demoHW9_modified

December 4, 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']=22
plt.rcParams['xtick.labelsize']=22
plt.rcParams['xtick.minor.visible']=True
plt.rcParams['ytick.minor.visible']=True
plt.rcParams['xtick.major.size']=6
plt.rcParams['xtick.minor.size']=3
plt.rcParams['ytick.major.size']=6
plt.rcParams['ytick.minor.size']=3
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

[7]:
```
N3_1
```

[7]:

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

[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

[10]:
```
mu = 1.e7
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]
```

[11]:
```
eqns = []
for i in range(len(alph)):
    eqns.append(U.diff(alph[i]))
```

[12]:
```python
soln = solve(eqns,alph)
```

[13]:
```python
alph = alph.subs(soln)
```

[14]:
```python
alph
```

[14]:

$$\left[\begin{matrix}\frac{4.0\cdot10^{-6} l^{3} \left(256.0 l^{6} + 192.0 l^{4} w^{2} + 24.0 l^{2} w^{4} - w^{6}\right)}{h w \left(512.0 l^{8} + 640.0 l^{6} w^{2} + 176.0 l^{4} w^{4} + 24.0 l^{2} w^{6} + w^{8}\right)}\\\frac{3.2\cdot10^{-5} l^{6} \left(16.0 l^{2} + 5.0 w^{2}\right)}{h \left(512.0 l^{8} + 640.0 l^{6} w^{2} + 176.0 l^{4} w^{4} + 24.0 l^{2} w^{6} + w^{8}\right)}\\\frac{1.6\cdot10^{-5} l^{3} \left(2.0 l^{2} + w^{2}\right)}{h w \left(16.0 l^{4} + 16.0 l^{2} w^{2} + w^{4}\right)}\\\frac{2.0\cdot10^{-6} l^{2} w^{2}}{h \left(32.0 l^{4} + 8.0 l^{2} w^{2} + w^{4}\right)}\\\frac{4.0\cdot10^{-6} l \left(256.0 l^{8} + 256.0 l^{6} w^{2} + 104.0 l^{4} w^{4} + 19.0 l^{2} w^{6} + w^{8}\right)}{h w \left(512.0 l^{8} + 640.0 l^{6} w^{2} + 176.0 l^{4} w^{4} + 24.0 l^{2} w^{6} + w^{8}\right)}\\- \frac{4.0\cdot10^{-6} l^{2} \left(128.0 l^{6} + 56.0 l^{4} w^{2} + 16.0 l^{2} w^{4} + w^{6}\right)}{h \left(512.0 l^{8} + 640.0 l^{6} w^{2} + 176.0 l^{4} w^{4} + 24.0 l^{2} w^{6} + w^{8}\right)}\end{matrix}\right]$$

[15]:
```python
subs_list = [(l,1), (w,1), (h,0.01)]
```

[16]:
```python
alph = alph.subs(subs_list)
alph
```

[16]:

$$\left[\begin{matrix}0.000139246119733925\\4.96674057649667\cdot10^{-5}\\0.000145454545454545\\4.8780487804878\cdot10^{-6}\\0.000188026607538803\\-5.94235033259423\cdot10^{-5}\end{matrix}\right]$$

[17]:
```python
import matplotlib.tri as tri
```

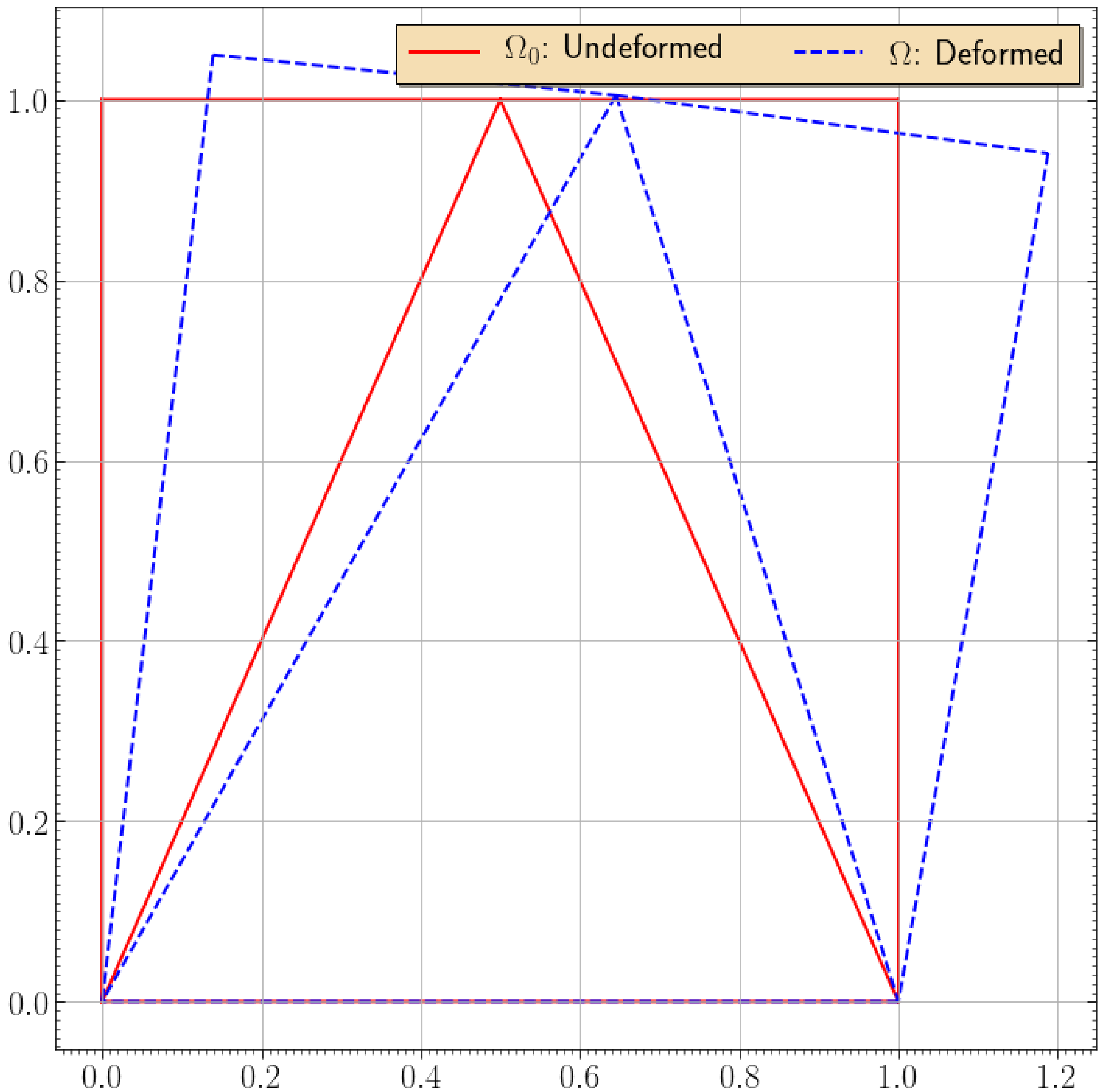
[18]:
```python
u = np.array([alph],float).flatten()
u_def = u * 1.e3
```

[19]:
```python
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)
```

[20]:
```python
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)
```

[26]:
```python
fig,ax = plt.subplots(1,1,figsize=(10,10))
ax.triplot(triangles,'r-',lw=2,label=r'$\Omega_0$: Undeformed')
ax.triplot(triangles_def,'b--',lw=2,label=r'$\Omega$: Deformed')
```

```
ax.grid(which='major')
ax.xaxis.set_minor_locator(AutoMinorLocator(20))
ax.yaxis.set_minor_locator(AutoMinorLocator(20))
h,l = ax.get_legend_handles_labels()
ax.legend(loc=0,handles = [h[0],h[2]],labels =␣
 ↪[l[0],l[2]],fontsize=22,ncol=4,fancybox=False,facecolor='wheat',edgecolor='k',shadow=True)
fig.tight_layout()
```




```
[ ]:
```