HW9_Calculations

December 6, 2019

1 HW9: Finite Elements (Rayleigh-Ritz) and Stability

[1]:
```
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
plt.rc('text',usetex=True)
init_printing()
plt.rcParams['ytick.right']='True'
plt.rcParams['ytick.direction']='in'
plt.rcParams['ytick.labelsize']=26
plt.rcParams['xtick.labelsize']=26
plt.rcParams['xtick.minor.visible']=True
plt.rcParams['ytick.minor.visible']=True
plt.rcParams['xtick.major.size']=8
plt.rcParams['xtick.minor.size']=4
plt.rcParams['ytick.major.size']=8
plt.rcParams['ytick.minor.size']=4
plt.rcParams['lines.markersize']=np.sqrt(36)
```

1.1 Initialize the degrees of freedom

[2]:
```
alph = zeros(6,1)
for i in range(len(alph)):
    alph[i] = Symbol('alpha_{}'.format(i+1),real=True)
```

[3]:
```
alph
```

[3]:

$$\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\alpha_4\\\alpha_5\\\alpha_6\end{bmatrix}$$

```
[4]: x = Symbol('x',real=True)
y = Symbol('y',real=True)

a1 = Symbol('a_1',real=True)
a2 = Symbol('a_2',real=True)
a3 = Symbol('a_3',real=True)

l = Symbol('l',real=True)
w = Symbol('w',real=True)
h = Symbol('h',real=True)

N1_1 = a1 + a2*x + a3 *y #shape function for node 1 subscript (elem: no.)
N3_1 = a1 + a2*x + a3 *y
N1_2 = a1 + a2*x + a3 *y
N1_3 = a1 + a2*x + a3 *y
N3_3 = a1 + a2*x + a3 *y
```

1.2 Determine the shape functions

```
[5]: # Triangle 1
sys1_1 = solve_linear_system(Matrix([[1.,0,l,1], [1.,0,0,0], [1,0.
 ↪5*w,l,0]]),a1,a2,a3)
sys2_1 = solve_linear_system(Matrix([[1.,0,l,0], [1.,0,0,0], [1,0.
 ↪5*w,l,1]]),a1,a2,a3)

# Triangle 2
sys1_2 = solve_linear_system(Matrix([[1.,0.5*w,l,1], [1.,0,0,0],␣
 ↪[1,w,0,0]]),a1,a2,a3)

# Triangle 3
sys1_3 = solve_linear_system(Matrix([[1.,0.5*w,l,1], [1.,w,0,0],␣
 ↪[1,w,l,0]]),a1,a2,a3)
sys2_3 = solve_linear_system(Matrix([[1.,0.5*w,l,0], [1.,w,0,0],␣
 ↪[1,w,l,1]]),a1,a2,a3)
```

```
[6]: # Triangle 1
N1_1 = N1_1.subs(sys1_1)
N3_1 = N3_1.subs(sys2_1)

# Triangle 2
N1_2 = N1_2.subs(sys1_2)

# Triangle 3
N1_3 = N1_3.subs(sys1_3)
N3_3 = N3_3.subs(sys2_3)
```

1.2.1 Look for the displacement field u and v

[21]:
```
N3_1
```

[21]:

$$\frac{2.0x}{w}$$

[8]:
```
# Triangle 1
u1_1 = N1_1*alph[0] + N3_1*alph[2]
v1_1 = N1_1*alph[1] + N3_1*alph[3]

# Triangle 2
u1_2 = N1_2*alph[2]
v1_2 = N1_2*alph[3]

# Triangle 3
u1_3 = N1_3*alph[2] + N3_3*alph[-2]
v1_3 = N1_3 * alph[3] + N3_3*alph[-1]
```

[9]:
```
eps_1 = Matrix([[u1_1.diff(x),0.5*(u1_1.diff(y) + v1_1.diff(x)) ], [0.5*(u1_1.
↪diff(y) + v1_1.diff(x)), v1_1.diff(y)]])
eps_2 = Matrix([[u1_2.diff(x),0.5*(u1_2.diff(y) + v1_2.diff(x)) ], [0.5*(u1_2.
↪diff(y) + v1_2.diff(x)), v1_2.diff(y)]])
eps_3 = Matrix([[u1_3.diff(x),0.5*(u1_3.diff(y) + v1_3.diff(x)) ], [0.5*(u1_3.
↪diff(y) + v1_3.diff(x)), v1_3.diff(y)]])
```

1.2.2 Material properties

[22]:
```
mu = 1.e7/2
nu = 0.
lmbda = 0.
P = 10.
# l = w = 1
# h = 0.01

# A_i are the elemental volumes
A3 = A1 = 1/4*w*l*h
A2 = 2*A3
# Defining the potential energy
U = mu*trace(eps_1*eps_1)*A1 + mu*trace(eps_3*eps_3)*A3 +␣
↪mu*trace(eps_2*eps_2)*A2 - P*alph[4]
```

[23]:
```
eqns = []
for i in range(len(alph)):
    eqns.append(U.diff(alph[i]))
```

[24]:
```
eqns[0].subs(subs_list)
eps_1[0,0]
```

[24]:

$$-\frac{2.0\alpha_1}{w} + \frac{2.0\alpha_3}{w}$$

[25]:
```
soln = solve(eqns,alph)
```

[26]:
```
alph_new = alph.subs(soln)
```

[27]:
```
alph_new
```

[27]:

$$\left[\begin{matrix}\frac{8.0\cdot10^{-6}l^3\left(256.0l^6+192.0l^4w^2+24.0l^2w^4-w^6\right)}{hw(512.0l^8+640.0l^6w^2+176.0l^4w^4+24.0l^2w^6+w^8)}\\\frac{6.4\cdot10^{-5}l^6\left(16.0l^2+5.0w^2\right)}{h(512.0l^8+640.0l^6w^2+176.0l^4w^4+24.0l^2w^6+w^8)}\\\frac{3.2\cdot10^{-5}l^3\left(2.0l^2+w^2\right)}{hw(16.0l^4+16.0l^2w^2+w^4)}\\\frac{4.0\cdot10^{-6}l^2w^2}{h(32.0l^4+8.0l^2w^2+w^4)}\\\frac{8.0\cdot10^{-6}l\left(256.0l^8+256.0l^6w^2+104.0l^4w^4+19.0l^2w^6+w^8\right)}{hw(512.0l^8+640.0l^6w^2+176.0l^4w^4+24.0l^2w^6+w^8)}\\-\frac{8.0\cdot10^{-6}l^2\left(128.0l^6+56.0l^4w^2+16.0l^2w^4+w^6\right)}{h(512.0l^8+640.0l^6w^2+176.0l^4w^4+24.0l^2w^6+w^8)}\end{matrix}\right]$$

[28]:
```
subs_list = [(l,1), (w,1), (h,0.01)]
```

[29]:
```
alph_new = alph_new.subs(subs_list)
alph_new * 10**6
```

[29]:

$$\left[\begin{matrix}278.492239467849\\99.3348115299335\\290.909090909091\\9.75609756097561\\376.053215077605\\-118.847006651885\end{matrix}\right]$$

[30]:
```
import matplotlib.tri as tri
```

[39]:
```
u = np.array([alph_new],float).flatten()
u_def = u * 5.e2
```

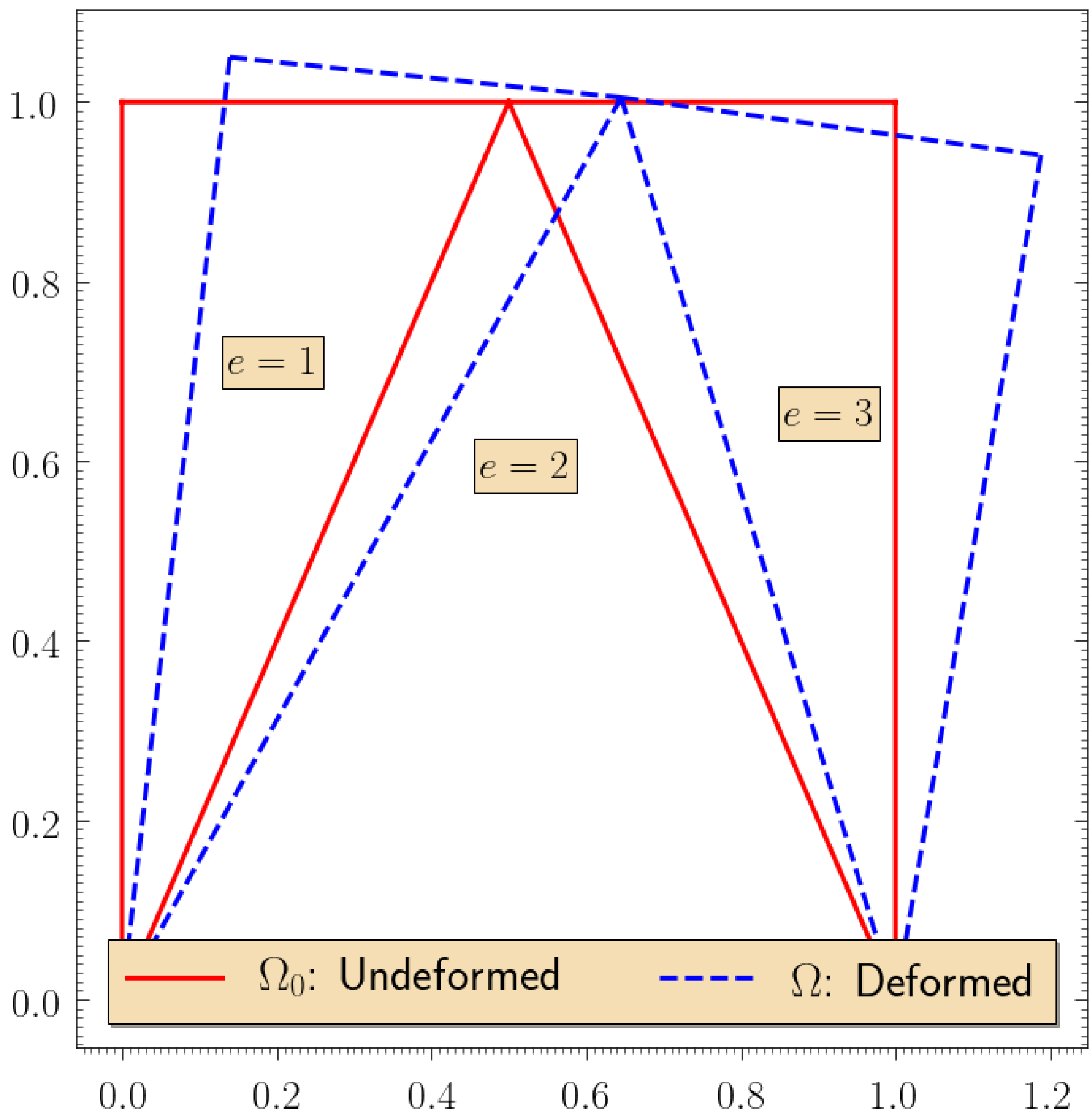
[40]:
```
xnodes = np.array([0., 1., 1., 0.5, 0.])
ynodes = np.array([0., 0., 1., 1, 1])
conn = [[0,3,4], [0,1,3], [1,2,3]]
triangles = tri.Triangulation(xnodes, ynodes, triangles = conn)
```

[41]:

```
xnodes_def = np.array([0.,1.,1+u_def[-2],0.5+u_def[-4],u_def[0]])
ynodes_def = np.array([0.,0.,1+u_def[-1],1+u_def[-3],1+u_def[1]])
triangles_def = tri.Triangulation(xnodes_def, ynodes_def, triangles = conn)
```

[44]:

```
fig,ax = plt.subplots(1,1,figsize=(10,10))
ax.triplot(triangles,'r-',lw=3,label=r'$\Omega_0$: Undeformed')
ax.triplot(triangles_def,'b--',lw=3,label=r'$\Omega$: Deformed')
# ax.grid(which='major')
ax.text(0.15,0.65,r'$e = 1$',fontsize=26,transform=ax.
 ↪transAxes,bbox=dict(facecolor='wheat',edgecolor='k'))
ax.text(0.4,0.55,r'$e = 2$',fontsize=26,transform=ax.
 ↪transAxes,bbox=dict(facecolor='wheat',edgecolor='k'))
ax.text(0.7,0.6,r'$e = 3$',fontsize=26,transform=ax.
 ↪transAxes,bbox=dict(facecolor='wheat',edgecolor='k'))
ax.tick_params(pad=10)
ax.xaxis.set_minor_locator(AutoMinorLocator(20))
ax.yaxis.set_minor_locator(AutoMinorLocator(20))
h,l = ax.get_legend_handles_labels()
ax.legend(loc='lower center',handles = [h[0],h[2]],labels =␣
 ↪[l[0],l[2]],fontsize=30,ncol=4,fancybox=False,facecolor='wheat',edgecolor='k',shadow=True)
fig.tight_layout()
fig.savefig(r'plotP1.eps')
```



[ ]:
```python
alph_subs = {alph[i]:alph_new[i] for i in range(len(alph))}
```

[ ]:
```python
# Calculate the strains in the elements

# Element 1:
eps1 = eps_1.subs(alph_subs).subs(subs_list)

# Element 2:
eps2 = eps_2.subs(alph_subs).subs(subs_list)

# Element 3:
eps3 = eps_3.subs(alph_subs).subs(subs_list)
```

[ ]:
```
eps1*10**6
```

[ ]:
```
eps2*10**6
```

[ ]:
```
eps3*10**6
```

[ ]:
```
eps1*10**7
```

[ ]:
```
eps2*10**7
```

[ ]:
```
eps3*10**7
```