Interlaced Composites Geometry Creation in Abaqus

Python version: 2.7.xx (I have 2.7.15 installed locally, Abaqus 2016 uses version 2.7.3)

Required packages: Shapely – I’ve attached the .whl file. To install, simply navigate to the directory where the file is stored and then enter: pip install Shapely-1.6.4.whl

See: <https://stackoverflow.com/questions/27885397/how-do-i-install-a-python-package-with-a-whl-file>

You may need to modify the path edits at the top of the shapelyToAbaqusExplicit module to point to the directories where your Python packages are installed.

There are three modules used to create the interlaced geometries, sigc (shapely interlaced geometry creation), tapePlacement, and shapelyToAbaqusExplicit. The first script implements the underlying logic to determine tape paths through the laminate. The tapePlacement module simply defines where the tapes will be placed. The last module takes the grid of cells and converts them to Abaqus parts, as well as defining the rest of the model.

The objPlot module simply plots the cells/grids in matplotlib. It is only used for debugging and visualization pre-Abaqus (Abaqus does not play nice with matplotlib).

**Sigc script:**

Part 1: Grid creation

1. Define polygon corresponding to specimen dimensions. For an impact specimen: 100 mm x 150 mm (origin is at the center of the panel)
2. Define the width of tapes used to create a laminate (e.g. 10mm), and their angles (e.g. 0/90/45) as well as the width of the resin rich region bounding the tape along its length. See below:

“Resin” regions

Tape region

These regions are needed to define the undulations as will become evident later.

1. Starting from the origin, split the specimen polygon created in step 1 into a grid of polygons according to the tape dimensions and angles defined in step 2. Each polygon is assigned a unique ID.

Figures 1-3 below illustrate the grids for various tape angle combinations (all 10mm width tapes)

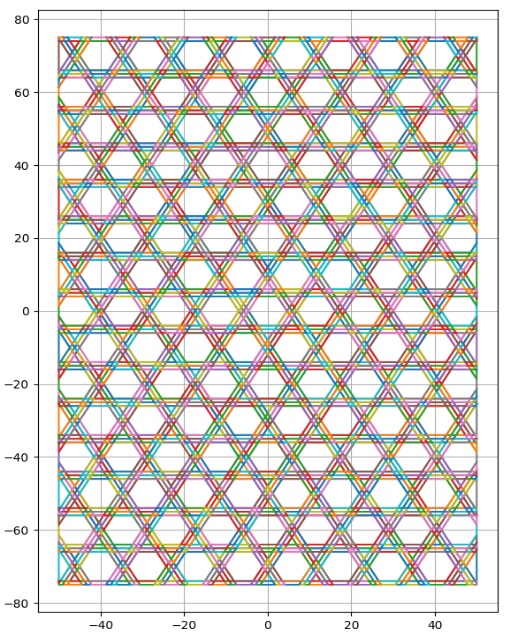
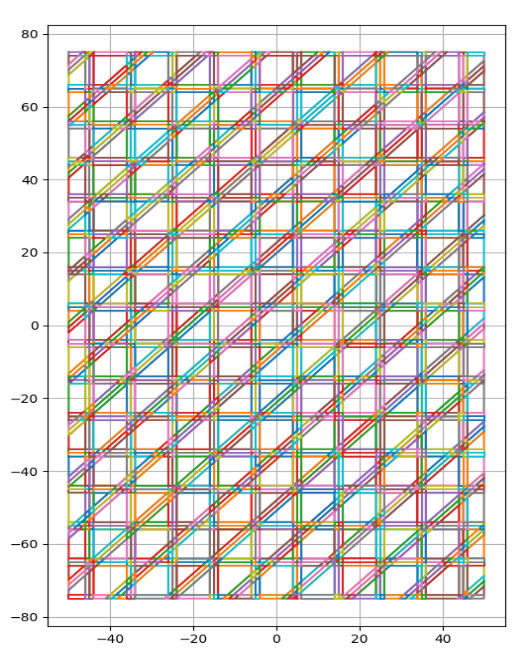
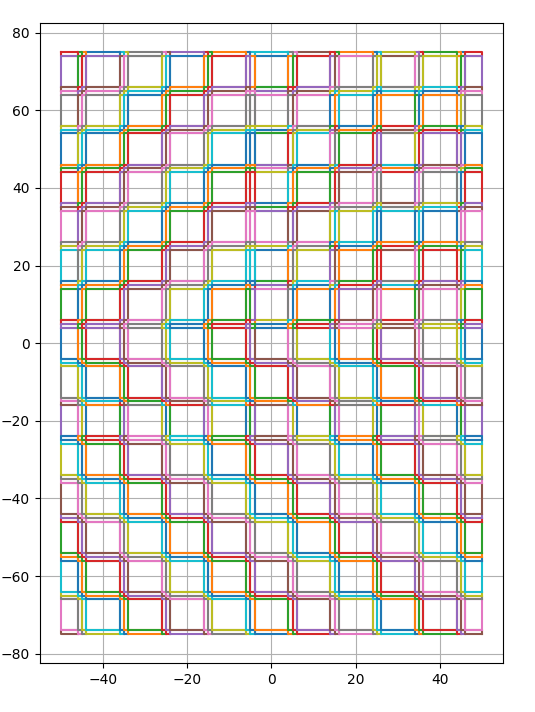


Figure 1-3. Polygon grids for laminate angle sets: (0/90), (0,60,-60),(0,45 ,90)

1. Create a dictionary containing grids for each layer of the laminate. The number of layers will depend on the number of tape angles, e.g. a 0/90 laminate will only have 2 layers. Polygon IDs correspond to the same polygon in each layer.
2. An ‘objectType’ attribute is created for each Polygon instance in the grid. Its initial value is None.

Part 2: Tape Placement

1. The order in which tape angles are defined determines the order in which tapes are placed to create the laminate. So in a (0/90/45) laminate, the 0 degree tape is placed first, followed by the 90 degree tape, and finally the 45 degree tape.
2. A polygon is created with coordinates dependent on the tape dimensions and angle. Then, the program checks which of the polygon regions in the first layer grid (defined in part 1) are within the boundaries of the tape. A list of these regions is created which is called tapePath.
3. The program then iterates through each polygon object in tapePath.

* Grid polygons within the boundaries of the tape polygon whose ‘objectType’ attribute == None. are assigned the objectType: ‘Tape’. If a grid polygon’s objectType is None it means it has not yet been assigned, so in essence it means there is not already another tape in that position.
* If a grid polygon within the boundaries of the tape polygon *has already been assigned an objectType other than None*, it means a tape has already been placed in that space in that layer. When this situation is encountered, the program will check whether *the* ***same*** *grid polygon in the* ***next*** *layer* has been assigned an objectType other than None (this search is done using the polygon ID assigned in Part 1 Step 3).
  + If the objectType of the grid polygon in the next layer is None, then it (the grid polygon in layer 2) is assigned objectType = ‘Tape’.
  + If the objectType of the grid polygon in the next layer **is not** None, then the program, in exactly the same way as before, checks whether the grid polygon in the next layer has objectType == None. This process continues until the tape can be placed in an empty spot in the dictionary of grids.

1. Resin regions bordering the tape region are created in the same way except for the fact their objectType attribute is set to ‘Resin’.
2. The same process of checking for empty cells is repeated for the resin regions. They are moved up layer by layer if they intersect with a previously placed tape or resin region.

Part 3: Undulation detection and assignment

Undulation regions are region in which the tapes undulate from one layer to the next. Detection and angle assignment in these regions is accomplished as follows:

1. The tapePath list is sorted into sublists; each sublist contains all the Tape and Resin objects in a layer.
2. For each layer, the objects in that layer are merged into one polygon
3. The buffer method is then applied to the union of the polygon to expand its boundaries by the same amount as the undulation region width
4. The program then iterates through the list of objects in the layer above the current layer and checks which of these objects is within the boundaries of the merged and buffered bottom polygon.
5. If the object is within the boundaries, then **both** the polygon object in the bottom and the top layer are assigned the objectType ‘Undulation’.
6. The setAngle function is used to assign the material orientation angle. It is a combination of the angle of the tape making the undulation, and the angle of the tape under the undulating tape.

**The tapePlacement script**

1. The laminateCreation function calculates the coordinates of each of the tapes to be placed on the laminate (depending on their width, angle, and spacing), then calls machinesPass from sigc to place the tapes in the grid.

Note that coordinates for the machinePass function are always for horizontal tapes! Tape coordinates are rotated to the desired tape angle internally within the machinePass \_\_init\_\_ method.

**shapelyToAbaqusExplicit script**

This script imports the sigc and tapeplacement modules to create a complete set of layer grids. Each polygon is then created as a part in Abaqus.

1. Layer grids are created by invoking sigc.createGrids
2. Grids are populated by invoking tp.laminateCreation
3. Model created, materials defined (ignore the values for now, they are just placeholders), step created, fieldOutputs intervals set.
4. The definePart function:
   1. takes a polygon from the grid, extracts its coordinates
   2. extrudes it to a depth defined by variable cpt (cured ply thickness)
   3. Assigns a section and material orientation
   4. Creates the instance in the model assembly and translates it to its correct position
5. After the parts are created, the program merges all the parts that constitute one Tape into a single instance.
6. Contact properties are defined
7. The contact detection function is used to determine the surface to surface contacts between parts. This creates a large number of individual surface to surface interactions which Abaqus Explicit does not allow, so…
8. The program iterates over the individual interactions, and transfers them to ‘Individual Property Assignments, under a general contact interaction. The individual interactions are then deleted.
9. Boundary condition assignment, coupling etc. etc.