MECH 151L: Finite Element Theory and Applications
Lab Two: Auxetic Structures and Periodic Boundary Conditions

German Markaryan

October 30, 2025

Abstract

The experiment goal is to investigate how voids can transform a structure with a positive Poisson's ratio into an auxetic structure with a negative Poisson's ratio for a 40 mm by 40 mm plate. Finite element analysis was executed in Abaqus CAE to determine the Poisson's ratio of the structure with porosities of 2% and 5% with the hole ratio varying in a specific array between 1 and 40. Results showed the 5% porosity structure became auxetic at an aspect ratio of 15, while for 2% porosity, that number was 30. This means that higher porosity increases the slope of the Poisson's ratio of the structure towards the negative Poisson's ratio. The key finding of this experiment was that the effective Poisson's ratio is directly dependent on the Lmin/Lo ratio.

1. Introduction

Auxetic structure has unique physical properties. Their Poisson's ratio is negative, and this means that if they are compressed, they will get longer in the axes of the applied force and get smaller in perpendicular axes. The Poisson's ratio is defined as

$$\nu = -\frac{\varepsilon_{xx}}{\varepsilon_{yy}}$$

where ε_{xx} is a strain in x-axis and ε_{yy} is strain in y-axis. The material of the auxetic structure used in the experiment has a Poisson's ratio = 0.03. The target for this work is to calculate a Poisson's ratio of the entire structure, defined as

$$\bar{\nu} = -\frac{\bar{\varepsilon}_{xx}}{\bar{\varepsilon}_{yy}} = -\frac{u_x^{Vp_x}}{u_y^{Vp_y}}$$

where $u_x^{Vp_x}$ is a displacement of the virtual node computed using finite element analysis using Abaqus CAE on x-axis and $u_y^{Vp_y}$ is a displacement specified to be (-0.005).
The tructure used for this experiment is a square plate with a size of 40 mm by 40mm that has four holes in it. Those holes are defined using porosity and aspect ratio. Porosity varies only between 2% and 5% but aspect ratio varies between 1, 5, 10, 15, 20, 25, 30, 35, 40.

$$\frac{L_{\min}}{L_o} = \frac{1}{2}\left[1 - \left(1 + \frac{a}{b}\right)\sqrt{\frac{b\,\psi}{a\,\pi}}\right]$$

where ψ is the porosity, b is the smaller length of the hole, and a is the longer length of the hole.

The main idea behind the experiments is that holes in a specific location and aspect ratio can create auxetic structures even though the material used has a positive Poisson's ratio. The expected observation is that the more holes influence the structure of the plate, the more auxetic the structure becomes. This means that by increasing porosity or aspect ratio, Poisson's ratio will decrease and eventually become negative.

2. Results and Discussion

The first part contains simulations of the square plate with a porosity of 0.02. Figure 1 shows a specimen with an aspect ratio of 40, and this means that it is expected to be an auxetic structure due to its influence on the square plate. Figure 2 shows the same specimen but with the holes whose aspect ratio is equal to 1, and hence they are

circular holes. This means that they should behave the same way as the material because their influence on the specimen structure is insignificant. This means that the aspect ratio of 1 must have a positive Poisson's ratio.

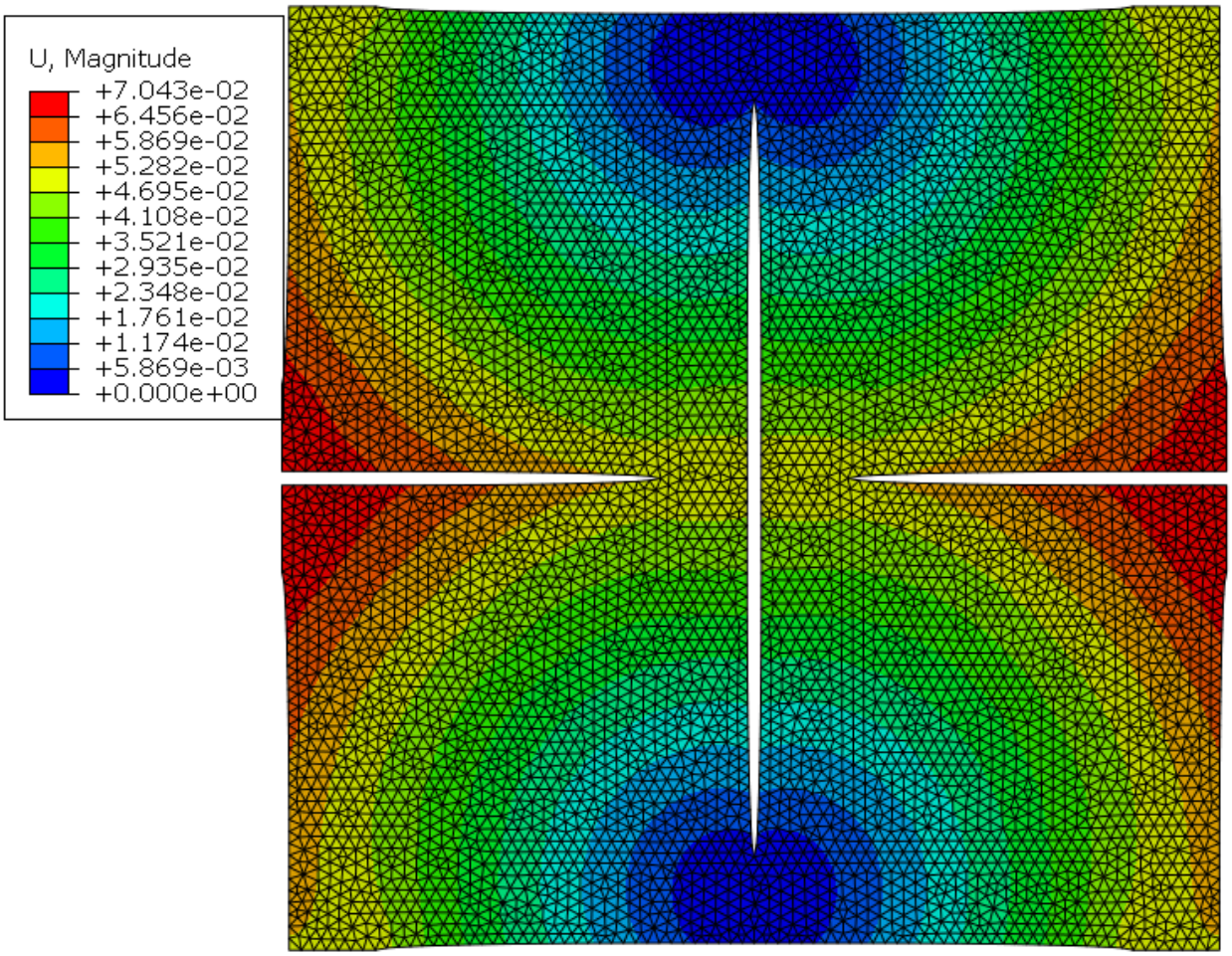

Figure 1: The deformation magnitude for a 40mm by 40 mm square plate with a porosity of 2% and aspect ratio of 40

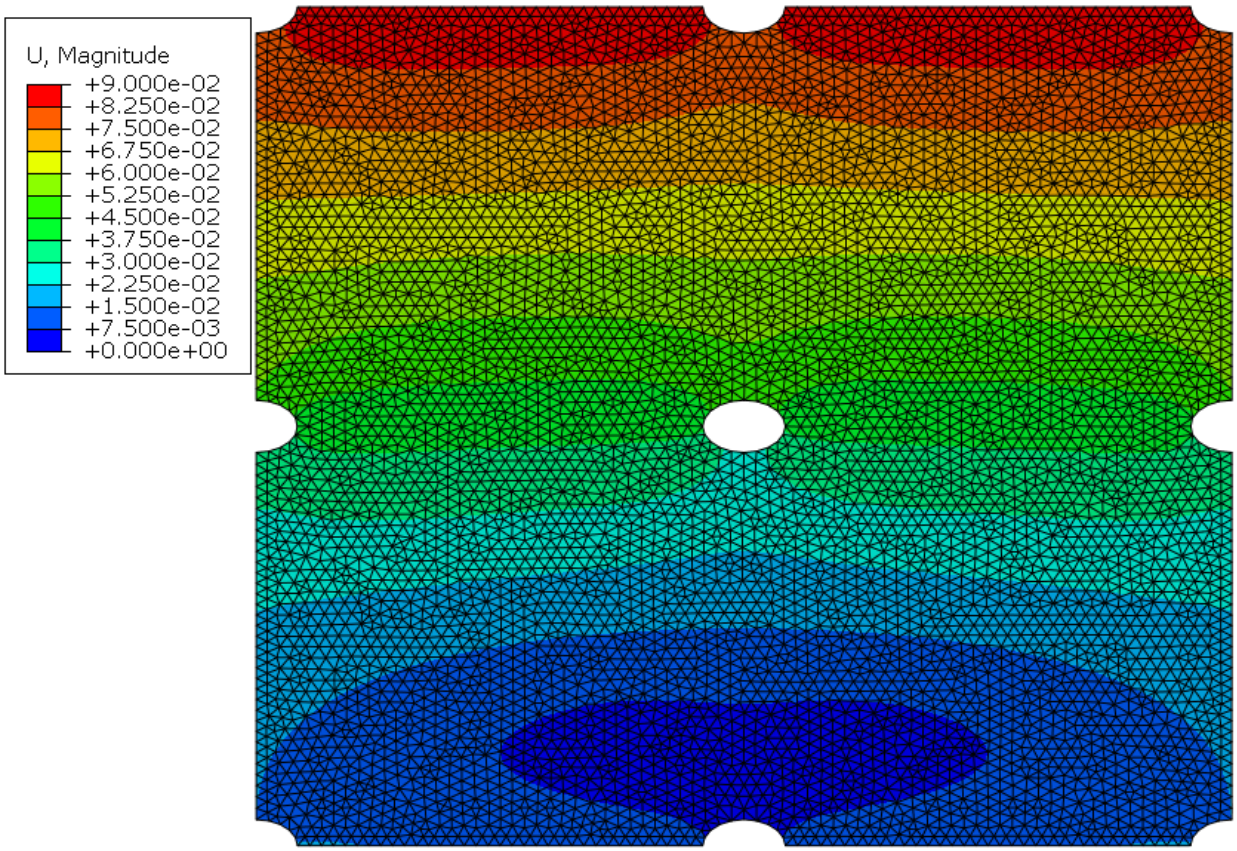

Figure 2: The deformation magnitude for a 40mm by 40 mm square plate with a porosity of 2% and aspect ratio of 1

The second part contains simulations of the square plate with a porosity of 0.05. Figure 1 shows a specimen with an aspect ratio of 40, and this means that it is expected to be an auxetic structure due to its influence on the square plate. Figure 2 shows the same specimen but with the holes whose aspect ratio is equal to 1, and hence they are circular holes. This means that they should behave the same way as the material because their influence on the specimen structure is insignificant. This means that the aspect ratio of 1 must have a positive Poisson's ratio.

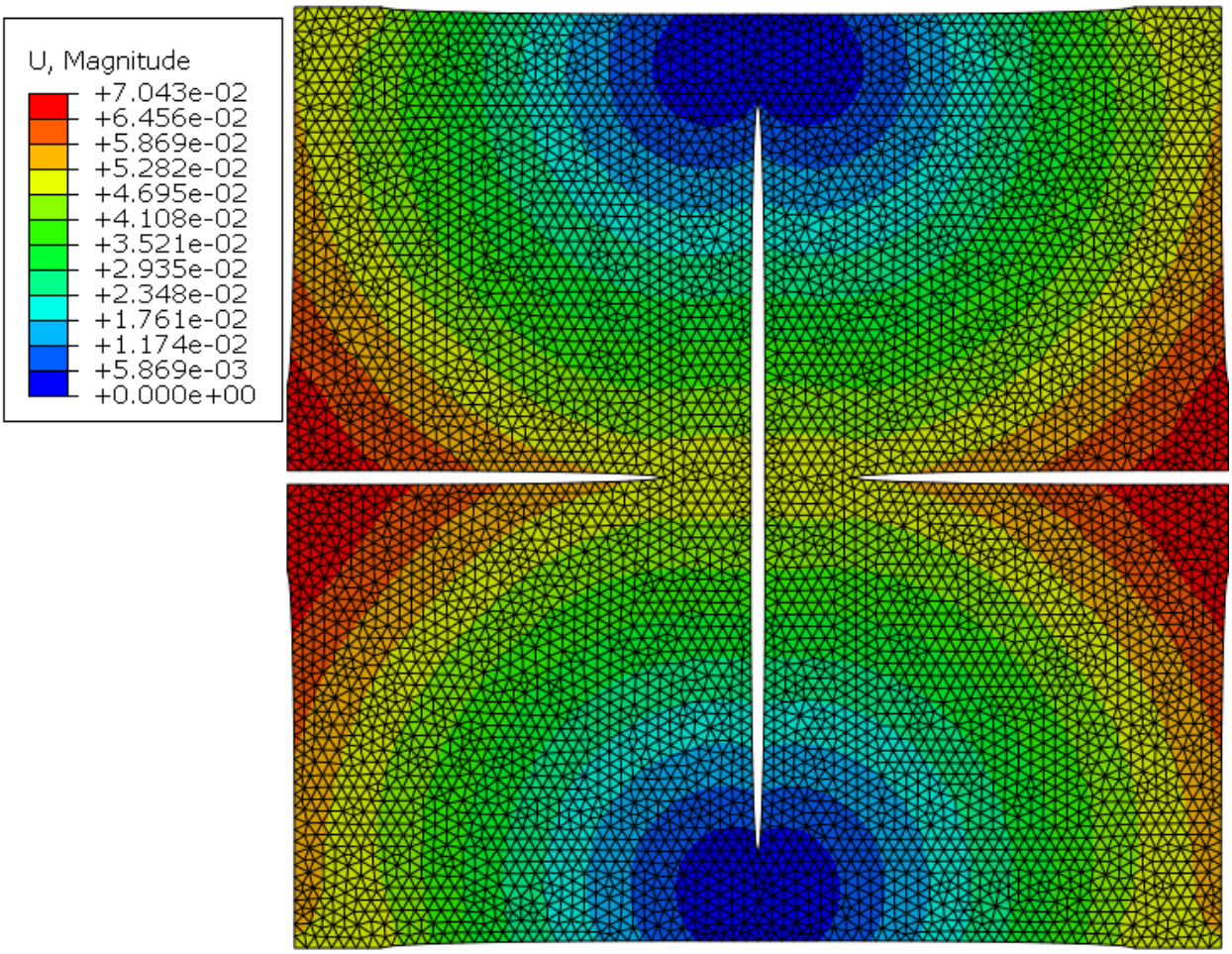

Figure 3: The deformation magnitude for a 40mm by 40 mm square plate with a porosity of 5% and aspect ratio of 40

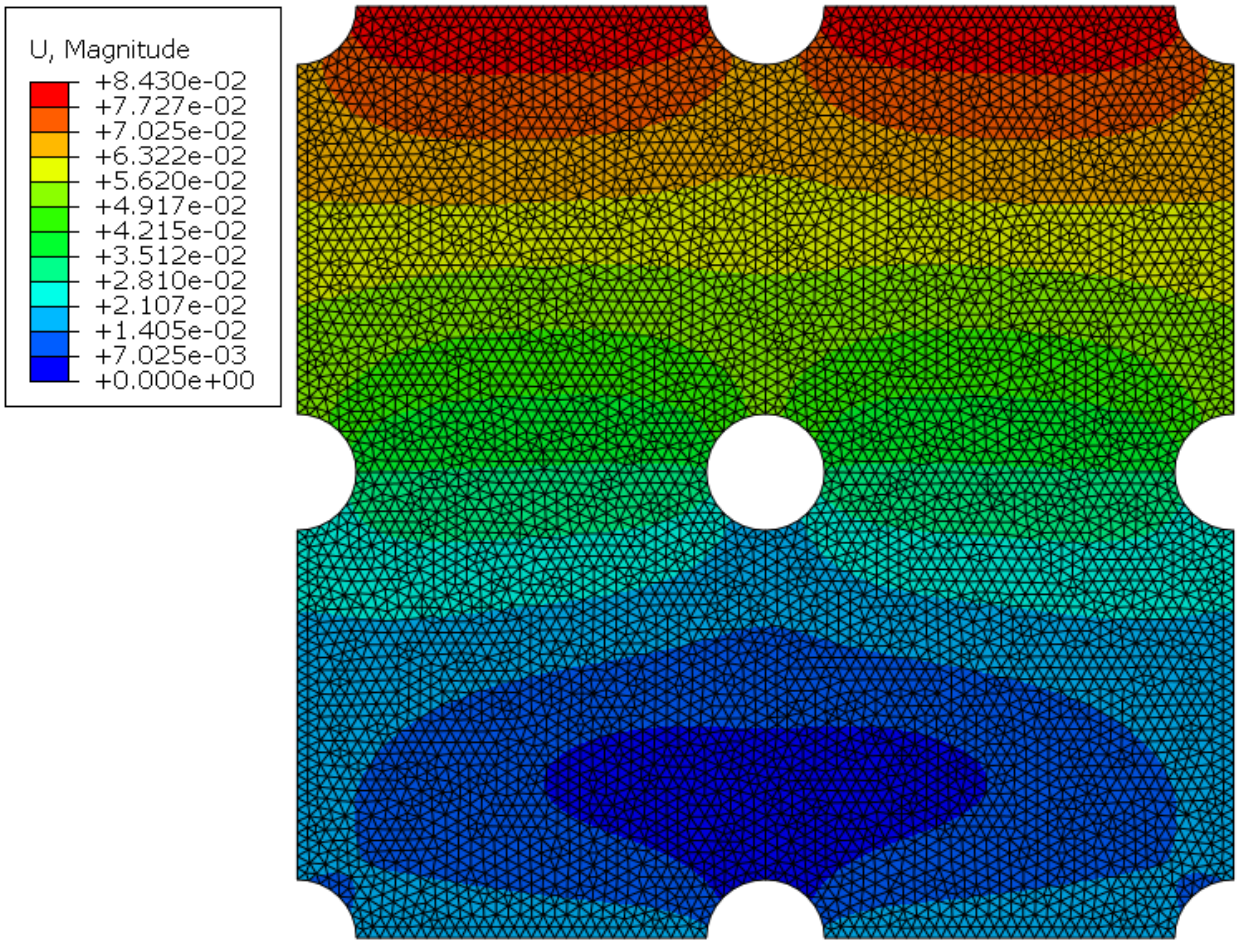

Figure 4: The deformation magnitude for a 40mm by 40 mm square plate with a porosity of 5% and aspect ratio of 1

Abacus CAE finite element analysis of the specimen with all different parameters was executed using the Python script. Using the displacement data from the FEA simulation, the Poisson's ratio was calculated using the equation below, and all the results were input into Table 1:

$$\bar{\nu} = -\frac{\bar{\varepsilon}_{xx}}{\bar{\varepsilon}_{yy}} = -\frac{u_x^{Vp_x}}{u_y^{Vp_y}}$$

Using Table 1, Plot 5 was created, and its goal is to visualize how quickly the structure can become auxetic depending on the porosity. The trend for both porosities is similar; they both decrease from the initial point. The difference is in their slope. They start from the same point, but the specimen with 2% porosity becomes an auxetic structure at an aspect ratio of 30, while 5% porosity specimen achieves it at an aspect ratio of only 15. This means that higher porosity increases the slope towards a negative Poisson's ratio.

Table 1: Calculations of the Poisson's Ratio for variety of Aspect Ratios and Porosity

Aspect Ratio	Strain X (mm) P=0.02	Strain Y (mm) P=0.02	Poisson's Ratio P=0.02	Strain X (mm) P=0.05	Strain Y (mm) P=0.05	Poisson's Ratio P=0.05
1	1.50149E-03	-5E-03	3.00298E-01	1.4961E-03	-5E-03	2.9922E-01
5	1.39055E-03	-5E-03	2.7811E-01	1.09461E-03	-5E-03	2.18922E-01
10	1.16988E-03	-5E-03	2.33976E-01	2.35122E-04	-5E-03	4.70244E-02
15	8.81568E-04	-5E-03	1.763136E-01	-8.19426E-04	-5E-03	-1.638852E-01
20	5.67946E-04	-5E-03	1.135892E-01	-1.85169E-03	-5E-03	-3.70338E-01
25	1.81746E-04	-5E-03	3.63492E-02	-2.72973E-03	-5E-03	-5.45946E-01
30	-2.32895E-04	-5E-03	-4.6579E-02	-3.41512E-03	-5E-03	-6.83024E-01
35	-6.38508E-04	-5E-03	-1.277016E-01	-3.94514E-03	-5E-03	-7.89028E-01
40	-1.03767E-03	-5E-03	-2.07534E-01	-4.32159E-03	-5E-03	-8.64318E-01

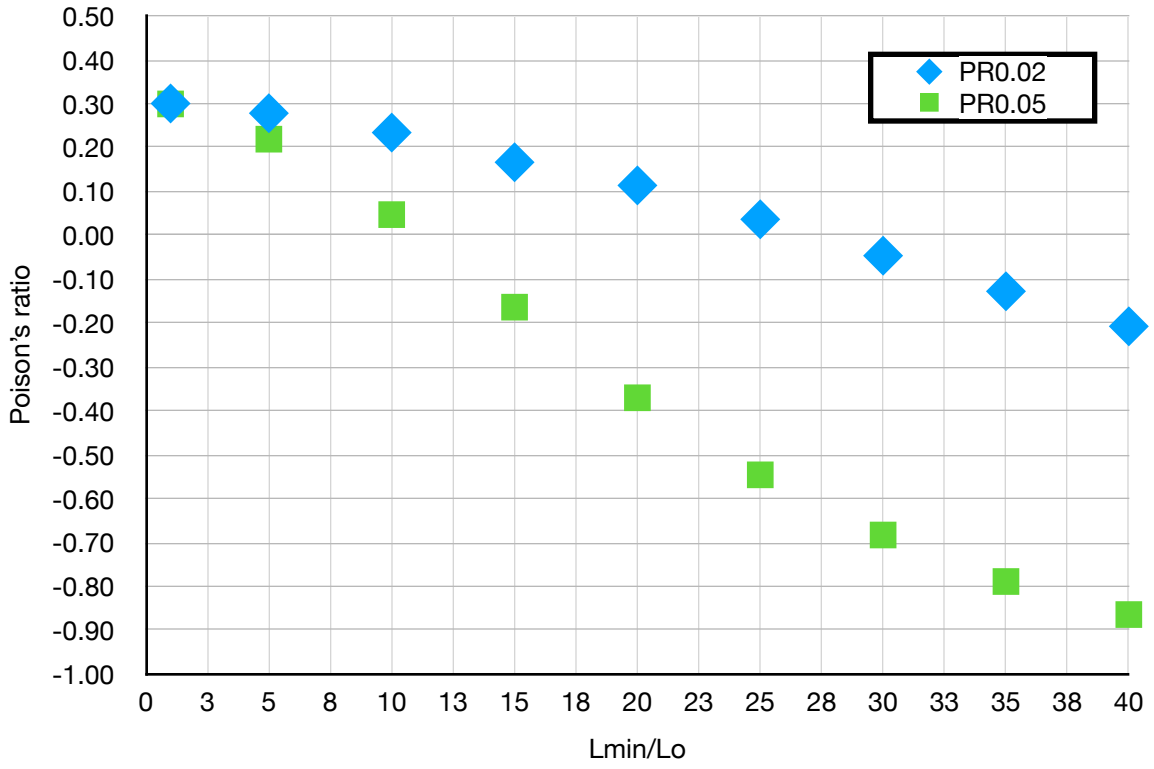

Figure 5: Plot of the effective Poisson's ratio vs. hole aspect ratio for aspect ratios of 1, 5, 10, 15, 20, 25, 30, 35, and 40. Blue rhombus represents Poisson's ratio for porosity of 2% and green square represents Poisson's ratio for porosity of 5%.

Using the data from the Python script, the ratio of Lmin/Lo was calculated for each parameter set of simulation using the equation below:

$$\frac{L_{\min}}{L_o} = \frac{1}{2}\left[1 - \left(1 + \frac{a}{b}\right)\sqrt{\frac{b\,\psi}{a\,\pi}}\right]$$

The results of the calculations were input into Tables 2 and 3. Using data from both tables, Figure 6 shows that the effective Poisson's ratio depends not specifically on aspect ratio or porosity but on the ratio of Lmin/Lo; hence, data of 5% and 2% porosities share the same trend and values.

Table 2: Calculations of Lmin/Lo for porosity of 0.05.

AR	Porosity	Area_hole (mm^2)	Lmin/Lo
1	0.05	0.5	3.74E-01
5	0.05	0.5	3.31E-01
10	0.05	0.5	2.81E-01
15	0.05	0.5	2.39E-01
20	0.05	0.5	2.04E-01
25	0.05	0.5	1.72E-01
30	0.05	0.5	1.43E-01
35	0.05	0.5	1.16E-01
40	0.05	0.5	9.11E-02

Table 3: Calculations of Lmin/Lo for porosity of 0.02.

AR	Porosity	Area_hole (mm^2)	Lmin/Lo
1	0.02	0.2	4.20E-01
5	0.02	0.2	3.93E-01
10	0.02	0.2	3.61E-01
15	0.02	0.2	3.35E-01
20	0.02	0.2	3.13E-01
25	0.02	0.2	2.93E-01
30	0.02	0.2	2.74E-01
35	0.02	0.2	2.57E-01
40	0.02	0.2	2.41E-01

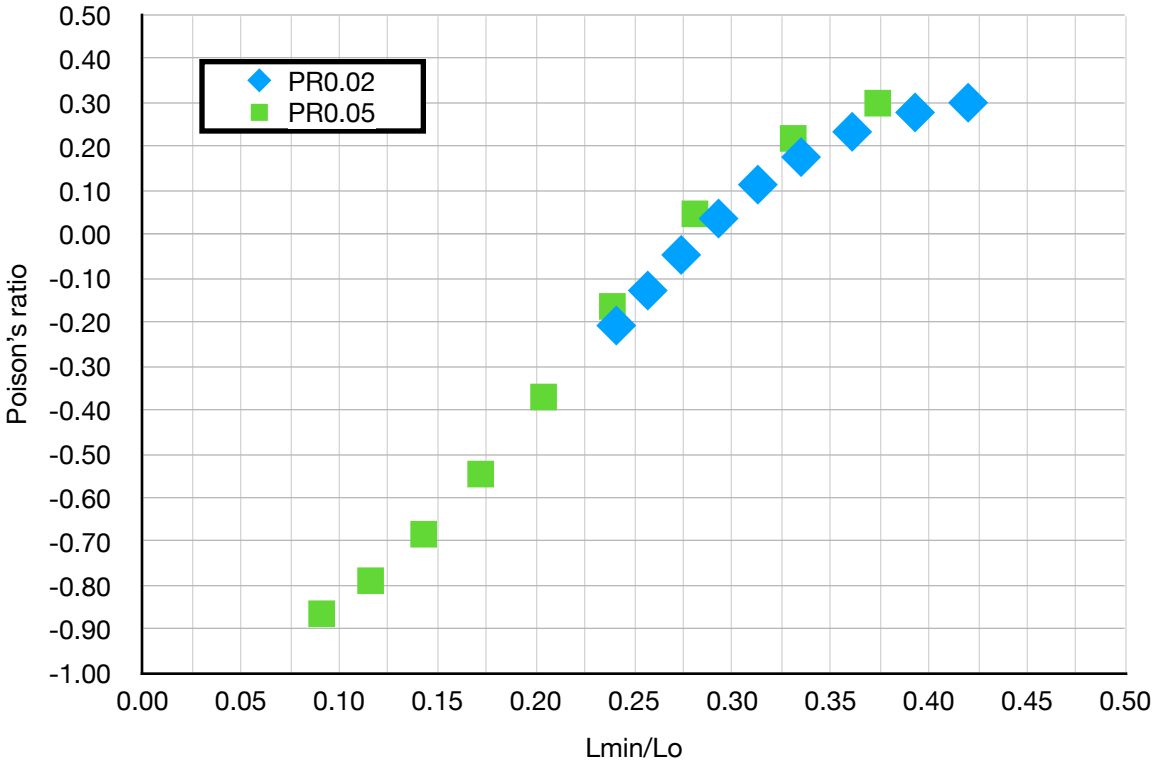

Figure 6: Plot of the effective Poisson's ratio vs. Lmin/Lo ratio for aspect ratios of 1, 5, 10, 15, 20, 25, 30, 35, and 40. Blue rhombus represents Poisson's ratio for porosity of 2% and green square represents Poisson's ratio for porosity of 5%.

3. Conclusion

This experiment confirmed that holes in specific positions and with specific properties can transform a material with a positive Poisson's ratio into an auxetic structure. Direct correlation and dependence were shown between the effective Poisson's ratio and the Lmin/Lo ratio.

Appendix

```
# ----------------------------------------------------------------------------------------------------
# German Markaryan
# 11/6/2025
#
# MECH 151/251: Lab 3
# 2D Periodic RVE with Adjustable Hole Shape
# ----------------------------------------------------------------------------------------------------

Mdb()
pathName = "Z:/dcengr/My Documents/MECH151L/Lab 3/"
os.chdir(pathName)

# Includes ---------------------------------------------------------------------------------------------
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)
# ----------------------------------------------------------------------------------------------------

# Rename model -----------------------------------------------------------------------------------------
modelName = 'model_2DVoidPlate'
mdb.models.changeKey(fromName='Model-1', toName=modelName)
# ----------------------------------------------------------------------------------------------------


# Create virtual point parts ---------------------------------------------------------------------------
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='part_VPx',
type=DEFORMABLE_BODY)
mdb.models[modelName].parts['part_VPx'].ReferencePoint(point=(0.0, 0.0, 0.0))

mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='part_VPy',
type=DEFORMABLE_BODY)
mdb.models[modelName].parts['part_VPy'].ReferencePoint(point=(0.0, 0.0, 0.0))
# ----------------------------------------------------------------------------------------------------


# Material properties ----------------------------------------------------------------------------------
YoungsMod = 200e3   # Young's modulus (in MPa)
PoissonRatio = 0.3
# ----------------------------------------------------------------------------------------------------
```

```
# Parameters ------------------------------------------------------------------------------
center_to_center = 10    # center to center distance between the holes (in mm)
porosity = 0.05                          # porosity
aspect_ratio = 40.0          # ratio between major and minor axis for the holes
thickness = 0.0                          # thickness of the plates
seed_mesh =0.25                          # seed-mesh (in mm)


width_plate = center_to_center*2.0       # width of plate  COMPLETE
height_plate = center_to_center*2.0      # height of plate COMPLETE
area_plate = height_plate*width_plate# area of the plate COMPLETE
area_hole = porosity*area_plate/4.0 # area of each void COMPLETE
major_axis_hole = sqrt(aspect_ratio*area_hole/pi) # major axis of each void COMPLETE
minor_axis_hole = area_hole/(pi*major_axis_hole) # minor axis of each void COMPLETE


subPath = pathName + "P" + str(int(porosity*100)) + "_AR" + str(aspect_ratio) + "_sS" +
str(int(seed_mesh*100)) + "/"
if not os.path.exists(subPath):
    os.makedirs(subPath)
os.chdir(subPath)
# ---------------------------------------------------------------------------------------------------


# Create parts and instances for plate ----------------------------------------------------------
partName='part_Plate'

#matrix
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].rectangle(point1=(0.0, 0.0),
    point2=(width_plate, height_plate))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Matrix',type=DEFORMABLE_BODY)
mdb.models[modelName].parts['Matrix'].BaseShell(sketch=mdb.models[modelName].sketches['__profile
__'])
del mdb.models[modelName].sketches['__profile__']

#void1
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].EllipseByCenterPerimeter(
    axisPoint1=(minor_axis_hole, 0.0), axisPoint2=(0.0, major_axis_hole), center=(0.0, 0.0))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Void1',type=DEFORMABLE_BODY)
mdb.models[modelName].parts['Void1'].BaseShell(sketch=mdb.models[modelName].sketches['__profile
__'])
del mdb.models[modelName].sketches['__profile__']

#void2
mdb.models[modelName].ConstrainedSketch(name='__profile__', sheetSize=200.0)
mdb.models[modelName].sketches['__profile__'].EllipseByCenterPerimeter(
    axisPoint1=(major_axis_hole, 0.0), axisPoint2=(0.0, minor_axis_hole), center=(0.0, 0.0))
mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR,
name='Void2',type=DEFORMABLE_BODY)
```

```
mdb.models[modelName].parts['Void2'].BaseShell(sketch=mdb.models[modelName].sketches['__profile
__'])
del mdb.models[modelName].sketches['__profile__']

#Assembly
#matrix
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Matrix-1', part=mdb.models[modelName].parts['Matrix'])
#Void 1A
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void1-1', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-1', ),
vector=(0.0, 0.0, 0.0))
#Void 1B
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void1-2', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-2', ),
vector=(2.0*center_to_center, 0.0, 0.0))
#Void 1C
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void1-3', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-3', ),
vector=(1.0*center_to_center, 1.0*center_to_center, 0.0))
#Void 1D
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void1-4', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-4', ),
vector=(0.0, 2.0*center_to_center, 0.0))
#Void 1E
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void1-5', part=mdb.models[modelName].parts['Void1'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void1-5', ),
vector=(2.0*center_to_center, 2.0*center_to_center, 0.0))
#Void 2A
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-1', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-1', ),
vector=(0.0, 1.0*center_to_center, 0.0))
#Void 2B
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-2', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-2', ),
vector=(1.0*center_to_center, 0.0, 0.0))
#Void 2C
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-3', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-3', ),
vector=(2.0*center_to_center, 1.0*center_to_center, 0.0))
#Void 2D
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
    'Void2-4', part=mdb.models[modelName].parts['Void2'])
mdb.models[modelName].rootAssembly.translate(instanceList=('Void2-4', ),
vector=(1.0*center_to_center, 2.0*center_to_center, 0.0))
```

```
#cut
mdb.models[modelName].rootAssembly.InstanceFromBooleanCut(name=partName,
      instanceToBeCut=mdb.models[modelName].rootAssembly.instances['Matrix-1'],
       cuttingInstances=(
       mdb.models[modelName].rootAssembly.instances['Void1-1'],
       mdb.models[modelName].rootAssembly.instances['Void1-2'],
       mdb.models[modelName].rootAssembly.instances['Void1-3'],
       mdb.models[modelName].rootAssembly.instances['Void1-4'],
       mdb.models[modelName].rootAssembly.instances['Void1-5'],
       mdb.models[modelName].rootAssembly.instances['Void2-1'],
       mdb.models[modelName].rootAssembly.instances['Void2-2'],
       mdb.models[modelName].rootAssembly.instances['Void2-3'],
       mdb.models[modelName].rootAssembly.instances['Void2-4'],
       ),originalInstances=SUPPRESS)


instName = 'inst_2DVoidPlate'
mdb.models[modelName].rootAssembly.features.changeKey(fromName=
   'part_Plate-1', toName=instName)

#
----------------------------------------------------------------------------------------------------------------------
-----


# Create instances for 2 virtual points ------------------------------------------------------------------
mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)

# virtual point to constrain x direction
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name='inst_VPx',
part=mdb.models[modelName].parts['part_VPx'])

# virtual point to constrain y motion
mdb.models[modelName].rootAssembly.Instance(dependent=ON, name='inst_VPy',
part=mdb.models[modelName].parts['part_VPy'])
# ----------------------------------------------------------------------------------------------------


# Create mesh ----------------------------------------------------------------------------------------
# COMPLETE THIS SECTION USING QUADRATIC TRIANGULAR ELEMENTS
# WHEN DEFINING THE SEED, MAKE SURE CURVATURE CONTROL IS OFF

mdb.models['model_2DVoidPlate'].parts['part_Plate'].setMeshControls(elemShape=
   TRI, regions=
mdb.models['model_2DVoidPlate'].parts['part_Plate'].faces[:])
mdb.models['model_2DVoidPlate'].parts['part_Plate'].seedPart(size=seed_mesh)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].generateMesh()
# ----------------------------------------------------------------------------------------------------


# Create material ------------------------------------------------------------------------------------
# COMPLETE THIS SECTION USING A LINEAR ELASTIC MATERIAL MODEL
mdb.models[modelName].Material(name='kryptonite')
```

```
mdb.models[modelName].materials['kryptonite'].Elastic(table=((YoungsMod, PoissonRatio),
   ))
# ----------------------------------------------------------------------------------------------------


# Create section, assign to part ---------------------------------------------------------------------
# COMPLETE THIS SECTION
mdb.models['model_2DVoidPlate'].HomogeneousSolidSection(material='kryptonite',
   name='plateSection', thickness=thickness)
mdb.models['model_2DVoidPlate'].parts['part_Plate'].SectionAssignment(offset=
   0.0, offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
   faces=mdb.models['model_2DVoidPlate'].parts['part_Plate'].faces[:]), sectionName='plateSection'
   , thicknessAssignment=FROM_SECTION)

# ----------------------------------------------------------------------------------------------------

# Define sets containing all nodes and elements
-------------------------------------------------------------
mdb.models[modelName].parts[partName].Set(name='set_AllElements',
elements=mdb.models[modelName].parts[partName].elements)

mdb.models[modelName].parts[partName].Set(name='set_AllNodes',
nodes=mdb.models[modelName].parts[partName].nodes)
# ----------------------------------------------------------------------------------------------------


# Create arrays and Sets containing node numbers for all faces of plate
-----------------------------------------

# initialize arrays for faces (denoted by outward normal)
nodes_rightEdge = []
nodes_leftEdge = []
nodes_topEdge = []
nodes_bottomEdge = []
node_RBM = []


# define arbitrary tolerance for boolean comparison
eps = seed_mesh/10.0

offset = minor_axis_hole

# loop over all nodes and sort out nodes on the edges
for N in mdb.models[modelName].parts[partName].nodes:

   nodeCoord = N.coordinates

   if (fabs(nodeCoord[0]-center_to_center) < 10.0*eps) and (fabs(nodeCoord[1]-minor_axis_hole) <
10.0*eps):
         node_RBM.append(N.label)

   elif (fabs(nodeCoord[0]) < eps):
         nodes_leftEdge.append(N.label)
```

```
    elif (fabs(nodeCoord[0]-width_plate) < eps):
            nodes_rightEdge.append(N.label)

    elif (fabs(nodeCoord[1]) < eps):
            nodes_bottomEdge.append(N.label)

    elif (fabs(nodeCoord[1]-height_plate) < eps):
            nodes_topEdge.append(N.label)

mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesRightEdge',
nodeLabels=nodes_rightEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesLeftEdge',
nodeLabels=nodes_leftEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesTopEdge',
nodeLabels=nodes_topEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesBottomEdge',
nodeLabels=nodes_bottomEdge)
mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodeRBM',
nodeLabels=(node_RBM[0],))

# create sets for virtual points
mdb.models[modelName].parts['part_VPx'].Set(name='set_VPx',
referencePoints=(mdb.models[modelName].parts['part_VPx'].referencePoints[1], ))
mdb.models[modelName].parts['part_VPy'].Set(name='set_VPy',
referencePoints=(mdb.models[modelName].parts['part_VPy'].referencePoints[1], ))

# ----------------------------------------------------------------------------------------------------------


# Create sets of periodic node pairs ------------------------------------------------------------------------

# Look at left and right sides
for i in range (0, len(nodes_leftEdge)):
        leftCoords =
mdb.models[modelName].parts[partName].sets['set_NodesLeftEdge'].nodes[i].coordinates
        mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesLPair_' +
str(i), nodeLabels=(nodes_leftEdge[i],))
        for j in range (0, len(nodes_rightEdge)):
                rightCoords =
mdb.models[modelName].parts[partName].sets['set_NodesRightEdge'].nodes[j].coordinates
                if (fabs(leftCoords[1] - rightCoords[1]) < eps):

mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesRPair_' + str(i),
nodeLabels=(nodes_rightEdge[j],))


# Look at top and bottom sides
for i in range (0, len(nodes_topEdge)):
        topCoords =
mdb.models[modelName].parts[partName].sets['set_NodesTopEdge'].nodes[i].coordinates
        mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesTPair_' +
str(i), nodeLabels=(nodes_topEdge[i],))
        for j in range (0, len(nodes_bottomEdge)):
```

```
                    bottomCoords =
mdb.models[modelName].parts[partName].sets['set_NodesBottomEdge'].nodes[j].coordinates
                    if (fabs(topCoords[0] - bottomCoords[0]) < eps):

mdb.models[modelName].parts[partName].SetFromNodeLabels(name='set_NodesBPair_' + str(i),
nodeLabels=(nodes_bottomEdge[j],))

# ------------------------------------------------------------------------------------------------------------------


# Create analysis step ---------------------------------------------------------------------------
mdb.models[modelName].StaticStep(name='step_Compression', previous='Initial')
# ------------------------------------------------------------------------------------------------------------------


# Set up BCs
------------------------------------------------------------------------------------------------------------------

# fix point to prevent rigid body motion
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
    'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_preventRBM', region=
    mdb.models[modelName].rootAssembly.instances[instName].sets['set_NodeRBM']
    , u1=0.0, u2=0.0, ur3=UNSET)


# externally applied strain through the virtual points (x-dir)
#-------------------------------------------------------------
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
    'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_VPx', region=
    mdb.models[modelName].rootAssembly.instances['inst_VPx'].sets['set_VPx']
    , u1=UNSET, u2=0.0, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)


# externally applied strain through the virtual points (y-dir)
#-------------------------------------------------------------
mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
    'step_Compression', distributionType=UNIFORM, fieldName='', fixed=OFF,
    localCsys=None, name='bc_VPy', region=
    mdb.models[modelName].rootAssembly.instances['inst_VPy'].sets['set_VPy']
    , u1=0.0, u2=-0.005, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
#
------------------------------------------------------------------------------------------------------------------
---


# Set up periodic constraint equations
------------------------------------------------------------------------------

# right and left edges
for i in range(0,len(nodes_leftEdge)):

    # preparation of Coefficients
```

```
    leftCoord=mdb.models[modelName].parts[partName].sets['set_NodesLPair_' +
str(i)].nodes[0].coordinates
    rightCoord=mdb.models[modelName].parts[partName].sets['set_NodesRPair_' +
str(i)].nodes[0].coordinates

    coeff1 = -(rightCoord[0]-leftCoord[0])

    # x-coordinate (Ux_Vpx, H11)
    mdb.models[modelName].Equation(name='constraint_xLR_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesRPair_' + str(i), 1),
        (-1.0, 'inst_2DVoidPlate.set_NodesLPair_' + str(i), 1),
        (coeff1, 'inst_VPx.set_VPx', 1)))

    # y-coordinate (Uy_Vpx, H21)
    mdb.models[modelName].Equation(name='constraint_yLR_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesRPair_' + str(i), 2),
        (-1.0, 'inst_2DVoidPlate.set_NodesLPair_' + str(i), 2),
        (coeff1, 'inst_VPx.set_VPx', 2)))

# top and bottom edges
for i in range(0,len(nodes_bottomEdge)):

    # preparation of Coefficients
    bottomCoord=mdb.models[modelName].parts[partName].sets['set_NodesBPair_' +
str(i)].nodes[0].coordinates
    topCoord=mdb.models[modelName].parts[partName].sets['set_NodesTPair_' +
str(i)].nodes[0].coordinates

    coeff2 = -(topCoord[1]-bottomCoord[1])

    # x-coordinate (Ux_Vpy, H12)
    mdb.models[modelName].Equation(name='constraint_xTB_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesTPair_' + str(i), 1),
        (-1.0, 'inst_2DVoidPlate.set_NodesBPair_' + str(i), 1),
        (coeff2, 'inst_VPy.set_VPy', 1)))

    # y-coordinate (Uy_Vpy, H22)
    mdb.models[modelName].Equation(name='constraint_yTB_' + str(i), terms=(
        ( 1.0, 'inst_2DVoidPlate.set_NodesTPair_' + str(i), 2),
        (-1.0, 'inst_2DVoidPlate.set_NodesBPair_' + str(i), 2),
        (coeff2, 'inst_VPy.set_VPy', 2)))

#
------------------------------------------------------------------------------------------------------------------------
---


# Field Output
-------------------------------------------------------------------------------------------------------------
# Force/Displacement at the Virtual Points
mdb.models[modelName].FieldOutputRequest(createStepName=
    'step_Compression', name='output_VPx', rebar=EXCLUDE, region=
    mdb.models[modelName].rootAssembly.instances['inst_VPx'].sets['set_VPx']
    , sectionPoints=DEFAULT, variables=('RF', 'U'))
```

```
mdb.models[modelName].FieldOutputRequest(createStepName=
    'step_Compression', name='output_VPy', rebar=EXCLUDE, region=
    mdb.models[modelName].rootAssembly.instances['inst_VPy'].sets['set_VPy']
    , sectionPoints=DEFAULT, variables=('RF', 'U'))
#
----------------------------------------------------------------------------------------------------------------------
---


# Create and submit the job for processing
--------------------------------------------------------------------------------
# COMPLETE THIS SECTION
mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=90, memoryUnits=PERCENTAGE, model=modelName, modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='auxeticstructure', nodalOutputPrecision=
    SINGLE, numCpus=1, numGPUs=0, queue=None, resultsFormat=ODB, scratch='',
    type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
mdb.jobs['auxeticstructure'].submit(consistencyChecking=OFF)
mdb.jobs['auxeticstructure'].waitForCompletion()
mdb.saveAs(pathName=subPath + 'auxeticstructure.cae')
#
----------------------------------------------------------------------------------------------------------------------
---


# # Postprocessing THIS CODE IS NOT WORKING AND DR. Taylor solution didnt work
----------------------------------------------------------------------------------------------------------
# # CREATE A .RPT FILE CONTAINING NODAL VALUES OF ALL RF AND U COMPONENTS
# #session.viewports['Viewport: 1'].setValues(displayedObject=None)
# o1 = session.openOdb(name=subPath + 'auxeticstructure.odb')
# #session.viewports['Viewport: 1'].setValues(displayedObject=o1)
# #session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
# #    CONTOURS_ON_DEF, ))
# #odb = session.odbs[subPath + 'auxeticstructure.odb']
# odb = session.odbs[subPath+'auxeticstructure.odb']
# session.writeFieldReport(fileName=pathName + "P" + str(int(porosity*100)) + "_AR" + str(aspect_ratio)
+ "_sS" + str(int(seed_mesh*100)) + "/"+'disReport.rpt', append=ON,
    # sortItem='Node Label', odb=odb, step=0, frame=1, outputPosition=NODAL,
    # variable=(('RF', NODAL), ('U', NODAL), ), stepFrame=SPECIFY)
# #
----------------------------------------------------------------------------------------------------------------------
---
```