**Name: Nguyen Xuan Binh**

**Student ID: 887799**

**Fracture Mechanics Assignment 5**

A picture containing text, diagram, screenshot, font

Description automatically generated

A picture containing text, font, screenshot, line

Description automatically generated

A picture containing text, screenshot, font, algebra

Description automatically generated

In this exercise, I use Python for data processing and calculations

First, we can plot the stress strain curve recorded in the Tension\_test.xlsx file

*# Read from Tensile\_test.xlsx*

df = pd.read\_excel('Tensile\_test.xlsx')

forceTensile = df['Force (N)']

displacementTensile = df['Extension (mm)']

*# Plot stress strain curve*

plt.title('Tensile test curve')

plt.plot(displacementTensile, forceTensile)

plt.xlabel('Displacement (mm)')

plt.ylabel('Force (N)')

A picture containing text, diagram, line, plot

Description automatically generated

Force displacement curve cannot derive the Young’s modulus, so the geometry of the dogbone specimen needs to be considered. Stress is obtained by dividing the force by the cross-sectional area, and strain is obtained by dividing displacements by original length

A picture containing sketch, diagram, drawing, line

Description automatically generated

*# Cross section properties*

width = 6 *# mm*

thickness = 1.68 *# mm*

*# Original length*

L0 = 33 *# mm*

area = width \* thickness *# mm^2*

stress = forceTensile / area *# N/mm^2 = MPa*

strain = displacementTensile / L0 *# mm/mm = dimensionless*

*# Plot stress strain curve*

plt.title('Stress strain curve')

plt.plot(strain, stress)

plt.xlabel('Strain, -')

plt.ylabel('Stress, MPa')

A graph of stress strain curve

Description automatically generated with medium confidence

I proceed to fit a linear regression to the first 300 datapoints to measure the slope of the regression, which is the Young’s modulus. The result is

 (ANSWER)

A graph of stress strain curve

Description automatically generated with medium confidence

However, this result varies greatly depending on the number of initial points. If 400 datapoints are taken, Young’s modulus would be 2180 MPa.

A picture containing text, screenshot, font, algebra

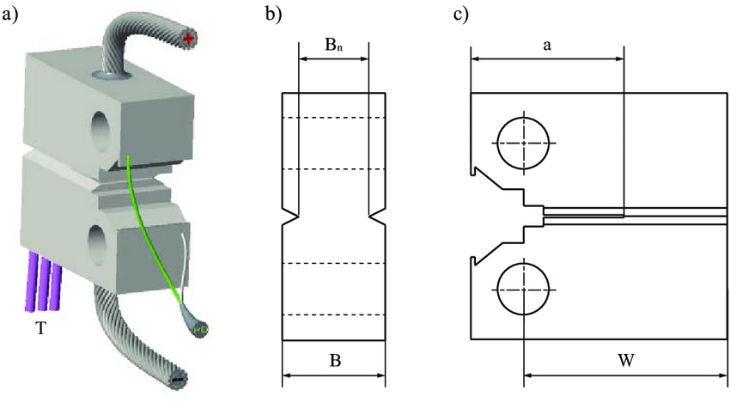
Description automatically generated

The formula to compute is in page 18-19 in the ASTM E399 file. The formulas are based on the compact specimen geometry in Appendix A4

A picture containing sketch, diagram, technical drawing, drawing

Description automatically generated

A picture containing diagram, plan, technical drawing, line

Description automatically generated

A picture containing text, screenshot, font, algebra

Description automatically generated

A picture containing text, font, screenshot, white

Description automatically generated

A picture containing text, font, line, diagram

Description automatically generated

A picture containing text, font, screenshot, number

Description automatically generated

In this case, B = Bn according to the geometry given in the exercise

First, I plot the toughness test curve

df = pd.read\_excel('Toughness\_test.xlsx')

forceToughness = df['Force (N)']

displacementToughness = df['Extension (mm)']

plt.title('Toughness test curve')

plt.plot(displacementToughness, forceToughness)

plt.xlabel('Displacement (mm)')

plt.ylabel('Force (N)')

A picture containing text, line, diagram, plot

Description automatically generated

In the graph, the force is applied until the fracture occurs and the force is dropped to 0.

*# Maximum force at fracture moment*

P = max(forceToughness)

print(f"Critical force: {round(P)} N")

Critical force at fracture moment:



Then, we can apply the formula above:

*# Compact tension specimen*

a = 0.030 *# m*

W = 0.06 *# m*

*# We have B = Bn*

B = 0.015 *# m*

Bn = 0.015 *# m*

def f(*a*, *W*):

    aW = *a* / *W*

    nominator = (2 + aW) \* (0.886 + 4.64 \* aW - 13.32 \* aW\*\*2 + 14.72 \* aW\*\*3 - 5.6 \* aW\*\*4)

    denominator = (1 - aW) \*\* (3/2)

*return* nominator/denominator

*# Return Kic of unit Pa√(m)*

def Kic\_equation(*P*, *B*, *Bn*, *a*, *W*):

*return* *P*/(np.sqrt(*B* \* *Bn*) \* np.sqrt(*W*)) \* f(*a*, *W*)

*# The fracture toughness Kic of the material is:*

Kic = Kic\_equation(P, B, Bn, a, W)

print(f"Fracture toughness: {round(Kic) \* 1e-6} MPa√(m)")

The fracture toughness is:

 (ANSWER)

A picture containing text, font, screenshot, line

Description automatically generated

These are the model and output files of the FE model of the part shown in Figure 1

A screenshot of a computer

Description automatically generated with medium confidence

Originally, I thought the commercial software refers to the license version, but it turns out to be Abaqus’s student version. Therefore, the output files above can only be loaded in Student’s version and not the Full License version

A picture containing text, screenshot, font, document

Description automatically generated

**Geometry modelling**

Creating geometry without the crack using shell planar 2D sketch. Sketch drawing size of 0.2 (200 mm). We only need to model half of the specimen as it is Y-symmetrical

A picture containing screenshot, plot

Description automatically generated

Add the crack at the lower end. The angle of the crack is 30 degrees

A picture containing screenshot, plot

Description automatically generated

A white object with a circle

Description automatically generated with low confidence

Partitioning Face sketch. This is helpful for creating different seed sizes for the two separate regions, as the region near the boundary conditions should have finer mesh.

A picture containing screenshot, multimedia software, text, graphics software

Description automatically generated

**Polymer material definition and section assignment**

Elasticity: Young’s Modulus of 2.289e9 Pa and Poisson’s ratio of 0.3

Plasticity: no plasticity for polymer material

After that, we assign the material to a section. Then we assign the model to the section

A screenshot of a computer

Description automatically generated A green object with a circle

Description automatically generated with low confidence

**Create assembly, add time step, loading definition and boundary conditions**

Time step: choosing default settings

Loading: apply the load on as a concentrated mechanical force in upward direction (Y-axis) on the top of the circular nod

Boundary condition: set the line at the lower end of the model as the boundary condition (Ysymm option)

A picture containing screenshot, graphics software, text, multimedia software

Description automatically generated

**Define the crack and history output request.**

The crack is defined by Interaction > Special tab

After definition of the crack, it will appear in Assembly > Engineering Features in the model tree. Then we proceed to add the history output request like below, which defines the number of contours for stress intensity factors. We choose number of contours = 20

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

**Meshing the specimen**

The top part above the line partition: seed edges of 0.003 (mm) for coarse mesh

The below part under the line partition: seed egdes of 0.001 (mm) for fine mesh

A picture containing screenshot, graphics software, multimedia software, software

Description automatically generated

**Submitting the simulation job**

A screenshot of a computer

Description automatically generated

**Result visualization, such as von Mises stress and strain**

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, screenshot, font, document

Description automatically generated

Table 1: Values of stress intensity factor  for selected number of contours used to compute the contour integral. All results are for an applied load P = 1N

|  |  |
| --- | --- |
| Number of contours |  |
| C01 | 3926.92 |
| C02 | 5092.67 |
| C03 | 5188.54 |
| C04 | 5229.28 |
| C05 | 5287.29 |
| C06 | 5322.84 |
| C07 | 5335.68 |
| C08 | 5338.94 |
| C09 | 5342.16 |
| C10 | 5345.00 |
| C11 | 5246.52 |
| C12 | 5123.87 |
| C13 | 5085.03 |
| C14 | 5127.16 |
| C15 | 5188.79 |
| C16 | 5124.87 |
| C17 | 5253.49 |
| C18 | 5332.69 |
| C19 | 5332.69 |
| C20 | 4104.00 |

We can see the value of stress intensity factor fluctuates around 5250-5350 . I will choose the sixth contour as the stress intensity factor for testing

To find , we can try different magnitude of the loading such that the stress intensity factor at C06 reaches 1114850 . Since the LEFM is assumed, the stress intensity factor is linearly proportional to the loading magnitude. Therefore, we can run a few simulations to check the slope of the regression

|  |  |
| --- | --- |
| P (N) | at C06 |
| 1N | 5322.84 |
| 250N | 1.33071E+06 |
| 500N | 2.66142E+06 |
| 750N | 3.99213E+06 |
| 1000N | 5.32284E+06 |

*# Linear regression between stress intensity factor KI and applied force P*

P = np.array([1, 250, 500, 750, 1000])

KI = np.array([5322.84,1.33071E+06, 2.66142E+06,3.99213E+06,5.32284E+06])

*# fit linear regression*

P = P.reshape(-1, 1)

KI = KI.reshape(-1, 1)

reg = LinearRegression()

reg.fit(KI, P)

*# Slope*

print(f"Slope of linear regression between P and KI: {reg.coef\_[0][0]}")



*# Predict Pmax at stress intensity factor KI = KIc = 1114850 Pa√(m)*

KIc = 1114850

Pmax = reg.predict([[KIc]]).item()

print(f"Maximum force at fracture moment: Pmax = {round(Pmax, 2)} N")

 (ANSWER)

We can now plug in Pmax in Abaqus to verify if  is near 

A screenshot of a computer

Description automatically generated with low confidence

It is equal to the critical stress intensity factor, so we can conclude that Pmax = 209.45N which will cause the specimen to fracture

A picture containing text, screenshot, font, document

Description automatically generated

Formula in the appendix

A picture containing text, screenshot, font, number

Description automatically generated

*# Arc-shape tension specimen*

a = 0.01 *# m*

B = 0.01 *# m*

r1 = 0.03 *# m*

r2 = 0.06 *# m*

W = 0.03 *# m*

X = 0.015 *# m*

KIc = 1114850 *# Pa√(m)*

def f(*a*, *W*):

    aW = *a* / *W*

    firstTerm = np.sqrt(aW)/(1-aW)\*\*(3/2)

    secondTerm = 3.74 – 6.30 \* aW + 6.32 \* aW\*\*2 – 2.43 \* aW \*\*3

*return* firstTerm \* secondTerm

*# Return KI of unit Pa√(m)*

def KQ\_equation(*P*, *B*, *X*, *r1*, *r2*, *a*, *W*):

    firstTerm = *P* /(*B* \* np.sqrt(*W*))

    secondTerm = 3 \* (*X*/*W*) + 1.9 + 1.1 \* (*a*/*W*)

    thirdTerm = 1 + 0.25 \* (1 – *a*/*W*) \*\* 2 \* (1 – *r1*/*r2*)

    fourthTerm = f(*a*, *W*)

*return* firstTerm \* secondTerm \* thirdTerm \* fourthTerm

*# Return KI of unit Pa√(m), no other params exept P*

def KQ\_equation\_P(*P*):

    firstTerm = *P* /(B \* np.sqrt(W))

    secondTerm = 3 \* (X/W) + 1.9 + 1.1 \* (a/W)

    thirdTerm = 1 + 0.25 \* (1 – a/W) \*\* 2 \* (1 – r1/r2)

    fourthTerm = f(a, W)

*return* firstTerm \* secondTerm \* thirdTerm \* fourthTerm – Kic

*# KQ at Pmax = 209.45*

KQ = KQ\_equation(Pmax, B, X, r1, r2, a, W)

print(f"Stress intensity factor KQ at Pmax: {round(KQ)} Pa√(m)")



We can see the predicted Pmax produces very close to the critical stress intensity factor . We can also reverse the function to find Pmax given 

sol = optimize.root\_scalar(KQ\_equation\_P, *bracket* = [0,1000],  *method*='brentq')

print(F"Pmax given KIc is {round(sol.root,2)} N")



The analytical solution is close to the numerical one above, which is Pmax = 209.45. Therefore, we can conclude that the FE model in Abaqus is quite reliable. However, the gap of 5N in Pmax prediction is also noticeable. Several factors can explain the variation between the analytical and numerical results:

1. Simplified assumptions: analytical estimations rely on theoretical models that may not fully capture all microstructure complexities present in the tested specimen.
2. Material Behavior: the FE model in Abaqus only features elasticity properties without any other specifications. In reality, there could be also thermality or conductivity
3. Mesh Discretization: The FE model's accuracy is influenced by the mesh seeds. If the mesh is not fine enough, it can lead to deviations from the true numerical result

Limitations of the FE model in Abaqus:

1. Mesh sensitivity: The accuracy of the results can be sensitive to the mesh density. Especially the student version’s version is limited to only 1000 elements on the mesh. Additionally, only squares mesh can support stress intensity factor, so if we choose different mesh geometry, the FE model would not work
2. Difficult to work with fix mode scenarios: this FE model has  and defined separately, but if we want mix mode, then it is harder to quantify the effect

How to improve the FE model:

1. Refine Mesh: Create denser mesh under the line partition near the boundary conditions to give better predictions for mode I stress intensity factors
2. Define more material properties, such as damage for elastomers, thermal and electrical conductivity. This will better accurately capture the behavior of the polymer