HW08-sympy

November 3, 2015

0.1 Assignment 8 - Brandon Lampe

0.2 Problem 7.5

Show that for plane stress, $\nu = 1/3$ and $c = \frac{\sqrt{3}}{2}ELt$ using Equation 7.13

Import symbolic packages

```
In [33]: from sympy import *
         from scipy import linalg as LA
         import numpy as np
         from sympy import solve
         from sympy import Matrix
         # from sympy import init_printing
         # init_printing()

         np.set_printoptions(precision=3, suppress=False) # precision for numpy operations
```

Define symbolic variables

```
In [2]: nu, Y, L, t, c, c_1, c_2, V = symbols('nu Y L t c c_1 c_2 V')

In [3]: N = np.array(([1,0,0],[1/4.,3/4.,3**0.5/4],[1/4.,3/4.,-3**0.5/4.]))
        print N

[[ 1.          0.          0.        ]
 [ 0.25        0.75        0.4330127]
 [ 0.25        0.75       -0.4330127]]

In [4]: D_pstrs = Y / (1-nu**2)*np.array(([1,nu,0],[nu,1,0],[0,0,(1-nu)/2]))
        print D_pstrs

[[Y/(-nu**2 + 1) Y*nu/(-nu**2 + 1) 0]
 [Y*nu/(-nu**2 + 1) Y/(-nu**2 + 1) 0]
 [0 0 Y*(-nu/2 + 1/2)/(-nu**2 + 1)]]

In [5]: LHS_arr = c*L/2 * np.transpose(N).dot(N)

In [6]: RHS_arr = V/2. * D_pstrs
```

Define the system as a symbolic matrix

```
In [7]: system = Matrix(LHS_arr - RHS_arr)

In [8]: eq1 = system[0,0] + system[0,1] + system[0,2]
        eq2 = system[1,0] + system[1,1] + system[1,2]
        eq3 = system[2,0] + system[2,1] + system[2,2]
```

```
In [9]: ans = solve([eq1,eq3],(nu,c))
        nu_5 = ans[0][0]
        c_5 = ans[0][1]
```

```
In [10]: nsimplify(nu_5)
```

```
Out[10]: 1/3
```

```
In [11]: nsimplify(c_5)
```

```
Out[11]: V*Y/L
```

```
In [12]: c_5.subs(V,3**0.5/2*L**2*t)
```

```
Out[12]: 0.866025403784439*L*Y*t
```

therefore; $\nu = 1/3$ and $c = \frac{\sqrt{3}}{2}ELt$

0.3 Problem 7.6

Show that for plane strain $\nu = 1/4$ and $c = \frac{8\sqrt{3}}{15}ELt$ using Equation 7.13

```
In [13]: D_pstrn = Y / ((1+nu)*(1-2*nu))*np.array(([1-nu,nu,0],[nu,1-nu,0],[0,0,(1-2*nu)/2]))
         print D_pstrn
```

```
[[Y*(-nu + 1)/((-2*nu + 1)*(nu + 1)) Y*nu/((-2*nu + 1)*(nu + 1)) 0]
 [Y*nu/((-2*nu + 1)*(nu + 1)) Y*(-nu + 1)/((-2*nu + 1)*(nu + 1)) 0]
 [0 0 Y*(-nu + 1/2)/((-2*nu + 1)*(nu + 1))]]
```

```
In [14]: RHS_strn = V/2. * D_pstrn
         system2 = Matrix(LHS_arr - RHS_strn)
```

```
In [15]: eq1_2 = system2[0,0] + system2[0,1] + system2[0,2]
         eq2_2 = system2[1,0] + system2[1,1] + system2[1,2]
         eq3_2 = system2[2,0] + system2[2,1] + system2[2,2]
```

```
In [16]: ans_2 = solve([eq1_2,eq2_2,eq3_2],(nu,c))
         nu_6 = ans_2[0][0]
         c_6 = ans_2[0][1]
```

```
In [17]: nsimplify(nu_6)
```

```
Out[17]: 1/4
```

```
In [18]: nsimplify(c_6)
```

```
Out[18]: 16*V*Y/(15*L)
```

therefore, for plane strain $\nu = 1/4$ and $c = \frac{8\sqrt{3}}{15}ELt$

0.4 Problem 7.11

build the 3D elasticity matrix

```
In [19]: D_3d = Y / ((1+nu)*(1-2*nu))*np.array(([1-nu,nu,0,nu,0,0],
                                                 [nu,1-nu,0,nu,0,0],
                                                 [0,0,(1-2*nu)/2,0,0,0],
                                                 [nu, nu,0,1-nu,0,0],
                                                 [0,0,0,0,(1-2*nu)/2,0],
                                                 [0,0,0,0,0,(1-2*nu)/2]))
```

```
In [20]: RHS_3d = V/2. * D_3d
         RHS_3d.shape
```

```
Out[20]: (6, 6)
```

build L_c array

```
In [21]: upper = np.eye(12,18)* c*L
         lower = np.roll(np.eye(6,18)*c_1*2**0.5 *L,12)
         Lc = np.concatenate((upper,lower),axis=0)
         # print Lc
```

bulid the N array for FCC

```
In [22]: n_1 = np.array(([1,0,0,0,0,0]))
         n_2 = np.array(([1,0,0,0,0,0]))
         n_3 = np.array(([.25,0.75,3.**0.5/4.,0,0,0]))
         n_4 = np.array(([.25,0.75,3.**0.5/4.,0,0,0]))
         n_5 = np.array(([0.25,0.75,-(3.**0.5)/4.,0,0,0]))
         n_6 = np.array(([0.25,0.75,-(3.**0.5)/4.,0,0,0]))
         n_7 = np.array(([0.25,3/36.,(3.**0.5)/12.,6/9.,18**0.5/18.,6**0.5/6.]))
         n_8 = np.array(([0.25,3/36.,(3.**0.5)/12.,6/9.,18**0.5/18.,6**0.5/6.]))
         n_9 = np.array(([0.25,3/36.,-(3**0.5)/12.,2./3.,18.**0.5/18.,-(6.**0.5)/6.]))
         n_10 = np.array(([0.25,3/36.,-(3**0.5)/12.,2./3.,18.**0.5/18.,-(6.**0.5)/6.]))
         n_11 = np.array(([0,3./9.,0,2./3.,-(18.**0.5)/9.,0]))
         n_12 = np.array(([0,3./9.,0,2./3.,-(18.**0.5)/9.,0]))
         n_13 = np.array(([0,4./3.,0,2./3.,2.*6**0.5/(3.*3.**0.5),0]))
         n_14 = np.array(([0,4./3.,0,2./3.,2.*6**0.5/(3.*3.**0.5),0]))
         n_15 = np.array(([1, 1/3., 3.**0.5/3., 2./3., -(18.**0.5)/9,-6.**0.5/3.]))
         n_16 = np.array(([1, 1/3., 3.**0.5/3., 2./3., -(18.**0.5)/9,-6.**0.5/3.]))
         n_17 = np.array(([1, 1/3., -(3.**0.5)/3., 2./3., -(18.**0.5)/9.,6.**0.5/3.]))
         n_18 = np.array(([1, 1/3., -(3.**0.5)/3., 2./3., -(18.**0.5)/9.,6.**0.5/3.]))
         N_18 = np.array((n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9,n_10,n_11,n_12,n_13,n_14,n_15,n_16,n_17,n
```

```
In [23]: LHS_arr = 0.25 * np.transpose(N_18).dot(Lc).dot(N_18)
         LHS_arr.shape
```

```
Out[23]: (6, 6)
```

```
In [24]: system3 = Matrix(LHS_arr - RHS_3d)
```

```
In [25]: eq1_3 = system3[0,0] + system3[0,1] + system3[0,2] + system3[0,3] + system3[0,4] + system3[0,5
         eq3_3 = system3[2,0] + system3[2,1] + system3[2,2] + system3[2,3] + system3[2,4] + system3[2,5
         eq5_3 = system3[4,0] + system3[4,1] + system3[4,2] + system3[4,3] + system3[4,4] + system3[4,5
```

```
In [26]: ans_3 = solve([eq1_3,eq3_3,eq5_3],(nu,c, c_1))
         nu_3 = ans_3[0][0]
         c_3 = ans_3[0][1]
         c1_3 = ans_3[0][2]
         ans_3
```

```
Out[26]: [(0.250000000000000, 0.800000000000001*V*Y/L, 0.0707106781186546*V*Y/L)]
```

which is the equivalent of $c = \frac{2\sqrt{2}EL^2}{5}$, $c_1 = \frac{EL^2}{5}$, $\nu = 1/4$

0.5 Problem 7.9

In [27]:
```
Lc_9=Lc[0:12,0:12]
```

In [28]:
```
LHS_arr_9 = 0.25 * np.transpose(N_18[0:12,:]).dot(Lc_9).dot(N_18[0:12,:])
LHS_arr_9.shape
```

Out[28]: (6, 6)

In [29]:
```
system9 = Matrix(LHS_arr_9 - RHS_3d)
```

In [30]:
```
eq1_9 = system9[0,0] + system9[0,1] + system9[0,2] + system9[0,3] + system9[0,4] + system9[0,5]
eq3_9 = system9[2,0] + system9[2,1] + system9[2,2] + system9[2,3] + system9[2,4] + system9[2,5]
eq5_9 = system9[4,0] + system9[4,1] + system9[4,2] + system9[4,3] + system9[4,4] + system9[4,5]
```

In [41]:
```
ans_9 = solve([eq1_9,eq3_9],(nu,c))
nu_9 = ans_9[0][0]
c_9 = ans_9[0][1]
print nu_9
print c_9
```

```
0.197712981359384
0.781007272614229*V*Y/L
```