element yarn orientation information and a .eld file contains yarn volume fraction information for each element.

2.9.6.2 Tetrahedral Volume Mesh Export

To perform simulations on a composite material both the yarn and matrix need to be meshed. In a TexGen model the resin volume is assumed to be represented by the specified domain. In the volume mesh export option a tetrahedral mesh is created. A mesh showing both the whole volume and just the yarns is shown in fig 2.9.25. If the Periodic option is selected then matching triangulated surfaces will be generated for opposite boundaries of the domain, ensuring that nodes and elements match for use in the periodic boundary conditions created in the input file. It should be noted that this method of mesh generation is robust for 2D weaves but less successful for textiles containing vertical or near-vertical yarns. In this case the algorithm used to generate the tetrahedra can result in elongated elements.

If the Periodic Boundary Conditions option is selected then equations are set up in the input file according to those described by Li (Li and Wongsto, 2004). A set of steps are also specified which, when used with the dataHandling.py and effectiveMatPropRVE.py scripts supplied with TexGen, allow material properties to be extracted. A tutorial for extraction of material properties using this method can be found at
http://texgen.sourceforge.net/index.php/Extraction_of_Material_Properties_using_Voxel_Meshing_and_Abaqus

An alternative method of generating a tetrahedral mesh in TexGen is available using the Tetgen Mesh option. This uses the tetgen library (Si, 2015) and may be used to generate meshes for both 2D and 3D woven textiles. The algorithm is not robust for textiles with yarns in contact with each other. To use this method a TexGen model should be created which enforces small gaps between all yarns, a limitation of the method as this is not necessarily a realistic representation of the actual textile.

2.9.6.3 Voxel Mesh Export

It can be challenging to generate conformal meshes of both yarn and matrix regions. Where yarns come into contact with each other there are liable to be small, wedge-shaped areas which are difficult to mesh and often give rise to badly distorted elements. In many cases it has been shown (Matveev et al., 2014) that a non-conformal voxel mesh may be adequate, particularly in the case of prediction of mechanical properties where damage mechanisms are ignored.

In TexGen the voxel mesh is generated by dividing the domain area into cuboids. The voxel is defined as belonging to either the yarn or matrix depending on whether the centre point of the cuboid (element) is located in the yarn or matrix. The information for the orientations and volume fractions of the elements are also based on the values at this centre point. Figure 2.9.26 shows a voxel mesh for a 3D textile with the matrix elements removed.

Periodic boundary conditions are set up as described in section 2.9.6.2 and the analysis for extracting material properties is carried out using the same method.

2.9.6.4 Geometry Export

An alternative approach to preparing models for simulation can be to export the geometry and then use third-party software to mesh the model. The disadvantage of this method is that it doesn’t allow the yarn orientations and volume fractions to be exported, although it is be possible to write a script to interrogate the TexGen model using the centre points of the meshed elements.

TexGen uses the (OpenCASCADE, 2019) library to create geometry in STEP and IGES formats. The algorithm is more robust for the Faceted option which creates a geometry which corresponds to the model surface mesh (as seen when the X-Ray rendering option is used). The Smooth option is less reliable and may not always generate a geometry. The Join Yarn Sections option forces one shape to be created for each yarn, rather than a separate shape for each yarn repeat, but may be slow to generate the geometry.

The Surface Mesh option exports the surface mesh in VTK .vtu format or either binary or ASCII stl format.

2.9.7 Conclusion

This chapter has aimed to give the reader a working knowledge of the TexGen software, giving them the knowledge necessary to create a wide range of textile structures using either the GUI or the Python API. For further information the reader should visit the TexGen webpage, www.texgen.sourceforge.net, which has a range of documentation including a user guide, usage examples and a list of publications which have used TexGen models for analysis. There is also an active user forum, http://texgen.sourceforge.net/phpBB3/index.php, which is both a source of information and a place to ask for help with creating TexGen models.

2.9.8 Acknowledgements

Continued development of TexGen software has been supported by Engineering and Physical Sciences Research Council RSE Fellowship [Grant number: EP/N019040/1].

References

BEAZLEY, D. & MATUS, M. 2007. *SWIG* [Online]. Available: www.swig.org [Accessed 26th September 2019].

BROWN, L. P., GOMMER, F., LONG, A. C., MATVEEV, M., ZENG, X. & YAN, S. 2019. *TexGen Scripting Guide* [Online]. Available: https://github.com/louisepb/TexGenScriptingGuide [Accessed 1st October 2019].

BROWN, L. P. & SHERBURN, M. 2019. *louisepb/TexGen: TexGen v3.11.0 (Version 3.11.0). Zenodo* [Online]. Available: http://doi.org/10.5281/zenodo.3241493 [Accessed 21st October 2019].

LI, S. & WONGSTO, A. 2004. Unit cells for micromechanical analyses of particle-reinforced composites. *Mechanics of Materials*, 36**,** 543-572.

MATVEEV, M. Y., LONG, A. C. & JONES, I. A. 2014. Modelling of textile composites with fibre strength variability. *Composites Science and Technology*, 105**,** 44-50.

Notepad++ [Online]. Available: https://notepad-plus-plus.org/ [Accessed 3rd July 2019].

OPENCASCADE. 2019. *OpenCASCADE* [Online]. Available: www.opencascade.com [Accessed 26th September 2019].

SHERBURN, M. 2007. *Geometric and Mechanical Modelling of Textiles.* PhD, University of Nottingham.

SI, H. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. on Mathematical Software*, 41(2).

TEXGEN. 2019. *TexGen* [Online]. Available: http://texgen.sourceforge.net [Accessed 3rd July 2019].

ZENG, X., BROWN, L. P., ENDRUWEIT, A., MATVEEV, M. & LONG, A. C. 2014. Geometrical modelling of 3D woven reinforcements for polymer composites: Prediction of fabric permeability and composite mechanical properties. *Composites Part A*, 56**,** 150-160.

ZENG, X., ENDRUWEIT, A., BROWN, L. P. & LONG, A. C. 2015. Numerical prediction of in-plane permeability for multilayer woven fabrics with manufacture-induced deformation. *Composites Part A: Applied Science and Manufacturing*, 77**,** 266-274.

Figures

Fig 2.9.1 Examples of TexGen models (a) Layered plain weaves (b) 3D orthogonal weave (c) Triaxial braid (d) Knitted

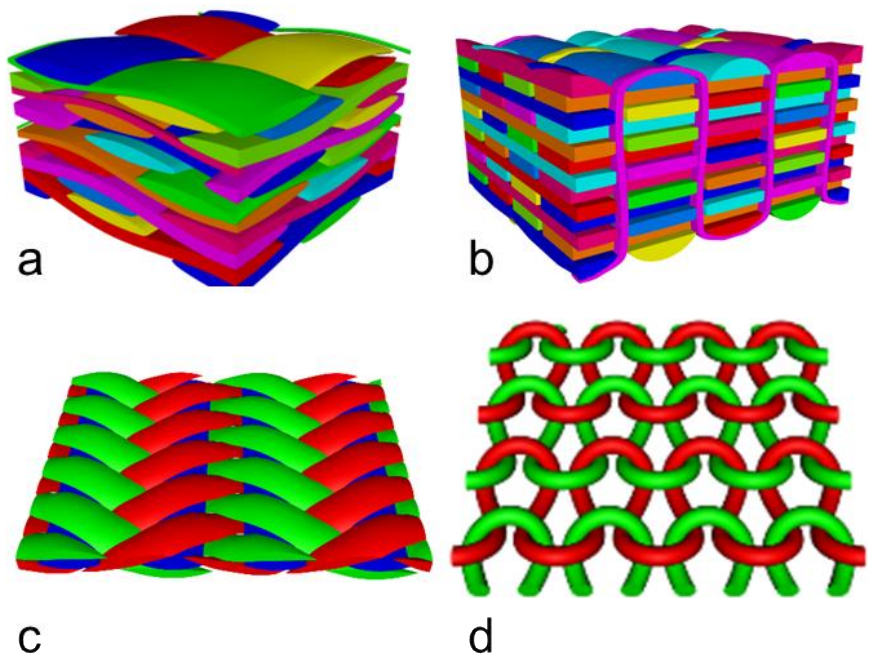


Fig 2.9.2 Yarn with 3 master nodes showing (a) non-periodic and (b) periodic interpolation

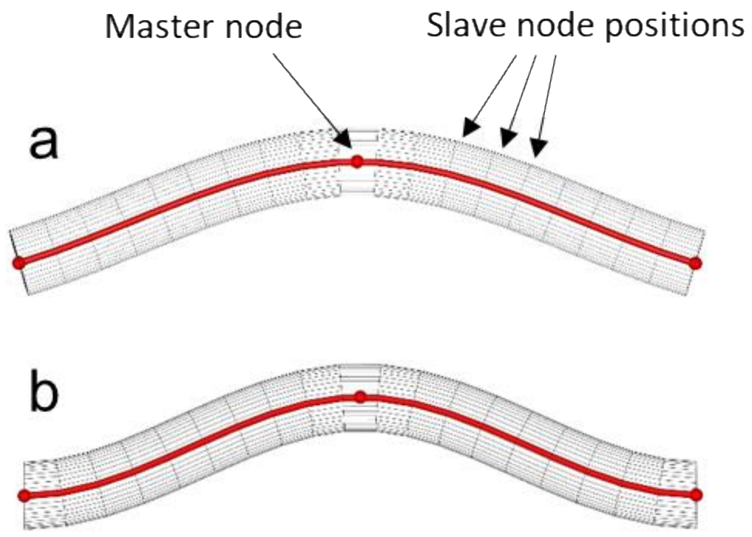


Fig 2.9.3 Yarn with varying cross-section

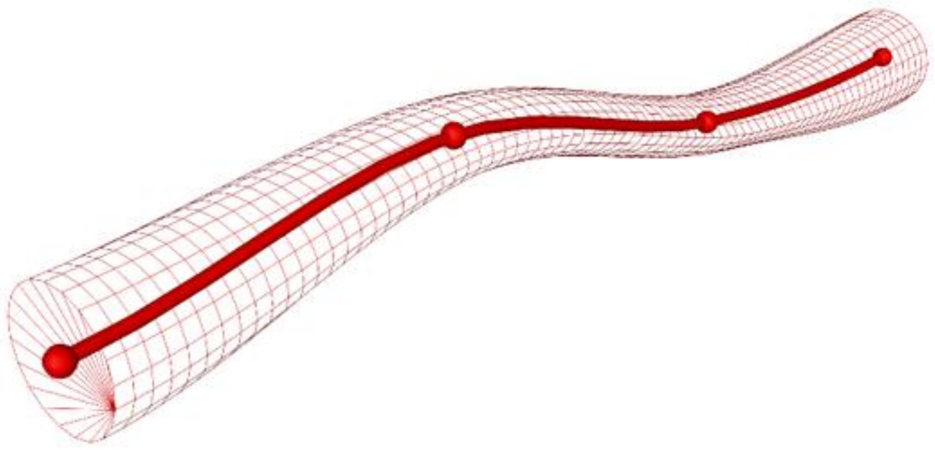


Fig 2.9.4 Section mesh

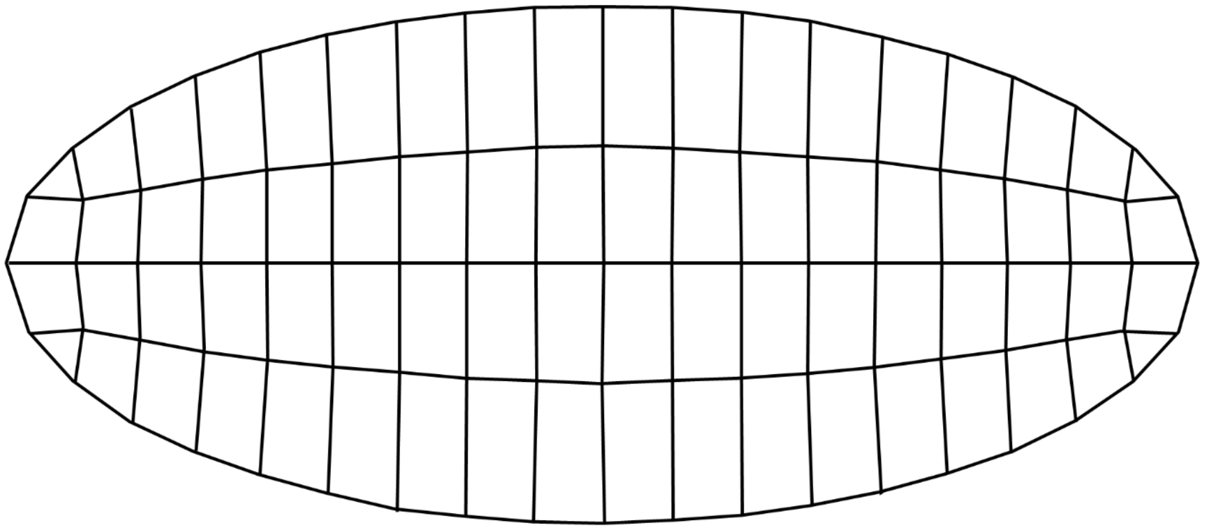


Fig 2.9.5 Textile class hierarchy

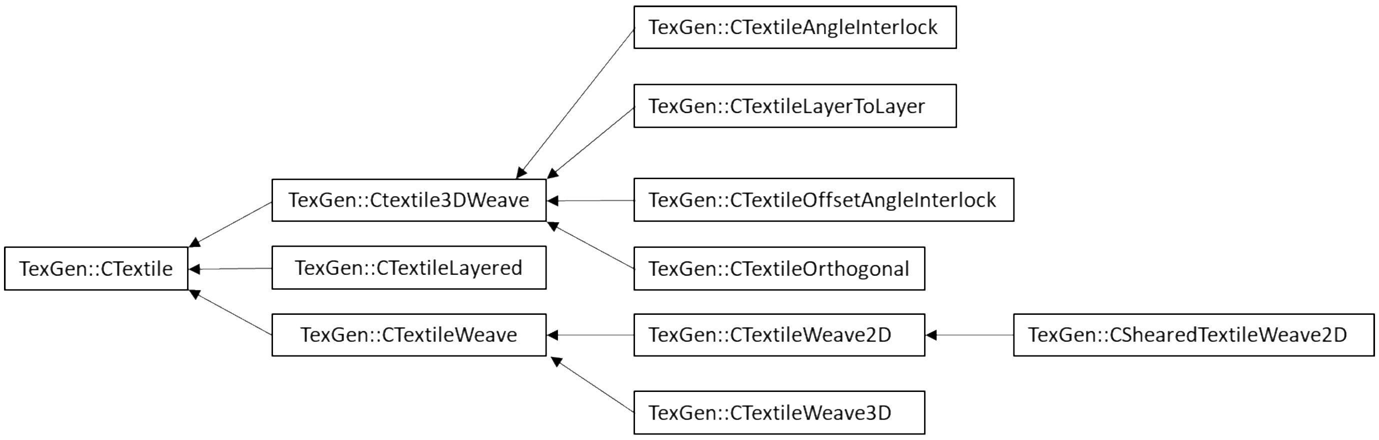
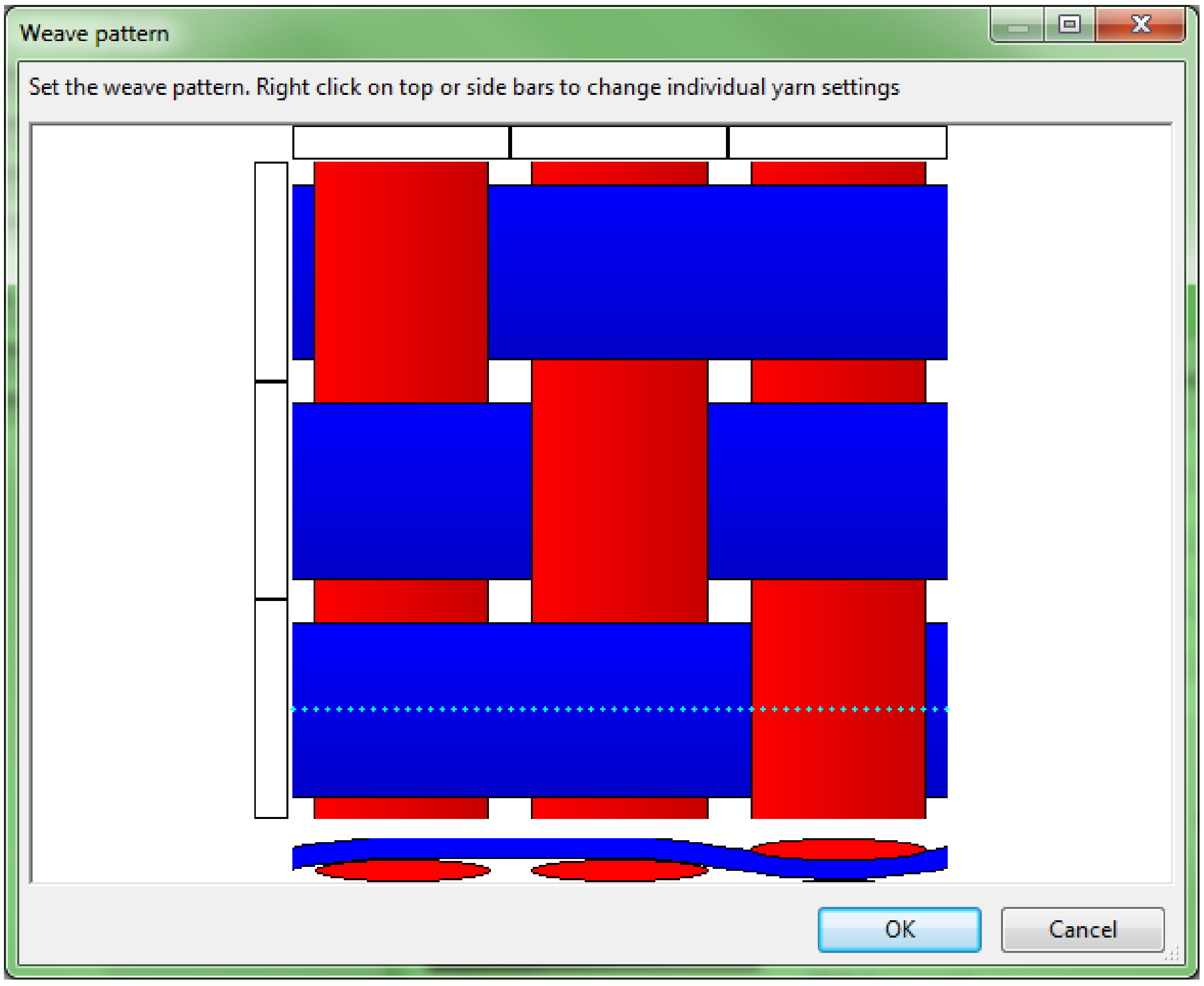
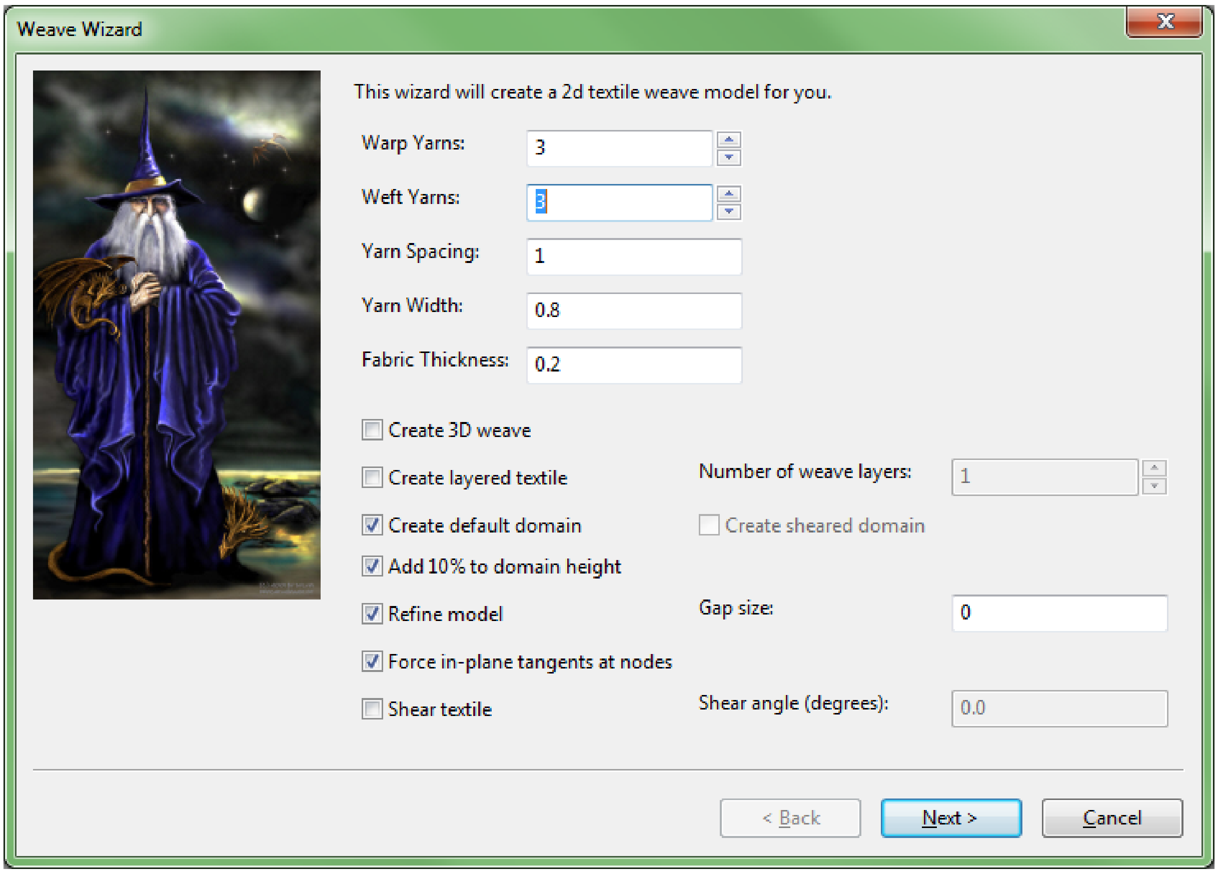

TexGen::CTextileAngleInterlock
TexGen::CTextileLayerToLayer
TexGen::Ctextile3DWeave
TexGen::CTextileOffsetAngleInterlock
TexGen::CTextile
TexGen::CTextileLayered
TexGen::CTextileOrthogonal
TexGen::CTextileWeave
TexGen::CTextileWeave2D
TexGen::CShearedTextileWeave2D
TexGen::CTextileWeave3D

Fig 2.9.6 2D weave pattern dialog



Set the weave pattern. Right click on top or side bars to change individual yarn settings

Fig 2.9.7 2D weave wizard


Weave Wizard

This wizard will create a 2d textile weave model for you.

Warp Yarns: 3

Weft Yarns: 3

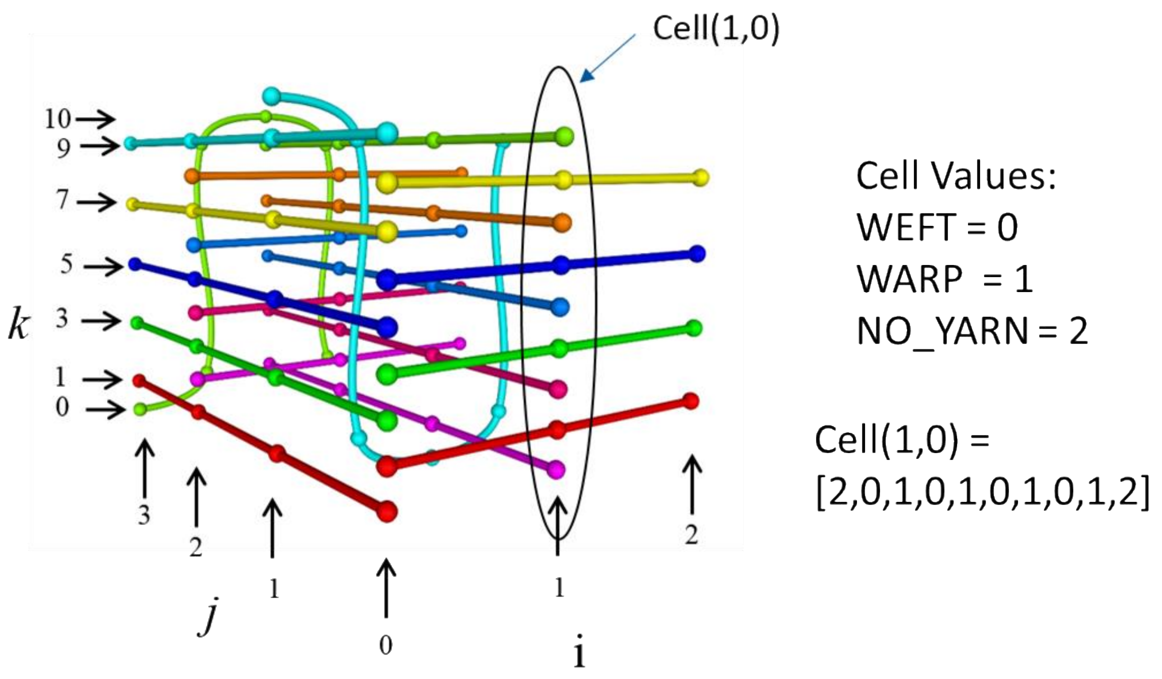
Yarn Spacing: 1

Yarn Width: 0.8

Fabric Thickness: 0.2

- [ ] Create 3D weave
- [ ] Create layered textile — Number of weave layers: 1
- [✓] Create default domain — [ ] Create sheared domain
- [✓] Add 10% to domain height
- [✓] Refine model — Gap size: 0
- [✓] Force in-plane tangents at nodes
- [ ] Shear textile — Shear angle (degrees): 0.0

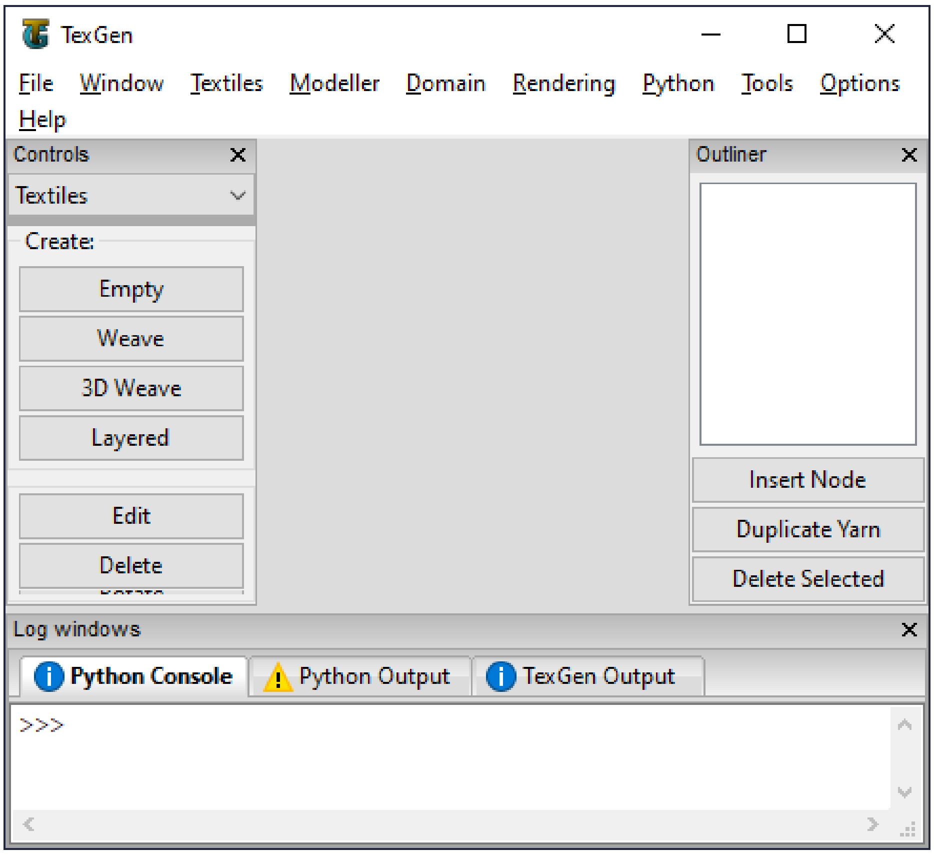
Fig 2.9.8 3D weave grid structure



Cell Values:
WEFT = 0
WARP = 1
NO_YARN = 2

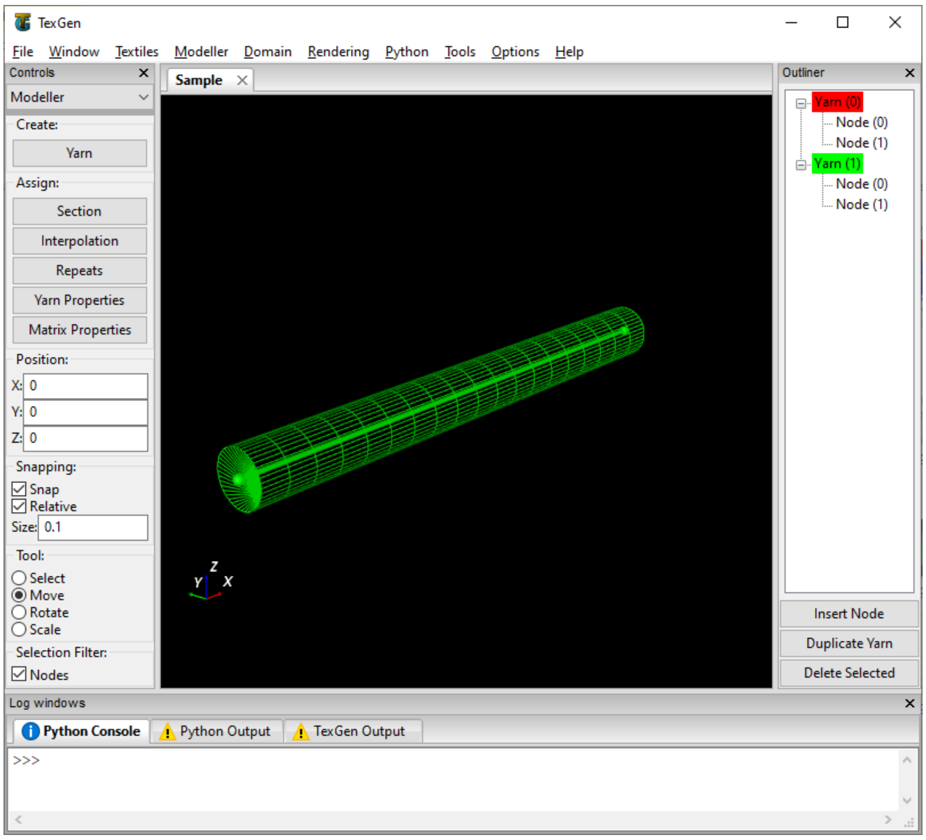
Cell(1,0) =
[2,0,1,0,1,0,1,0,1,2]

Fig 2.9.9 TexGen GUI showing Controls, Outliner and Log windows



```
>>>
```

Fig 2.9.10 Yarns created with two nodes



X: 0
Y: 0
Z: 0
[✓] Snap
[✓] Relative
Size: 0.1
( ) Select
(●) Move
( ) Rotate
( ) Scale
[✓] Nodes

```
>>>
```

Fig 2.9.11 TexGen model with four straight yarns

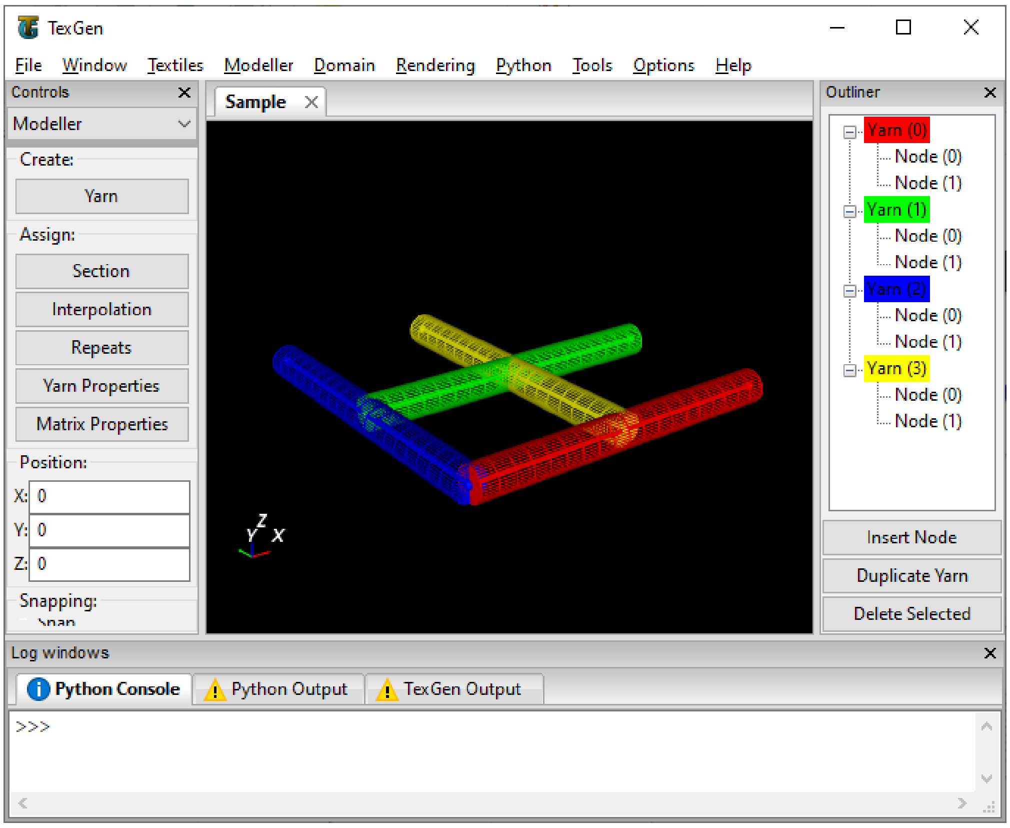


Fig 2.9.12 TexGen model with four yarns and nodal z positions set

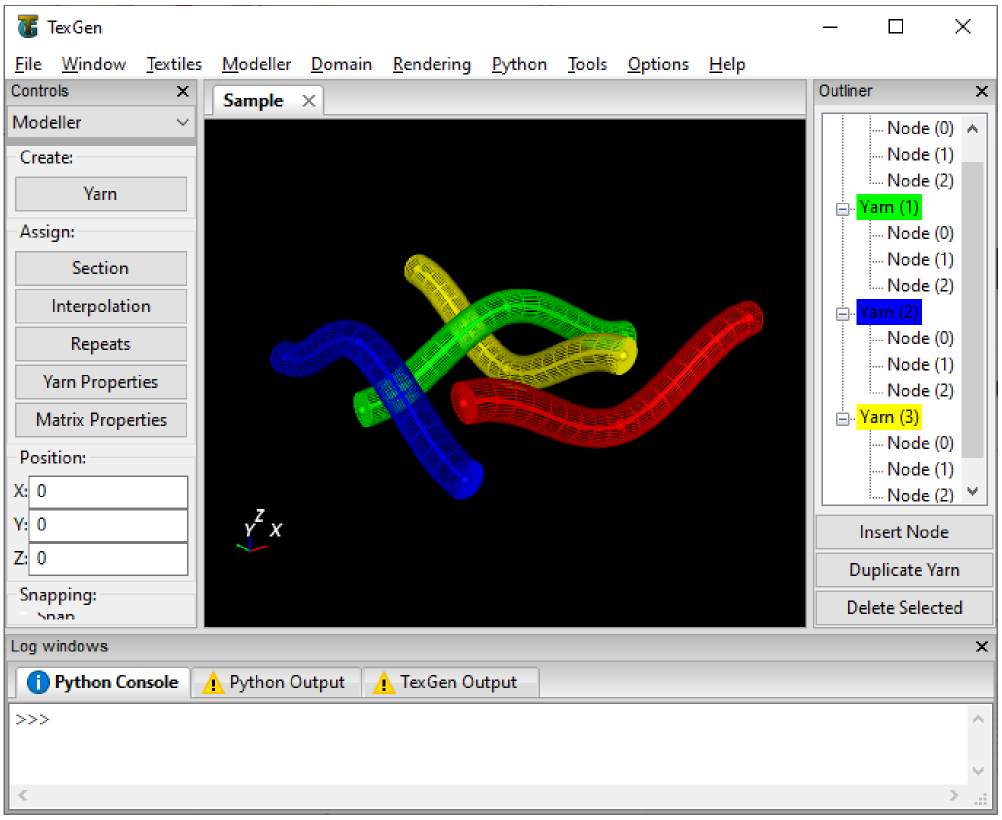
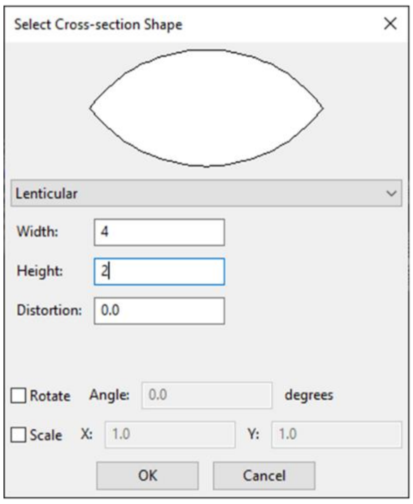


Fig 2.9.13 (a) Cross-section Shape dialog (b) TexGen model with lenticular sections



Select Cross-section Shape

Lenticular

Width:	4
Height:	2
Distortion:	0.0

☐ Rotate Angle: 0.0 degrees

☐ Scale X: 1.0 Y: 1.0

OK Cancel

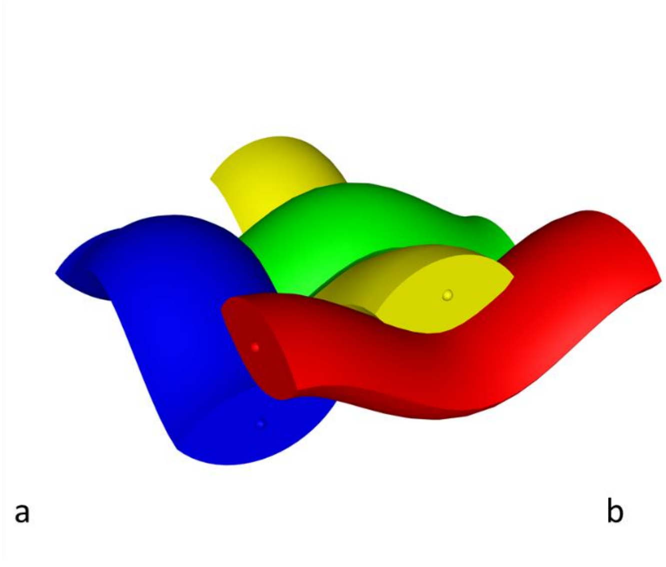


Fig 2.9.14 TexGen model with yarns trimmed to domain

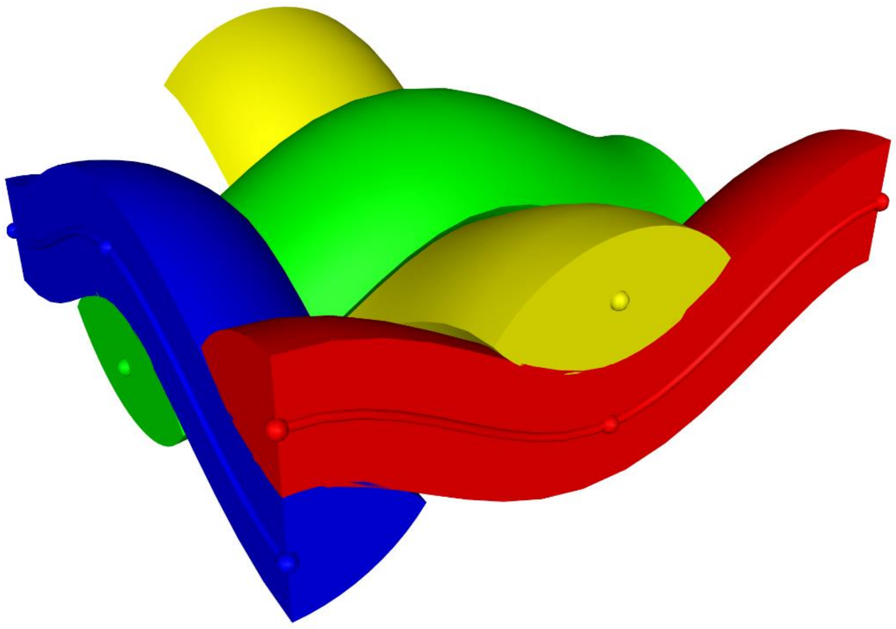
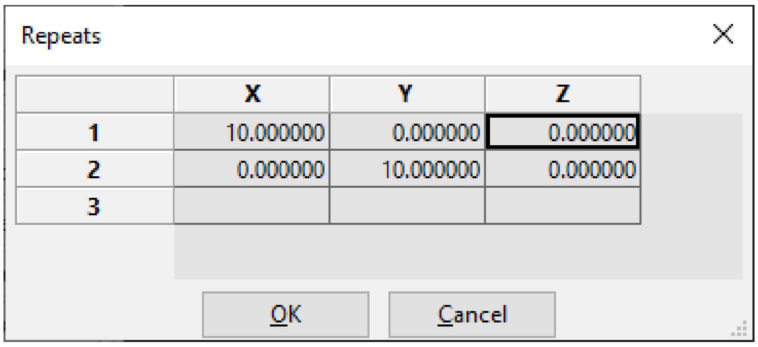


Fig 2.9.15 Yarn Repeat dialog



Repeats

	X	Y	Z
1	10.000000	0.000000	0.000000
2	0.000000	10.000000	0.000000
3			

OK Cancel

Fig 2.9.16 TexGen model with both yarn repeats and domain set

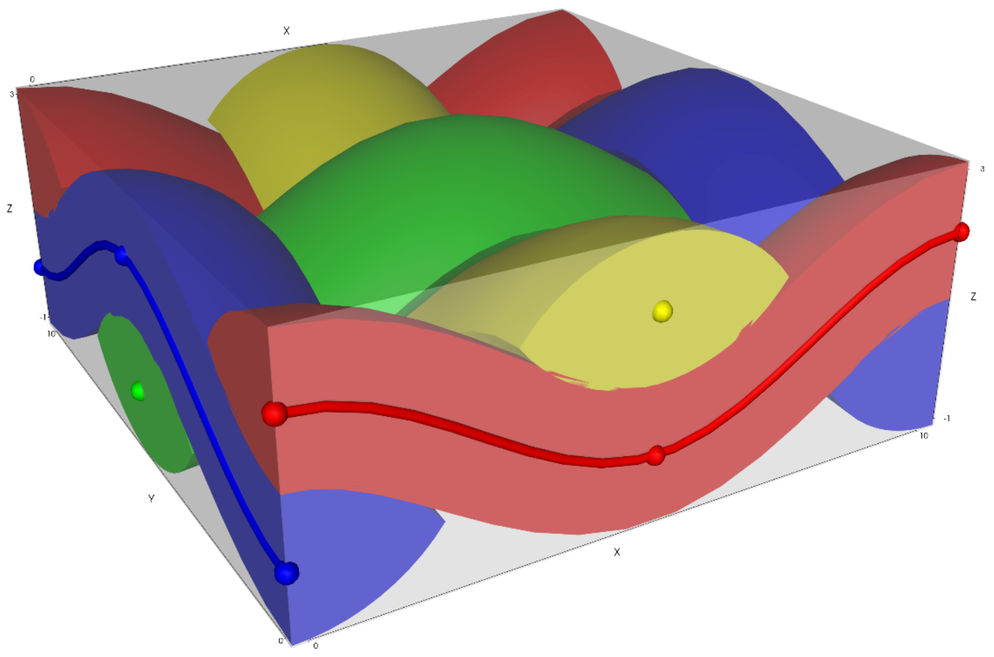


Fig 2.9.17 TexGen model showing interference depths

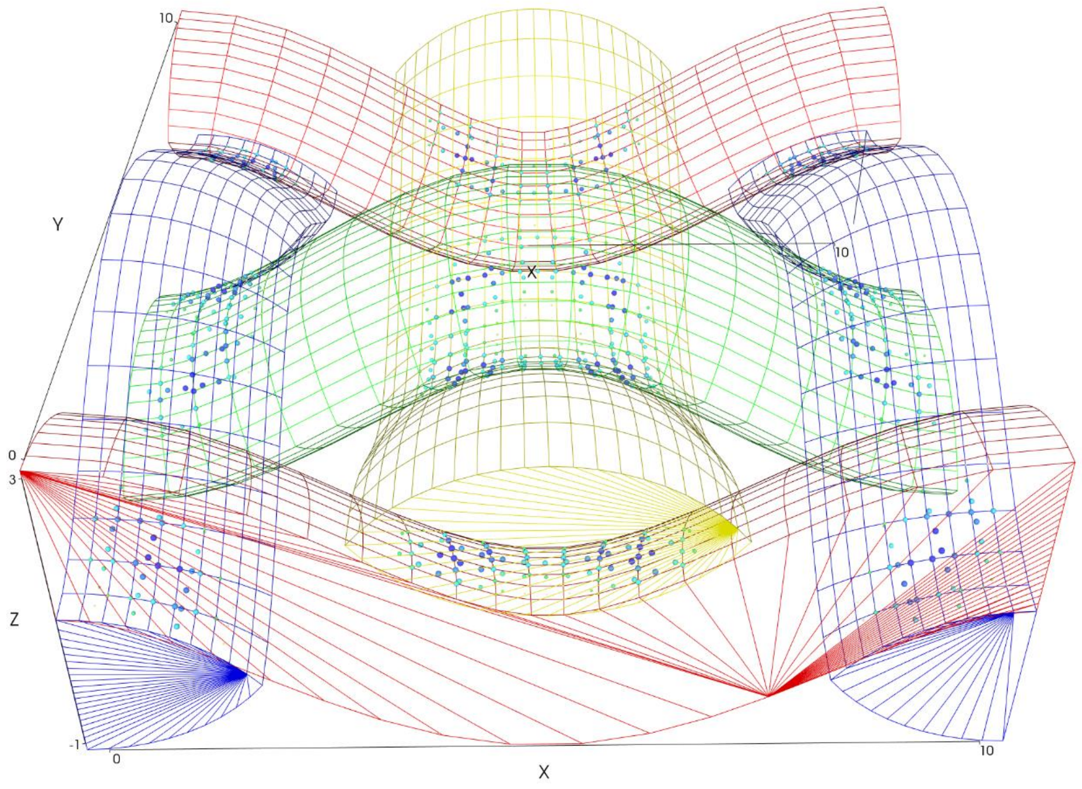
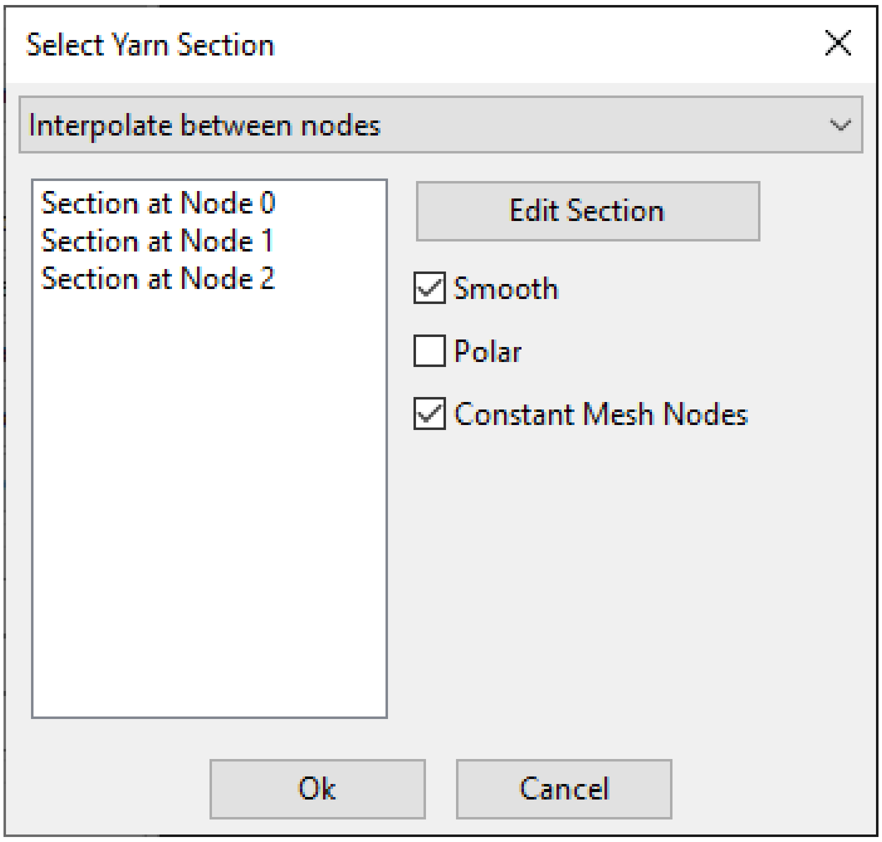


Fig 2.9.18 Select Yarn Section dialog with Interpolate between nodes selection


Select Yarn Section

Interpolate between nodes

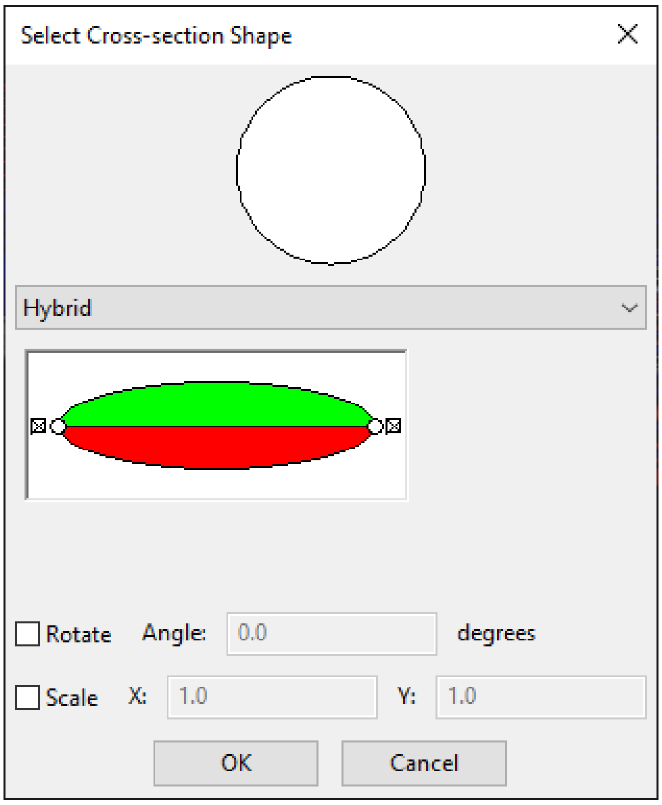
Section at Node 0
Section at Node 1
Section at Node 2

Edit Section

☑ Smooth
☐ Polar
☑ Constant Mesh Nodes

Ok
Cancel

Fig 2.9.19 Select Cross-section Shape dialog showing split for hybrid section

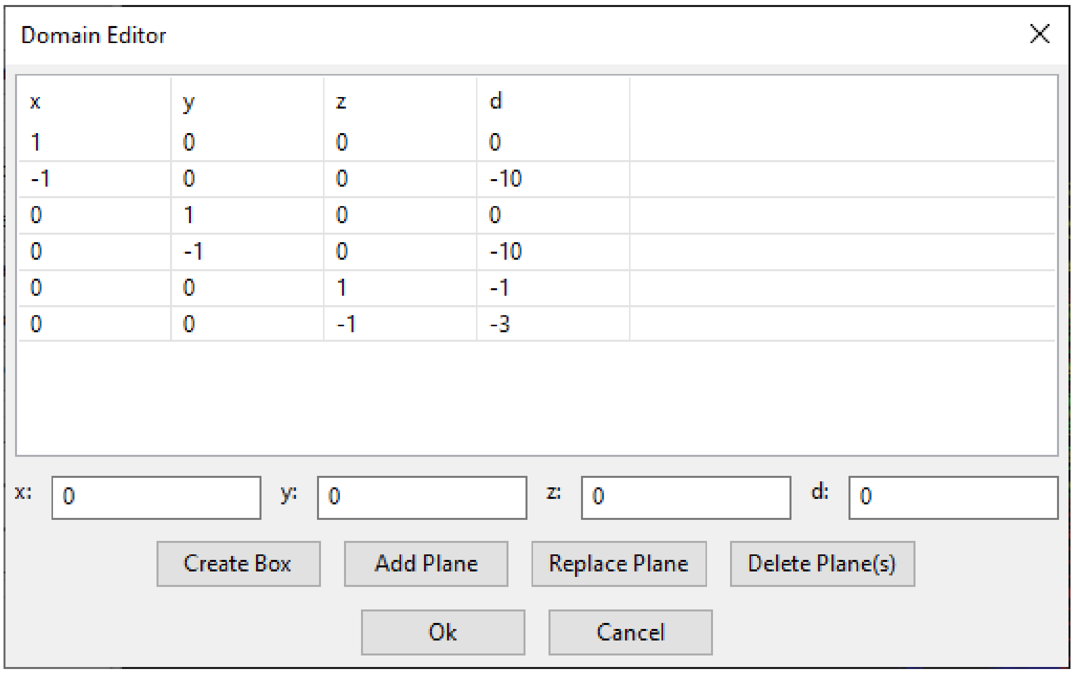

Select Cross-section Shape

Hybrid

☐ Rotate Angle: 0.0 degrees
☐ Scale X: 1.0 Y: 1.0

OK
Cancel

Fig 2.9.20 Domain Editor dialog


Domain Editor

x	y	z	d
1	0	0	0
-1	0	0	-10
0	1	0	0
0	-1	0	-10
0	0	1	-1
0	0	-1	-3

x: 0 y: 0 z: 0 d: 0

Fig 2.9.22 TexGen modules

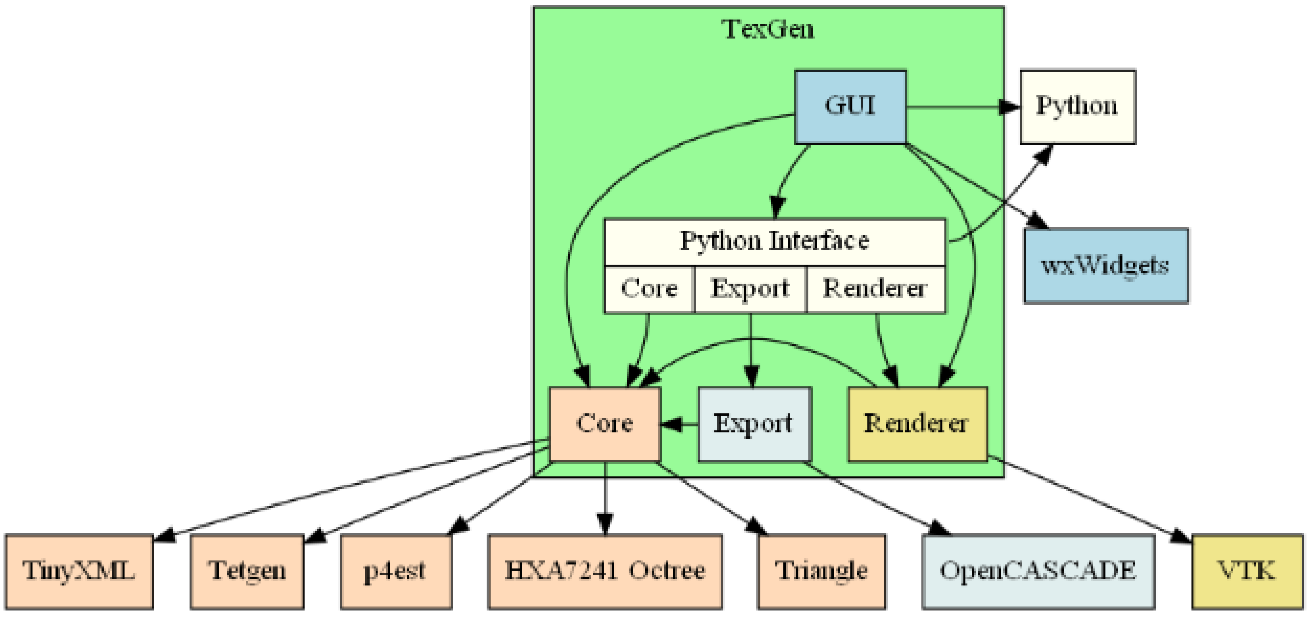


Fig 2.9.23 Textile model generated using Polyester.py script

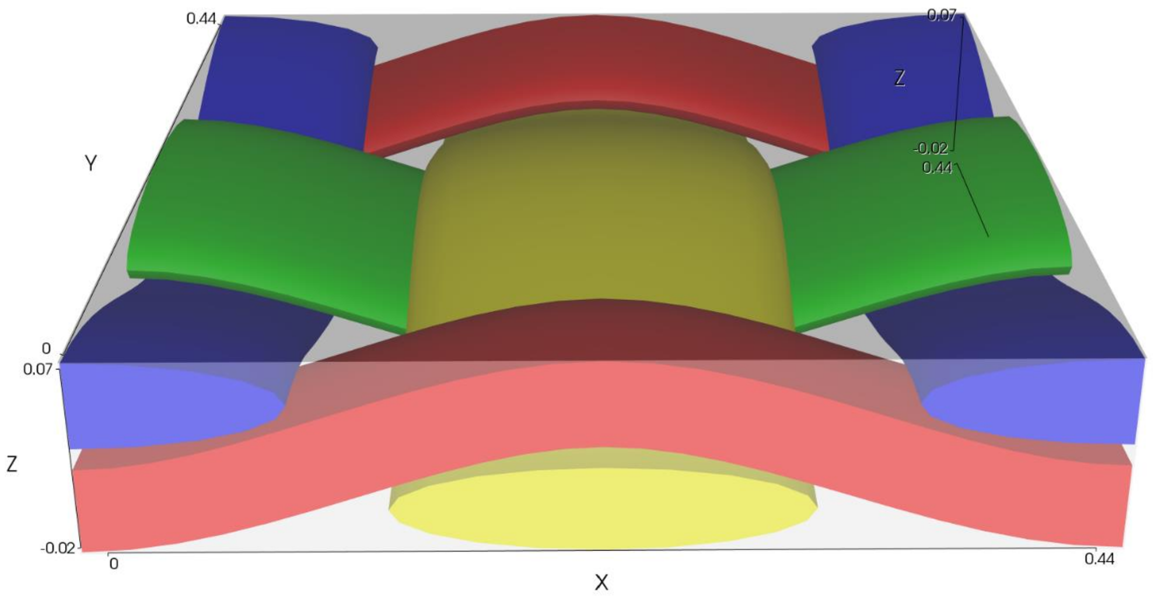


Fig 2.9.24 Dry fibre export mesh

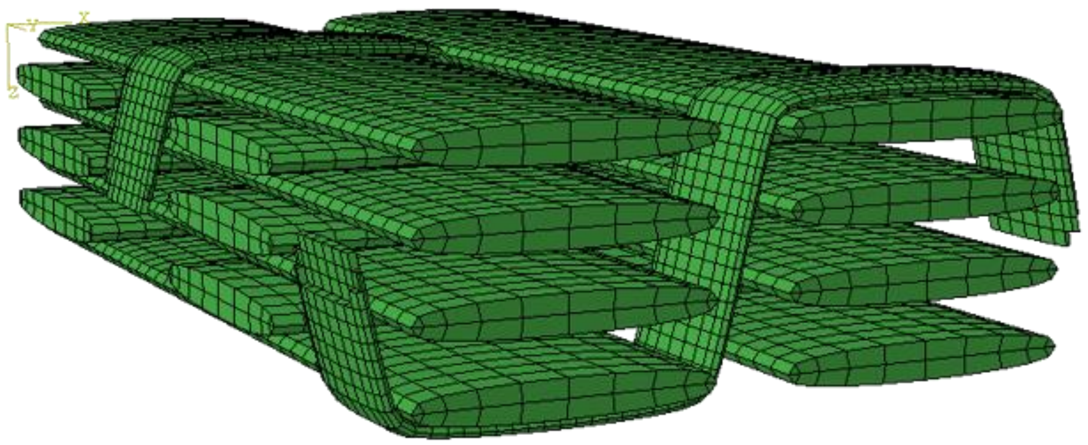


Fig 2.9.25 Tetrahedral mesh export

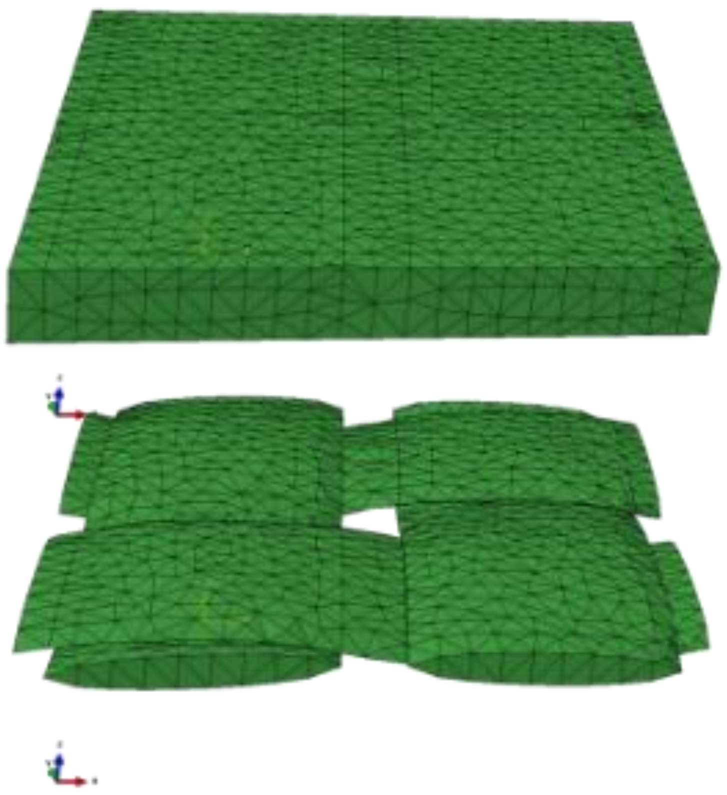


Fig 2.9.26 Voxel mesh export

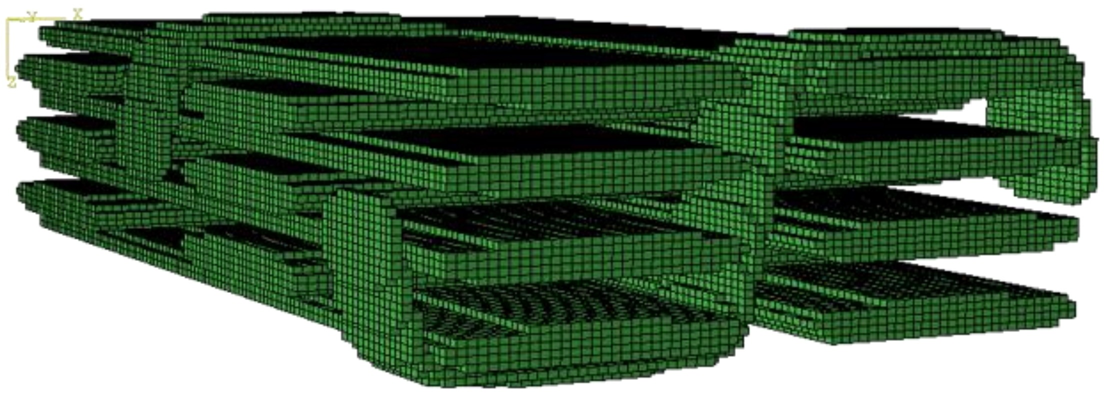
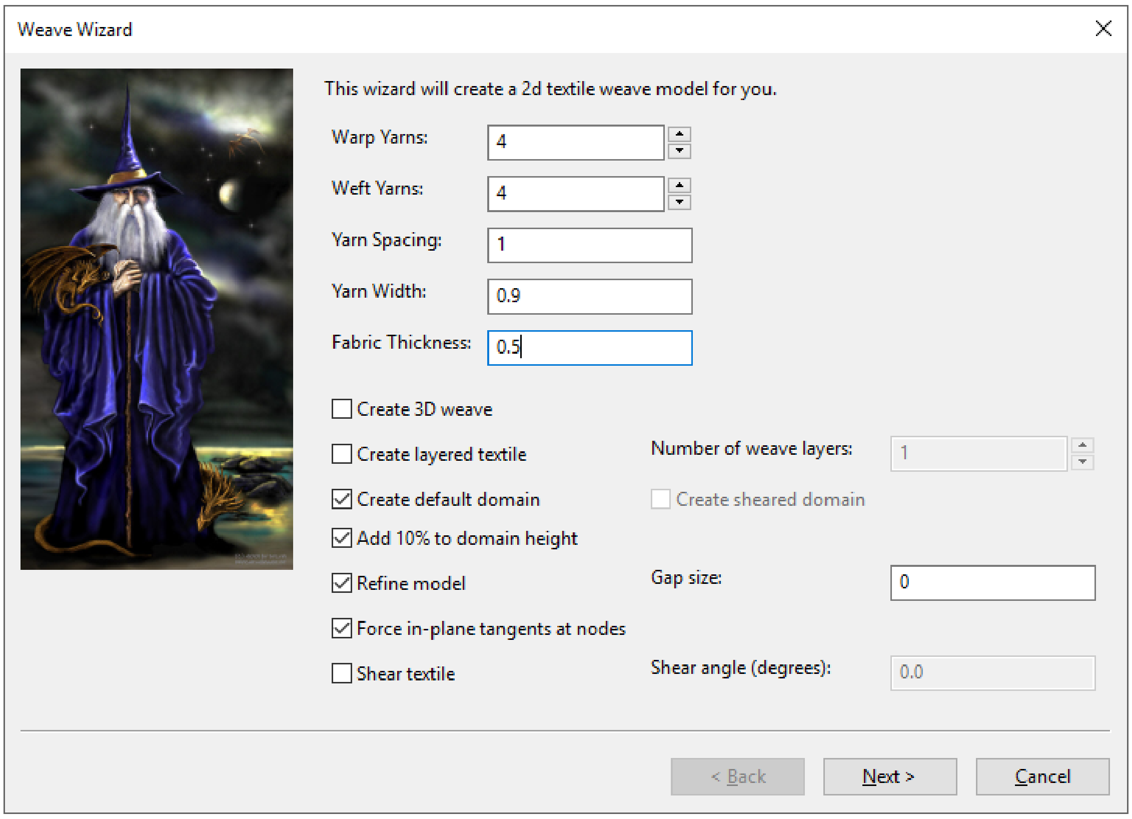


Fig 2.9.27 2D weave wizard yarn parameters



Weave Wizard

This wizard will create a 2d textile weave model for you.

Warp Yarns: 4

Weft Yarns: 4

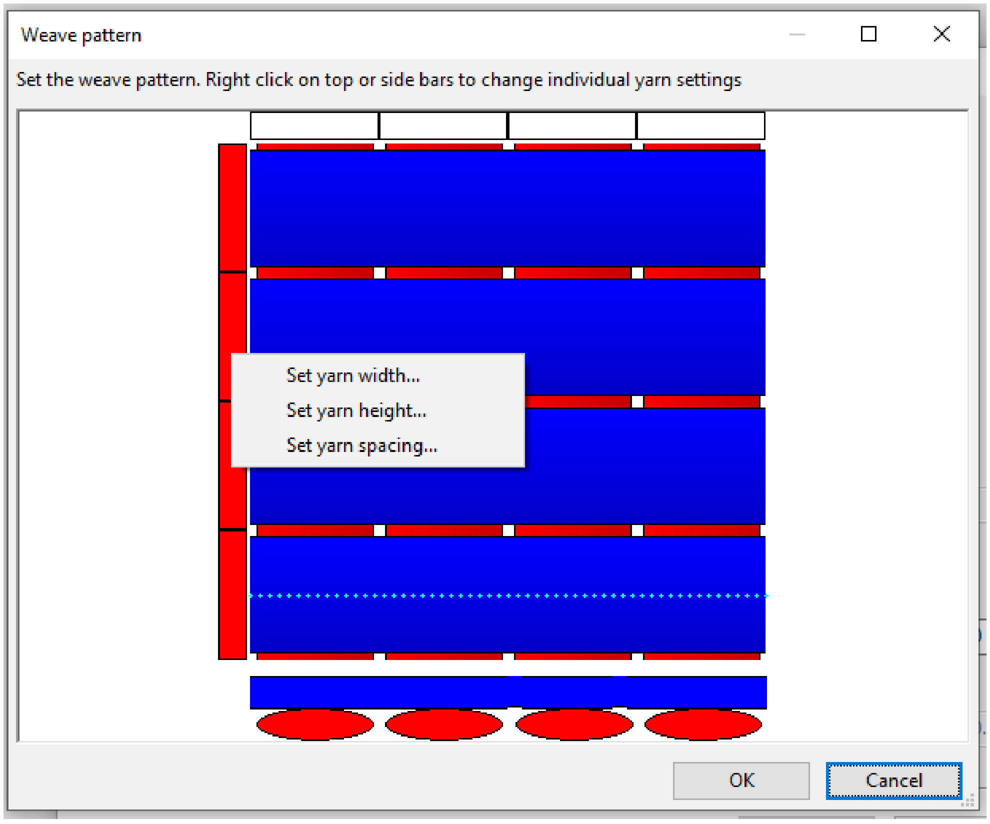
Yarn Spacing: 1

Yarn Width: 0.9

Fabric Thickness: 0.5

- ☐ Create 3D weave
- ☐ Create layered textile — Number of weave layers: 1
- ☑ Create default domain — ☐ Create sheared domain
- ☑ Add 10% to domain height
- ☑ Refine model — Gap size: 0
- ☑ Force in-plane tangents at nodes
- ☐ Shear textile — Shear angle (degrees): 0.0

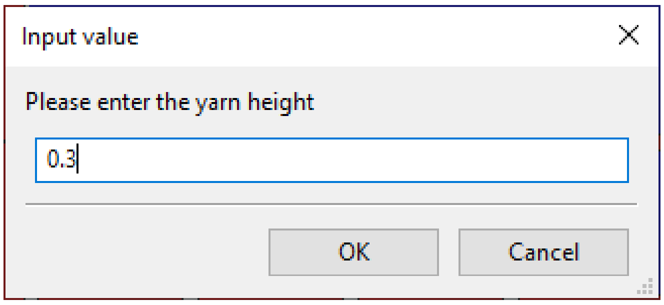
Fig 2.9.28 Options to change selected yarns in Weave Pattern Dialog



Weave pattern

Set the weave pattern. Right click on top or side bars to change individual yarn settings

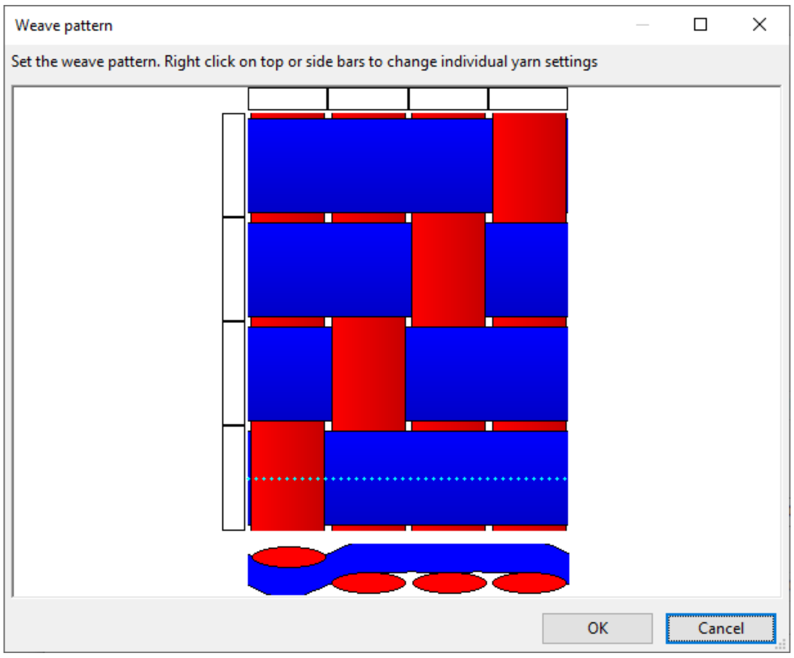
Fig 2.9.29 Dialog to input yarn height for selected yarns


Input value

Please enter the yarn height

0.3

Fig 2.9.30 Weave configuration for Exercise 2.9.1


Weave pattern

Set the weave pattern. Right click on top or side bars to change individual yarn settings

Fig 2.9.31 Exercise 2.9.1, finished textile

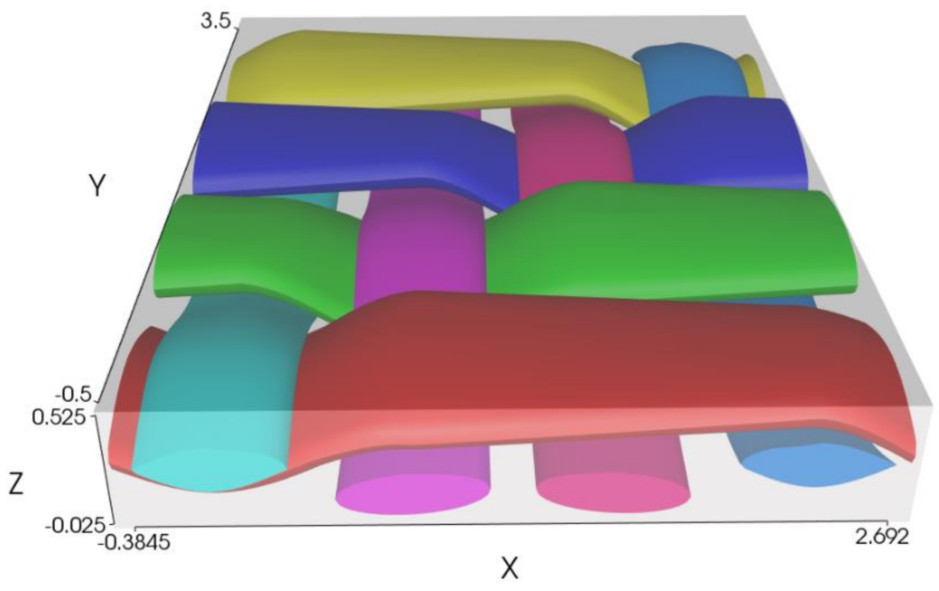
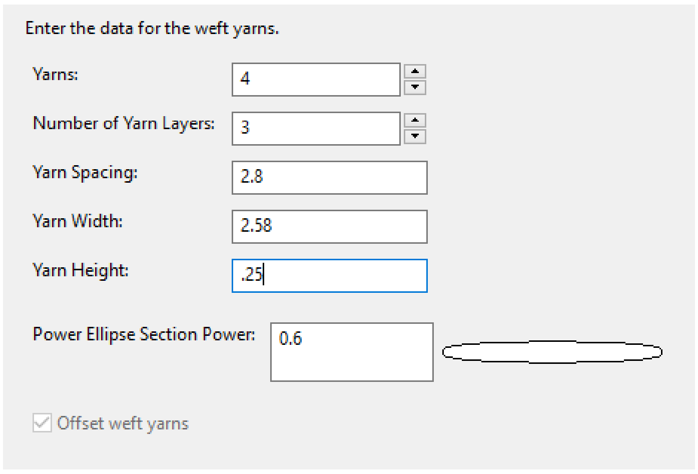


Fig 2.9.32 Weft yarn input for Exercise 2.9.2

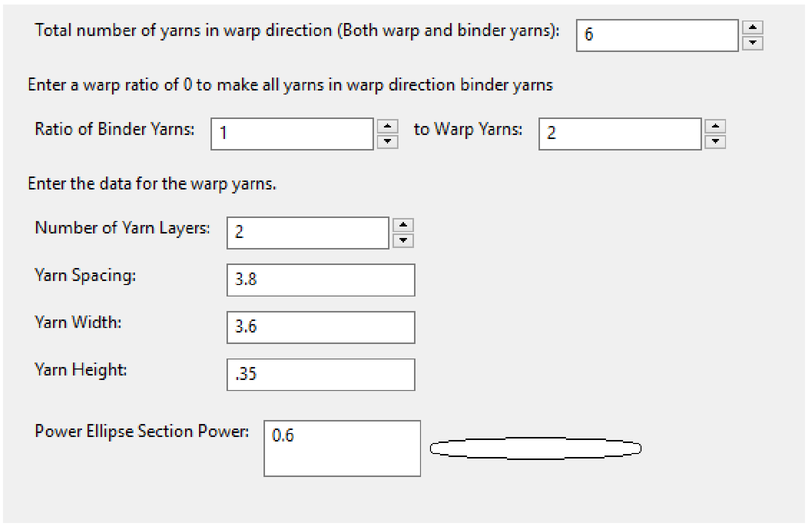


Enter the data for the weft yarns.

Yarns:	4
Number of Yarn Layers:	3
Yarn Spacing:	2.8
Yarn Width:	2.58
Yarn Height:	.25
Power Ellipse Section Power:	0.6

✓ Offset weft yarns

Fig 2.9.33 Warp yarn input for Exercise 2.9.2



Total number of yarns in warp direction (Both warp and binder yarns): 6

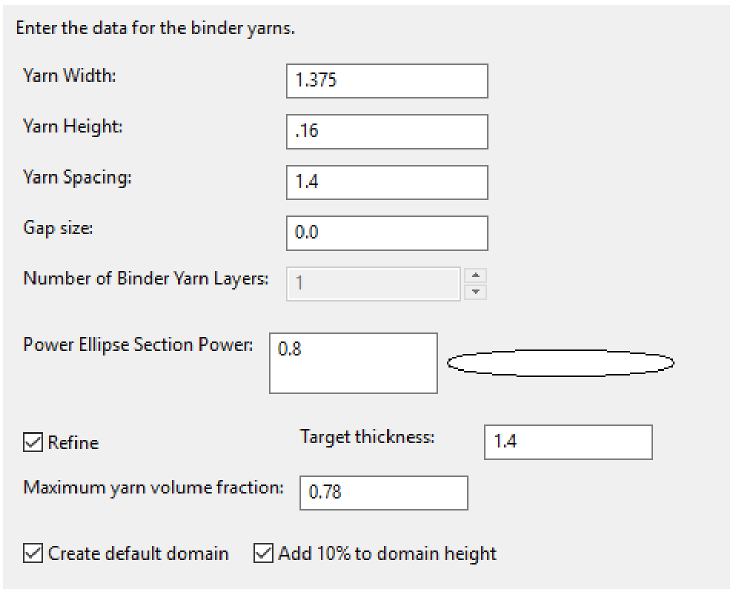
Enter a warp ratio of 0 to make all yarns in warp direction binder yarns

Ratio of Binder Yarns: 1 to Warp Yarns: 2

Enter the data for the warp yarns.

Number of Yarn Layers:	2
Yarn Spacing:	3.8
Yarn Width:	3.6
Yarn Height:	.35
Power Ellipse Section Power:	0.6

Fig 2.9.34 Binder yarn input for Exercise 2.9.2



Enter the data for the binder yarns.

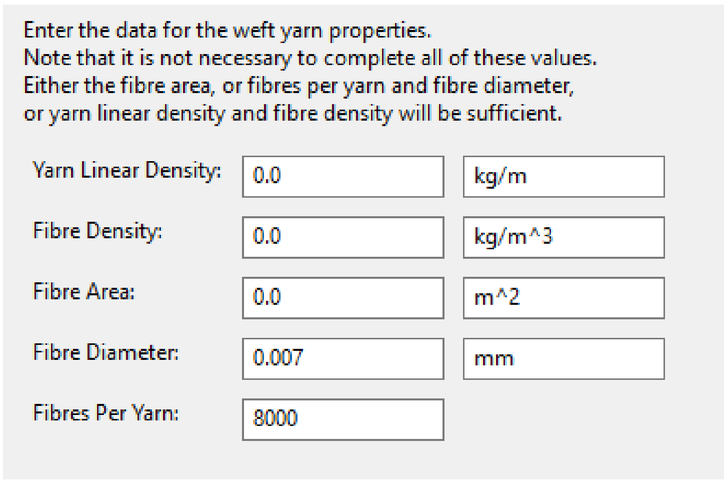
Yarn Width:	1.375
Yarn Height:	.16
Yarn Spacing:	1.4
Gap size:	0.0
Number of Binder Yarn Layers:	1
Power Ellipse Section Power:	0.8

✓ Refine Target thickness: 1.4

Maximum yarn volume fraction: 0.78

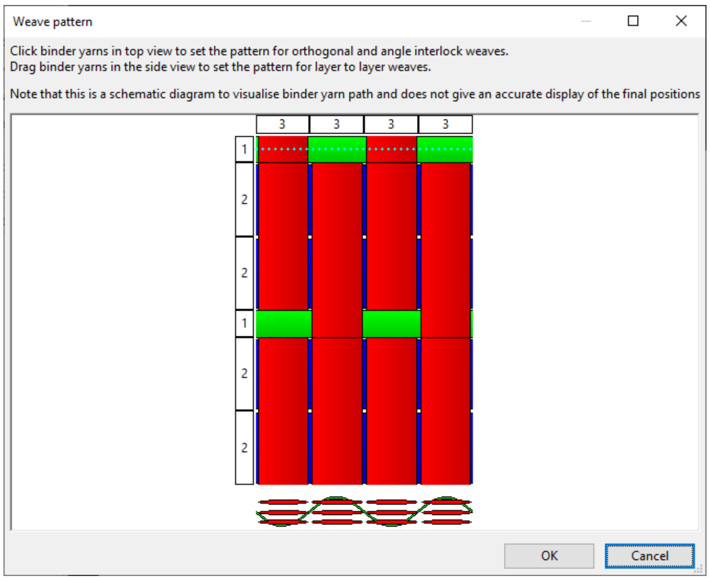
✓ Create default domain ✓ Add 10% to domain height

Fig 2.9.35 Weft yarn properties for Exercise 2.9.2


Enter the data for the weft yarn properties.
Note that it is not necessary to complete all of these values.
Either the fibre area, or fibres per yarn and fibre diameter,
or yarn linear density and fibre density will be sufficient.

Yarn Linear Density:	0.0	kg/m
Fibre Density:	0.0	kg/m^3
Fibre Area:	0.0	m^2
Fibre Diameter:	0.007	mm
Fibres Per Yarn:	8000	

Fig 2.9.36 Weave configuration for Exercise 2.9.2


Weave pattern

Click binder yarns in top view to set the pattern for orthogonal and angle interlock weaves.
Drag binder yarns in the side view to set the pattern for layer to layer weaves.

Note that this is a schematic diagram to visualise binder yarn path and does not give an accurate display of the final positions

Fig 2.9.37 Exercise 2.9.2, finished textile

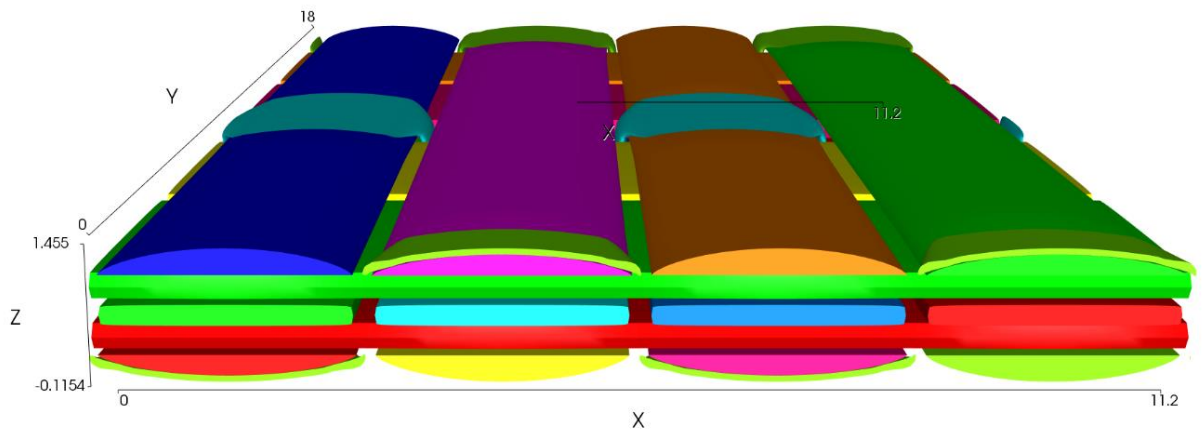
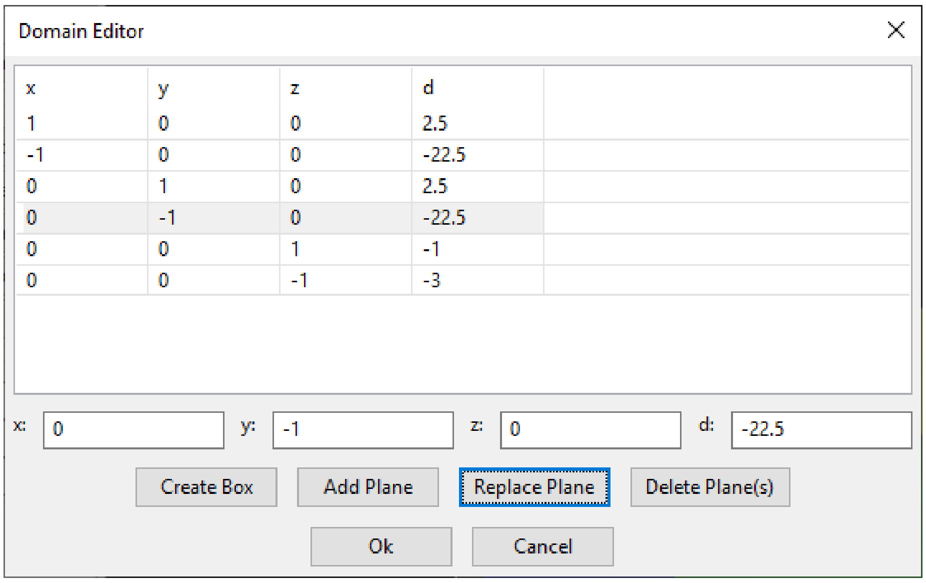


Fig 2.9.38 Domain planes for Exercise 2.9.3



Domain Editor

x	y	z	d
1	0	0	2.5
-1	0	0	-22.5
0	1	0	2.5
0	-1	0	-22.5
0	0	1	-1
0	0	-1	-3

x: 0 y: -1 z: 0 d: -22.5

Fig 2.9.39 Exercise 2.9.3, textile with domain specified for two repeats

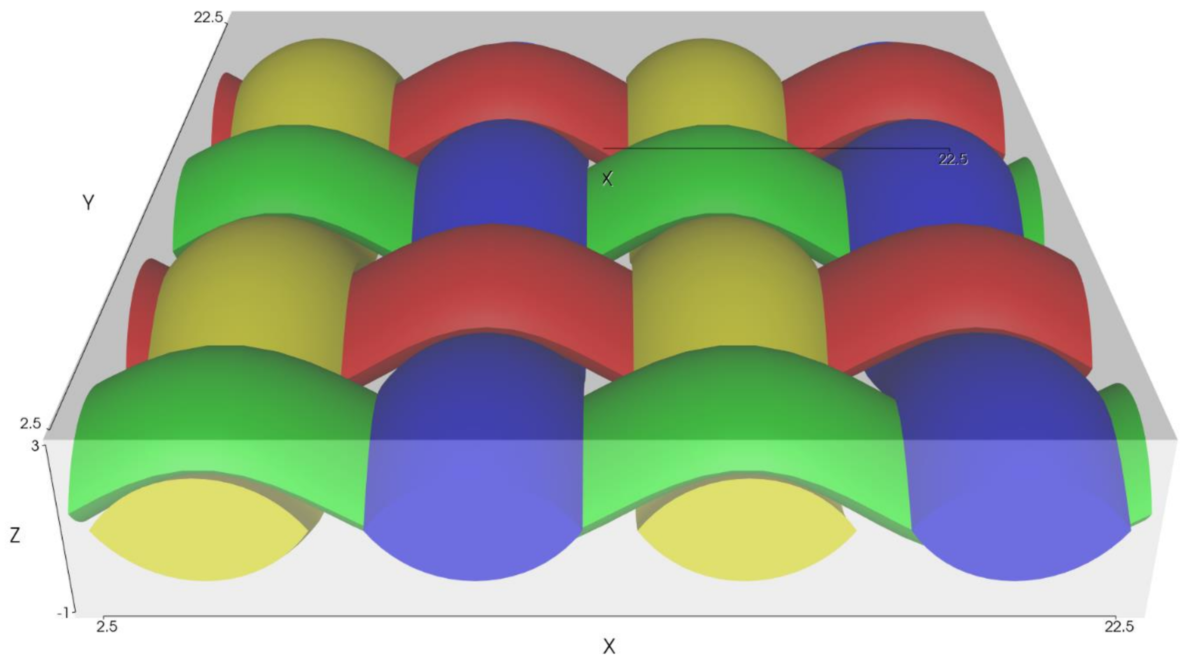
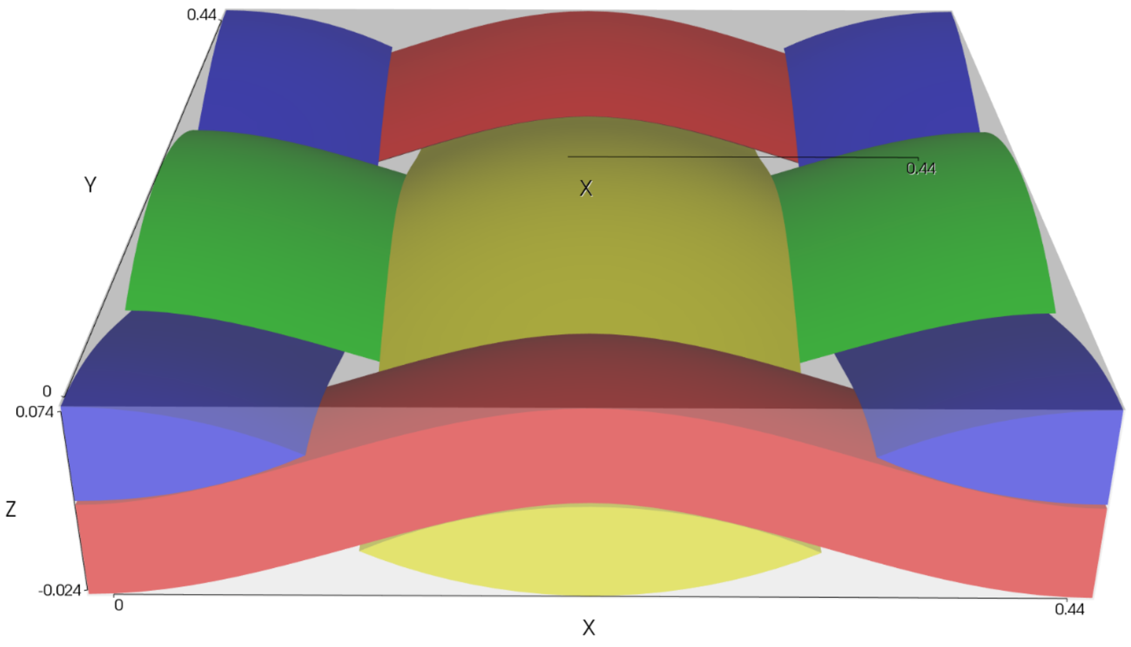


Fig 2.9.40 Exercise 2.9.4, finished textile



Appendix

Exercise 2.9.1 Solution

1. Either select *Textiles->Create Weave...* from the main menu or *Weave* from the *Textiles* tab of the Controls menu.
2. Set the number of warp and weft yarns to 4 (to create a 4x4 satin weave).
3. Use the warp yarn data to set the yarn spacing and yarn width (as these parameters are different for the warp and weft yarns the weft dimensions will be set later using the Pattern Dialog). The *Yarn Spacing* is set to 1.0 based on 10 yarns/cm.
4. The combined heights of the yarns is 0.5 mm so this is selected as the *Fabric Thickness.* The wizard will automatically set both yarn heights to 0.25mm (half the thickness) but, again, these can be adjusted in the Pattern Dialog. Steps 2-4 are shown in Figure 2.9.27.
5. Click the *Next* button to move to the *Weave pattern dialog.*
6. *Shift-click* on the bars at the left side of the weave pattern to select all of the warp yarns.
7. *Right-click* on one of the selected side bars to show options to change yarn parameters for these yarns (Figure 2.9.28)
8. Select *Set yarn height...* and enter the value 0.3 (Figure 2.9.29)
9. Repeat steps 6-8 to select the bars at the top of the weave pattern and set the weft yarn width, height and spacing to 0.7, 0.2 and 0.769 respectively. (The spacing is calculated using 13 yarns/cm)
10. Click on the crossovers on the weave pattern to create the desired weave configuration (Figure 2.9.30)
11. Select *OK.* The textile is created as shown in Figure 2.9.31. The textile can also be loaded from the Ex2_9_1.tg3 file.

Exercise 2.9.2 Solution

1. Either select *Textiles->Create 3D Weave…* from the main menu or *3D Weave* from the *Textiles* tab of the Controls menu.
2. Select the *Orthogonal* weave type and select *Next.*

3. Fill in the weft, warp and binder data as shown in Figures 2.9.32 to 2.9.34. In the binder yarn window select the *Refine* option and set the *Target Thickness* to 1.4.
4. In the three subsequent windows set the yarn properties. This is shown for the weft yarn in Figure 2.9.35.
5. In the final weave pattern window click on the points on the binder yarns to create the desired weave configuration (Figure 2.9.36)
6. Select *OK*. The textile is created as shown in Figure 2.9.37. The textile can also be loaded from the Ex2_9_2.tg3 file.

Exercise 2.9.3 Solution

1. Either select *Domain->Edit Domain...* from the main menu or *Edit* from the *Domain* tab of the Controls menu.
2. The x and y dimensions of the domain are to be changed so the first four planes (ie the first four lines in the dialog) need to be changed. Select the first row. The values from this row will be displayed in the x, y, z and d text boxes. Set the *d* value to 2.5 and select *Replace Plane.* Note that the values in the main window are not updated until *Replace Plane* is selected.
3. Repeat step 2 for the next 3 planes as shown in Figure 2.9.38.
4. Select *OK*. The size of the domain will be updated and the newly specified region of the textile will be displayed as shown in Figure 2.9.39. The updated textile can be found in Ex2_9_3.tg3.

Exercise 2.9.4 Solution

The amended script can be found in Ex2_9_4.py and the resulting textile is shown in Figure 2.9.40. The changes are as follows:

1 & 2.
```
# Assign a constant cross-section along the yarn of lenticular shape
Yarn.AssignSection(CYarnSectionConstant(CSectionLenticular(0.2, 0.048)))
```

3.
```
# Set the resolution of the mesh created
Yarn.SetResolution(40)
```

4.
```
# Create a domain and assign it to the textile
Textile.AssignDomain(CDomainPlanes(XYZ(0, 0, -0.024), XYZ(0.44, 0.44, 0.074)))
```