Project 2

Omkar Mulekar
AERO 4630-002
Aerospace Structural Dynamics

22 February 2019

Problem 1: Conducting a Tensile Test

Part 1a: Lamé Parameters

For $E = 200$ GPa $= 200 \times 10^9$ N/m^2 and $\nu = 0.3$, the Lamé parameters λ and μ can be calculated as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{1}$$

and

$$\mu = \frac{E}{2(1+\nu)} \tag{2}$$

Equations 1 and 2 yield $\lambda = 1.153 \times 10^{11}$ N/m^2 and $\mu = 7.69 \times 10^{10}$ N/m^2.

Part 1b: Tensile Test

The Paraview output for with stress plotted over the right face of the beam is show in Fig 1

The *Slice*, *ExtractSurface*, and *IntegrateVariables* filters were used to determine that the total force on the face in the x-direction is 936 kN.

Part 1c: Average Stress vs Strain

The average stresses and strains for displacements of the right hand side ranging from 0.001 mm to 0.004 mm were plotted, and are shown in Fig 2

The code used for Problem 1 is shown in **Appendix 1**.

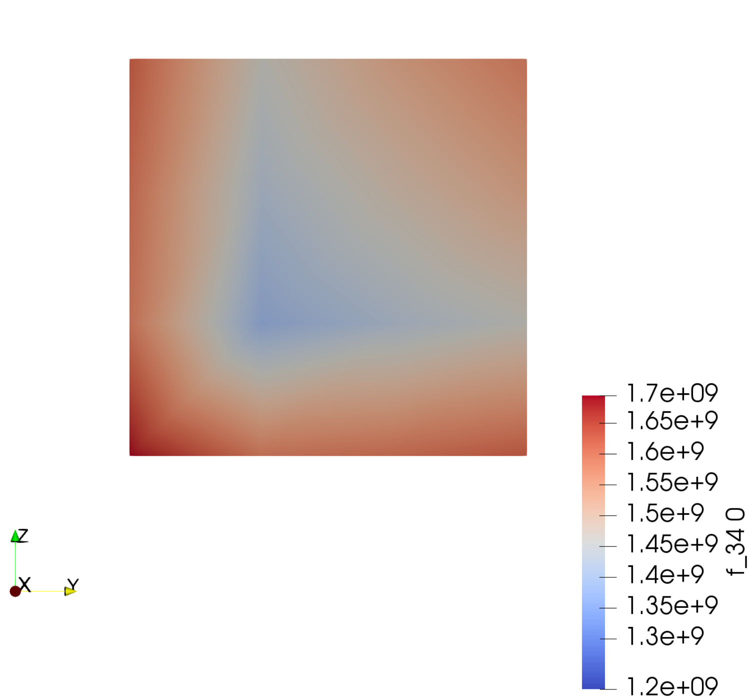

Figure 1: Paraview output for problem 1b

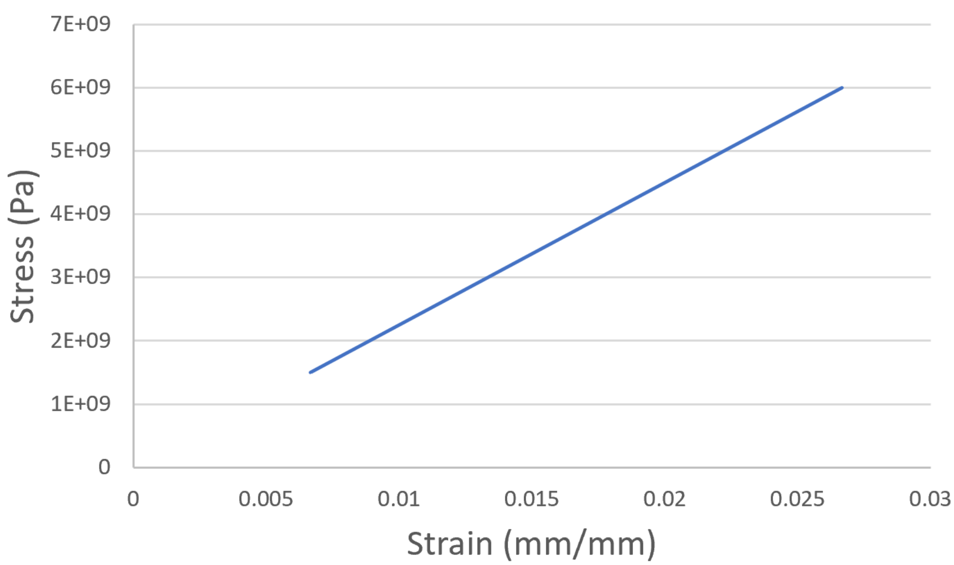

Figure 2: Paraview output for problem 1b

Problem 2: Applying a Force on a Beam Clamped at Both Ends

Part 2a: Vertical Displacement given Force Application Area

A force was applied to an area at the center of beam of length $(L - 2a)$ and width w, where a is $L/4$. The resulting vertical deflection at the middle is -2.322×10^{-8} m.

The Python code used for problem 2a is shown in **Appendix 2a**.

Part 2b: Changing Applied Force Application Area

Next, displacements at the center of the beam were calculated for different force application areas. The length along the beam at the center at which the force was distributed was $a = L/N$ for $N = 3, 4, 5$ and 6. A plot of vertical displacement of the beam over N is shown in Fig 3

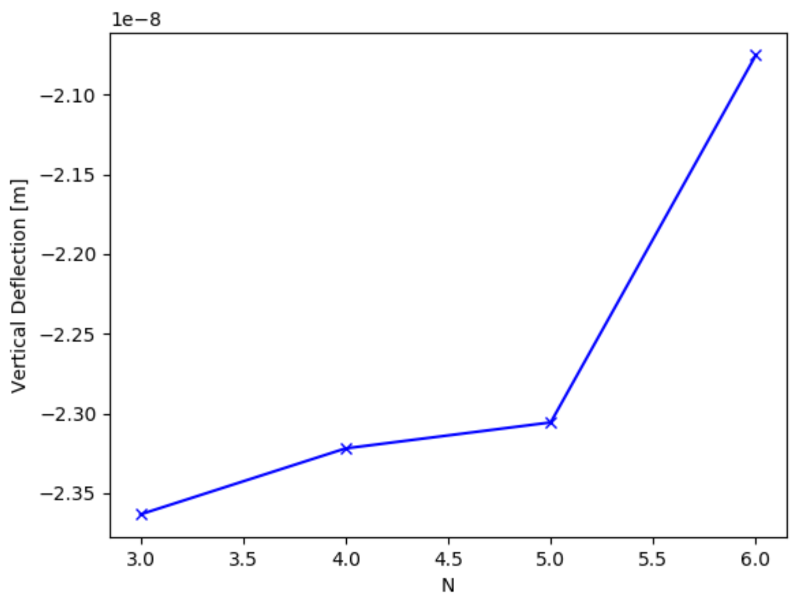

Figure 3: Error vs Mesh Sizing for Problem 2a

The code used in part 2b is provided in **Appendix 2b**.

Appendix 1a: Code for Problem 1

```
"""
Python script for Problem 1 of Project 2

Original Author:  Vinamra Agrawal
Date:             January 25, 2019

Edited By:          Omkar Mulekar
Date:             February 10, 2019

"""

from __future__ import print_function
from fenics import *
from ufl import nabla_div

length = .150;
W = 0.025;
H = 0.025;


mu = 7.69e10
rho = 7800
lambda_ = 1.153e11
g = 10

mesh = BoxMesh(Point(0,0,0),Point(length,W,H),10,3,3)
V = VectorFunctionSpace(mesh,'P',1)

tol = 1E-14

def boundary_left(x,on_boundary):
    return (on_boundary and near(x[0],0))

def boundary_right(x,on_boundary):
    return on_boundary and near(x[0],length)

bc_left = DirichletBC(V,Constant((0,0,0)),boundary_left)
bc_right = DirichletBC(V,Constant((0.004,0,0)),boundary_right)

def epsilon(u):
    return 0.5*(nabla_grad(u) + nabla_grad(u).T)
```

```
def sigma(u):
    return lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
        epsilon(u).T)

u = TrialFunction(V)
d = u.geometric_dimension()
v = TestFunction(V)

f = Constant((0,0,0))
T = Constant((0,0,0))

a = inner(sigma(u),epsilon(v))*dx
L = dot(f,v)*dx + dot (T,v)*ds

u = Function(V)
solve(a == L, u, [bc_left,bc_right])

vtkfile_u = File('deflection.pvd')
vtkfile_u << u

W = TensorFunctionSpace(mesh, "Lagrange", 1)
stress = lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
    epsilon(u).T)

vtkfile_s = File('stress.pvd')
vtkfile_s << project(stress,W)

s = sigma(u) - (1./3)*tr(sigma(u))*Identity(d) # deviatoric stress
von_Mises = sqrt(3./2*inner(s, s))
X = FunctionSpace(mesh, 'P', 1)
vtkfile_von = File('von_Mises.pvd')
vtkfile_von << project(von_Mises, X)
```

Appendix 2a: Code for Problem 2a

```
"""
Python script for Part 1b of Project 2a

Original Author:  Vinamra Agrawal
Date:             January 25, 2019

Edited By:           Omkar Mulekar
Date:             February 10, 2019

"""

from __future__ import print_function
from fenics import *
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from ufl import nabla_div

length = .150;
W = 0.025;
H = 0.025;

F = -10

A = length / 4

mu = 7.69e10
rho = 7800
lambda_ = 1.153e11
g = 10

mesh = BoxMesh(Point(0,0,0),Point(length,W,H),10,3,3)
V = VectorFunctionSpace(mesh,'P',1)

tol = 1E-14

def boundary_left(x,on_boundary):
    return (on_boundary and near(x[0],0))

def boundary_right(x,on_boundary):
    return on_boundary and near(x[0],length)
```

```
bc_left = DirichletBC(V,Constant((0,0,0)),boundary_left)
bc_right = DirichletBC(V,Constant((0,0,0)),boundary_right)

def epsilon(u):
    return 0.5*(nabla_grad(u) + nabla_grad(u).T)

def sigma(u):
    return lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
        epsilon(u).T)

u = TrialFunction(V)
d = u.geometric_dimension()
v = TestFunction(V)

f = Constant((0,0,0))
T = Expression(('0.0','(x[0] >= a && x[0] <= (L-a) && near(x[1],W)
    ) ? (F/(H*(L-2*a))) : 0.0','0.0'),L=length,a=A,F=F,W=W,H=H,
    degree=1)

a = inner(sigma(u),epsilon(v))*dx
L = dot(f,v)*dx + dot (T,v)*ds

u = Function(V)
solve(a == L, u, [bc_left,bc_right])

w = u(length/2.0,W/2.0,H/2.0)
print(w[1])

vtkfile_u = File('deflection.pvd')
vtkfile_u << u

W = TensorFunctionSpace(mesh, "Lagrange", 1)
stress = lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
    epsilon(u).T)

vtkfile_s = File('stress.pvd')
vtkfile_s << project(stress,W)

s = sigma(u) - (1./3)*tr(sigma(u))*Identity(d) # deviatoric stress
von_Mises = sqrt(3./2*inner(s, s))
X = FunctionSpace(mesh, 'P', 1)
vtkfile_von = File('von_Mises.pvd')
vtkfile_von << project(von_Mises, X)
```

Appendix 2a: Code for Problem 2b

```
"""
Python script for Part 1b of Project 2b

Original Author:  Vinamra Agrawal
Date:             January 25, 2019

Edited By:           Omkar Mulekar
Date:             February 10, 2019

"""

from __future__ import print_function
from fenics import *
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
from ufl import nabla_div

length = .150;
W = 0.025;
H = 0.025;

F = -10

n = [3,4,5,6]
# vertical_deflection = [] * len(n)
w = [0] * len(n)

for i in range(len(n)):
    print(i)
    A = length / n[i]

    mu = 7.69e10
    rho = 7800
    lambda_ = 1.153e11
    g = 10

    mesh = BoxMesh(Point(0,0,0),Point(length,W,H),10,3,3)
    V = VectorFunctionSpace(mesh,'P',1)

    tol = 1E-14
```

```
def boundary_left(x,on_boundary):
    return (on_boundary and near(x[0],0))

def boundary_right(x,on_boundary):
    return on_boundary and near(x[0],length)

bc_left = DirichletBC(V,Constant((0,0,0)),boundary_left)
bc_right = DirichletBC(V,Constant((0,0,0)),boundary_right)

def epsilon(u):
    return 0.5*(nabla_grad(u) + nabla_grad(u).T)

def sigma(u):
    return lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
        epsilon(u).T)

u = TrialFunction(V)
d = u.geometric_dimension()
v = TestFunction(V)

f = Constant((0,0,0))
T = Expression(('0.0','(x[0] >= a && x[0] <= (L-a) && near(x
    [1],W)) ? (F/(H*(L-2*a))) : 0.0','0.0'),L=length,a=A,F=F,W=W
    ,H=H,degree=1)

a = inner(sigma(u),epsilon(v))*dx
L = dot(f,v)*dx + dot (T,v)*ds

u = Function(V)
solve(a == L, u, [bc_left,bc_right])

w[i] = u(length/2.0,W/2.0,H/2.0)[1]
print(w[1])
# vertical_deflection[i] = w[1]

vtkfile_u = File('deflection.pvd')
vtkfile_u << u

W = TensorFunctionSpace(mesh, "Lagrange", 1)
stress = lambda_*nabla_div(u)*Identity(d) + mu*(epsilon(u) +
    epsilon(u).T)

vtkfile_s = File('stress.pvd')
vtkfile_s << project(stress,W)
```

```
    s = sigma(u) - (1./3)*tr(sigma(u))*Identity(d) # deviatoric
        stress
    von_Mises = sqrt(3./2*inner(s, s))
    X = FunctionSpace(mesh, 'P', 1)
    vtkfile_von = File('von_Mises.pvd')
    vtkfile_von << project(von_Mises, X)
    W = 0.025

plt.figure(1)
plt.plot(n,w, 'b-x')
plt.xlabel('N')
plt.ylabel('Vertical Deflection [m]')
plt.savefig('2afig1.png')
```