Homework 6 Solutions

November 1, 2019

1 Homework 6 Solutions

In this homework we will compare the predictions of crack propagation direction for various analytic methods to finite elements for mixed-mode fracture conditions.

```
In [42]: # first we set up the Python libraries we will need
         #only for live notebooks
         %matplotlib inline

         import numpy as np #numeric array library
         from scipy.optimize import fsolve #solver
         from matplotlib import pyplot as plt #plotting
```

1.1 Maximum Stress Criterion

For the maximum stress criterion we find the maximum hoop stress using

$$K_I \sin\theta + K_{II}(3\cos\theta - 1) = 0$$

```
In [43]: #configuration a, note that K_I = K_II/3
         def f_a(x):
             return np.sin(x)/3.+3*np.cos(x)-1

         #configuration b, note that K_I = 7/4*K_II
         def f_b(x):
             return 7.*np.sin(x)/4.+3*np.cos(x)-1

         t_a = fsolve(f_a,-1)*180./np.pi #solve for f_a=0, convert from radians to degrees
         print t_a[0]
         t_b = fsolve(f_b,-1)*180./np.pi #solve for f_b=0, convert from radians to degrees
         print t_b[0]
```

```
-64.3124382932
-43.009715535
```

1.2 S-Criterion

For the Strain Energy Density Criterion, we find when the strain energy density has a minimum value around the crack tip using

$$S = \frac{1}{16\mu\pi}(a_{11}K_I^2 + a_{12}K_I K_{II} + a_{22}K_{II}^2)$$

where

$$a_{11} = (\kappa - \cos\theta)(1 + \cos\theta)$$

$$a_{12} = sin\theta(2\cos\theta - \kappa + 1)$$

$$a_{22} = [(\kappa + 1)(1 - \cos\theta) + (1 + \cos\theta)(3\cos\theta - 1)$$

And the minimum is found as

$$\frac{\partial S}{\partial \theta} = 0$$

when

$$\frac{\partial^2 S}{\partial \theta^2} > 0$$

```
In [44]: #material properties
         mu = 45.5e9
         nu = 0.3
         #kappa = 3-4*nu #plane strain
         kappa = (3-nu)/(1+nu) #plane stress

In [45]: #for part a, K_I = 1/3 * K_II
         K_Ia = 1./3.
         #for part b, K_I = 7/4 * K_II
         K_Ib = 7./4.

         #initial theta for numerical derivative
         theta_a = np.linspace(t_a-20,t_a+20,200)*np.pi/180. #part a
         theta_b = np.linspace(t_b-20,t_b+20,200)*np.pi/180. #part b

         #a constants
         a11a = ((kappa-np.cos(theta_a))*(1+np.cos(theta_a)))
         a12a = np.sin(theta_a)*(2*np.cos(theta_a)-kappa+1)
         a22a = ((kappa+1)*(1-np.cos(theta_a))+(1+np.cos(theta_a))*(3*np.cos(theta_a)-1))

         #part b
         a11b = ((kappa-np.cos(theta_b))*(1+np.cos(theta_b)))
         a12b = np.sin(theta_b)*(2*np.cos(theta_b)-kappa+1)
         a22b = ((kappa+1)*(1-np.cos(theta_b))+(1+np.cos(theta_b))*(3*np.cos(theta_b)-1))

         #S
         Sa = 1./(16*mu*np.pi)*(a11a*K_Ia**2 + a12a*K_Ia + a22a)
         Sb = 1./(16*mu*np.pi)*(a11b*K_Ib**2 + a12b*K_Ib + a22b)
```

```
    #numerical derivative
    dta = theta_a[1]-theta_a[0]
    dtb = theta_b[1]-theta_b[0]
    Sap = np.gradient(Sa,dta)
    Sbp = np.gradient(Sb,dtb)

    #check second derivative
    Sapp = np.gradient(Sap,dta)
    Sbpp = np.gradient(Sbp,dtb)

    #find minimum, part a
    ia = np.abs(Sap).argmin() #find index at zero
    ta_s = theta_a[ia] #find theta
    print ta_s*180./np.pi #degrees
    print Sapp[ia] #check 2nd derivative, should be positive

    #find minimum, part b
    ib = np.abs(Sbp).argmin() #find index at zero
    tb_s = theta_b[ib] #find theta
    print tb_s*180./np.pi #degrees
    print Sbpp[ib] #check 2nd derivative, should be positive
```

```
-73.4581669364
2.40359342383e-12
-47.5323286004
1.78028999291e-12
```

1.3 ME-Criterion

The maximum energy release rate criterion assumes there is a small kink crack and we find the direction where the strain energy release rate of that kink crack is maximum to find the direction the crack will propagate.

In general we have

$$G_{kink} = \frac{\kappa+1}{8\mu}(k_I^2 + k_{II}^2)$$

where

$$k_I = C_{11}(\theta)K_I + C_{12}(\theta)K_{II}$$

$$k_{II} = C_{21}(\theta)K_I + C_{22}(\theta)K_{II}$$

There are several formulas in the text for C based on various assumptions (5.22, 5.23, 5.24), however it can be shown that 5.23 and 5.24 reduce to the maximum tensile stress criterion, hence we will use 5.22 here. (Note that the equations in 5.22 use a negative definition of the propagation angle, so if the formulas are used as-is, the negative angle needs to be taken).

The maximum can be found with

$$\frac{\partial G}{\partial \theta} = 0$$

Where

$$\frac{\partial^2 G}{\partial \theta^2} < 0$$

```
In [46]: #define angle arrays for numerical derivatives
         theta_a = np.linspace(-t_a-20,-t_a+20,200)*np.pi/180. #part a
         theta_b = np.linspace(-t_b-20,-t_b+20,200)*np.pi/180. #part b

         #define constants
         C_a = np.array([[np.cos(theta_a), 3./2.*np.sin(theta_a)],[-0.5*np.sin(theta_a), np.cos
         C_b = np.array([[np.cos(theta_b), 3./2.*np.sin(theta_b)],[-0.5*np.sin(theta_b), np.cos
         #k_i, k_ii, and G for part a, recall that K_I = K_II/3
         kia = C_a[0,0]/3.+C_a[0,1]
         kiia = C_a[1,0]/3.+C_a[1,1]
         G_a = (kappa+1)/(8*mu)*(kia**2+kiia**2)
         #k_i, k_ii, and G for part b, recall that K_I = 7K_II/4
         kib = C_b[0,0]*7./4.+C_b[0,1]
         kiib = C_b[1,0]*7./4.+C_b[1,1]
         G_b = (kappa+1)/(8*mu)*(kib**2+kiib**2)

         G_ap = np.gradient(G_a,dta) #first derivative
         G_app = np.gradient(G_ap,dta) #second derivative
         iame = np.abs(G_ap).argmin() #find zero
         tame = theta_a[iame]
         print -tame*180./np.pi #note: angle has different definition, take negative
         print G_app[iame] #check second derivative, should be negative

         G_bp = np.gradient(G_b,dtb) #first derivative
         G_bpp = np.gradient(G_bp,dtb) #second derivative
         ibme = np.abs(G_bp).argmin() #find zero
         tbme = theta_b[ibme]
         print -tbme*180./np.pi #note: angle has different definition, take negative
         print G_bpp[ibme] #check second derivative, should be negative
```

```
-69.0360563836
-6.07570319449e-11
-46.3262984496
-8.82277590755e-11
```

1.4 Finite Elements

I tested both COMSOL (maximum hoop stress) and ABAQUS (measured angle of XFEM crack propagation).

```
In [47]: #visual inspection from ImageJ
         #abaqus part a
         abq_a = 131.5-180
         #abaqus part b
```

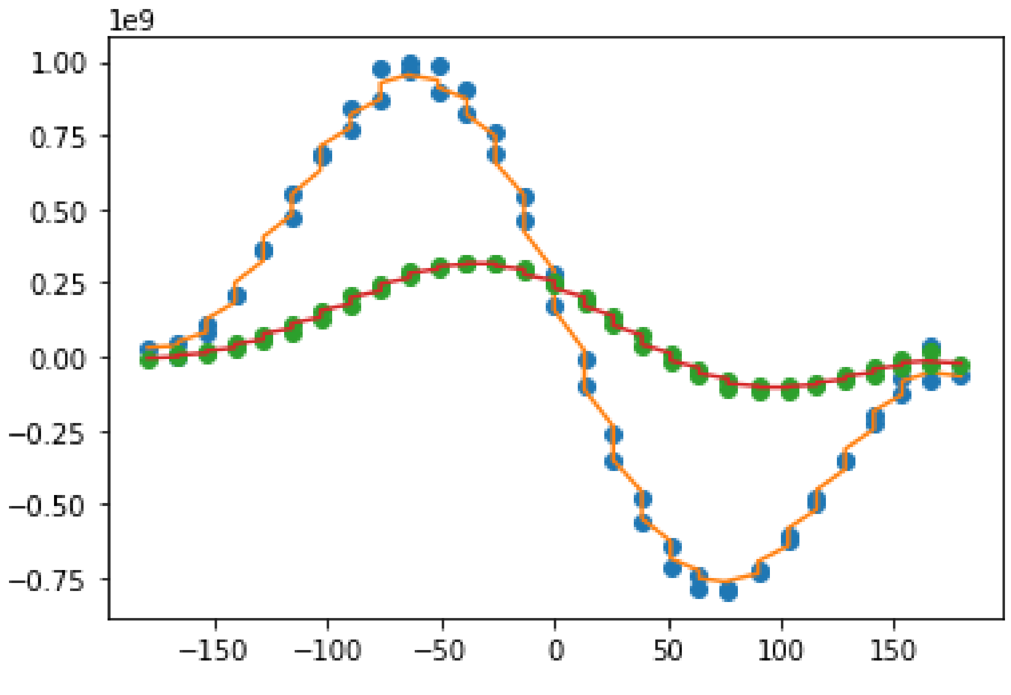
```
abq_b = 144.4-180

#COMSOL hoopstress data can be loaded in
data = np.loadtxt('hw6-comsol-hoop.csv',delimiter=',')

#filter noise from signal
from scipy import signal #useful filtering functions
b,a = signal.butter(1,.3)

parta = signal.filtfilt(b,a,data[:,1],padlen=5)
partb = signal.filtfilt(b,a,data[:,3],padlen=5)
#make sure filter makes sense
plt.plot(data[:,0],data[:,1],'o')
plt.plot(data[:,0],parta)
plt.plot(data[:,2],data[:,3],'o')
plt.plot(data[:,2],partb)

#find maxima
com_a = data[signal.argrelmax(parta),0][0][0]
com_b = data[signal.argrelmax(partb),0][0][0]
```



1.5 Comparison

In the table below we compare all the results

Method	Part a	Part b
Max Stress	-64.31	-43.01
S-Criterion	-73.46	-47.53
ME-Criterion	-69.04	-46.33
Abaqus	-48.50	-35.60
COMSOL	-63.99	-38.67

While the ABAQUS values do not compare well with the predictions, that may be caused by some inexact boundary conditions. The traditional method (an oblique crack at some angle β under remote tension), might give better results.