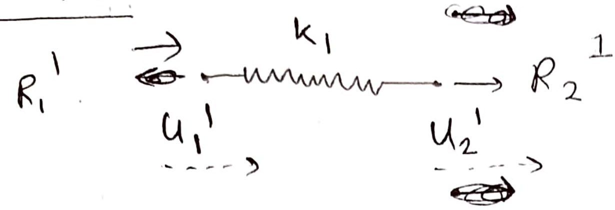
Rishi Raj, 22M0033
FEM Assignment -1
AERO MTECH(TA) STRUCTURES

1. Using Direct Method:

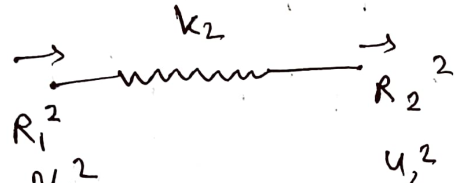
Element-1:




$$\begin{Bmatrix} R_1^1 \\ R_2^1 \end{Bmatrix} = \begin{bmatrix} K_1 & -K_1 \\ -K_1 & K_1 \end{bmatrix} \begin{Bmatrix} U_1^1 \\ U_2^1 \end{Bmatrix}$$

$$\left.\begin{aligned} K_1(-U_2^1 + U_1^1) &= R_1^1 \\ K_1(-U_1^1 + U_2^1) &= R_2^1 \end{aligned}\right\}$$

Element-2:




$$\left.\begin{aligned} K_2(U_1^2 - U_2^2) &= R_1^2 \\ K_2(U_2^2 - U_1^2) &= R_2^2 \end{aligned}\right\}$$

$$\begin{Bmatrix} R_1^2 \\ R_2^2 \end{Bmatrix} = \begin{bmatrix} K_2 & -K_2 \\ -K_2 & K_2 \end{bmatrix} \begin{Bmatrix} u_1^2 \\ u_2^2 \end{Bmatrix}$$

Simillarly for nth element

$$\begin{Bmatrix} R_1^n \\ R_2^n \end{Bmatrix} = \begin{bmatrix} K_n & -K_n \\ -K_n & K_n \end{bmatrix} \begin{Bmatrix} u_1^n \\ u_2^n \end{Bmatrix}$$

writing equilibrium eq$^n$

@ 4 nodes.

leftmost node is taken as 4th node.

@ 4th node, $R_1^1 = R_4$

@ 1st node, $R_2^1 + R_1^2 + R_1^3 + R_1^4 = R_1$

@ 2nd node, $R_2^2 + R_2^3 + R_1^5 = R_2$

@ 3rd node, $R_2^4 + R_2^5 = R_3$

Assembling all the matrices we get

$$\begin{Bmatrix} R_4 \\ R_1 \\ R_2 \\ R_3 \end{Bmatrix} = \begin{bmatrix} k_1 & -k_1 & 0 & 0 \\ -k_1 & [k_1+k_2+k_3+k_4] & -k_2-k_3 & -k_4 \\ 0 & -k_2-k_3 & k_2+k_3+k_5 & -k_5 \\ 0 & -k_4 & -k_5 & (k_4+k_5) \end{bmatrix} \begin{Bmatrix} U_4 \\ U_1 \\ U_2 \\ U_3 \end{Bmatrix} \quad - (A)$$

Energy method:

$$\Pi = \frac{1}{2} k_1 (U_2^1 - U_1^1)^2 - R_1^1 U_1^1 - R_2^1 U_2^1$$
$$+ \frac{1}{2} k_2 (U_2^2 - U_1^2)^2 - R_1^2 U_1^2 - R_2^2 U_2^2$$
$$+ \frac{1}{2} k_3 (U_2^3 - U_1^3)^2 - R_1^3 U_1^3 - R_2^3 U_2^3$$
$$+ \frac{1}{2} k_4 (U_2^4 - U_1^4)^2 - R_1^4 U_1^4 - R_2^4 U_2^4$$
$$+ \frac{1}{2} k_5 (U_2^5 - U_1^5)^2 - R_1^5 U_1^5 - R_2^5 U_2^5$$
$$= \frac{1}{2} k_1 (U_1 - U_4)^2 + \frac{1}{2} k_2 (U_2 - U_1)^2 + \frac{1}{2} k_3 (U_2 - U_1)^2$$
$$+ \frac{1}{2} k_4 (U_3 - U_1)^2 + \frac{1}{2} k_5 (U_3 - U_2)^2$$
$$- R_4 U_4 - R_1 U_1 - R_2 U_2 - R_3 U_3$$

Now,

$$\frac{\partial \Pi}{\partial U_1} = 0 \Rightarrow \frac{1}{2} k_1 \, 2 (U_1 - U_4) - \frac{1}{2} 2 (U_2 - U_1)(k_2 + k_3) - \frac{1}{2} k_4 \times 2 (U_3 - U_1) - R_1 = 0$$

$$R_1 = U_1 (k_1 + k_2 + k_3 + k_4) - k_1 U_4$$
$$U_2 (-k_2 - k_3) - k_4 U_3 \quad - (1)$$

$$\frac{\partial \Pi}{\partial U_2} = 0 \qquad R_2 = U_1 (-k_2 - k_3) + U_2 (k_2 + k_3 + k_5) + U_3 (-k_5) \quad - (2)$$

$$\frac{\partial \Pi}{\partial U_3} = 0 \qquad R_3 = U_1(-K_4) + U_2(-K_5) + U_3(K_4 + K_5) \quad \text{—(3)}$$

$$\frac{\partial \Pi}{\partial U_4} = 0 \qquad R_4 = U_4(K_1) + U_1(-K_1) \quad \text{—(4)}$$

From eq$^n$ (1), (2), (3) & (4), we get back

Same $\{F\} = [K]\{u\}$ matrix eq$^n$ (A).

# PYTHON CODE FOR FEM ASSIGNMENT 1

# question 2

# PART-B

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
elementNodes=np.array([[1 ,2],[2, 3],[2, 3],[2,4],[3, 4]])
numberElements=len(elementNodes)
numberNodes=4
```

# for structure:

# displacements: displacement vector

# force : force vector

# stiffness: stiffness matrix

In [3]:
```
displacements=np.zeros((numberNodes,1))
force=np.zeros((numberNodes,1))
stiffness=np.zeros((numberNodes,numberNodes))
```

# apply loads at node 2,3,4

In [4]:
```
force[1 : 4] = 1
```

# assembly of stiffness matrix

In [5]:
```
for i in range(numberElements):
    elementDof=elementNodes[i,]
    rows=np.array([[elementDof[0]-1, elementDof[0]-1],
                   [elementDof[1]-1, elementDof[1]-1]], dtype=np.intp)
    columns=np.array([[elementDof[0]-1, elementDof[1]-1],
                   [elementDof[0]-1, elementDof[1]-1]], dtype=np.intp)
    #print(elementDof)
```

```
    stiffness[rows,columns]=stiffness[rows,columns]+np.array([[1 ,-1],[-1 ,1]])
print(stiffness)
```

```
[[ 1. -1.  0.  0.]
 [-1.  4. -2. -1.]
 [ 0. -2.  3. -1.]
 [ 0. -1. -1.  2.]]
```

# boundary conditions and solution

In [6]:
```
prescribedDof=np.array([[0]]) # corresponding to reaction (restricted DOF)
activeDof=np.setdiff1d(np.linspace(0,numberNodes-1,num=numberNodes),prescribedDof)
```

In [7]:
```
reduced_stiffness1=np.delete(stiffness,prescribedDof , axis=0)
reduced_stiffness=np.delete(reduced_stiffness1,prescribedDof , axis=1)
reduced_force=np.delete(force,prescribedDof , axis=0)
displacements=np.linalg.solve(reduced_stiffness, reduced_force)
```

In [8]:
```
displacements
final_displacement_vector=np.zeros((numberNodes,1))
final_displacement_vector[activeDof.astype(int)]=final_displacement_vector[activeDof.a
```

In [9]:
```
final_force_vector=np.matmul(stiffness,final_displacement_vector)
```

In [10]:
```
final_force_vector
```

Out[10]:
```
array([[-3.],
       [ 1.],
       [ 1.],
       [ 1.]])
```

In [11]:
```
final_displacement_vector
```

Out[11]:
```
array([[0. ],
       [3. ],
       [3.6],
       [3.8]])
```

In [12]:
```
stiffness
```

Out[12]:
```
array([[ 1., -1.,  0.,  0.],
       [-1.,  4., -2., -1.],
       [ 0., -2.,  3., -1.],
       [ 0., -1., -1.,  2.]])
```

# PART-C

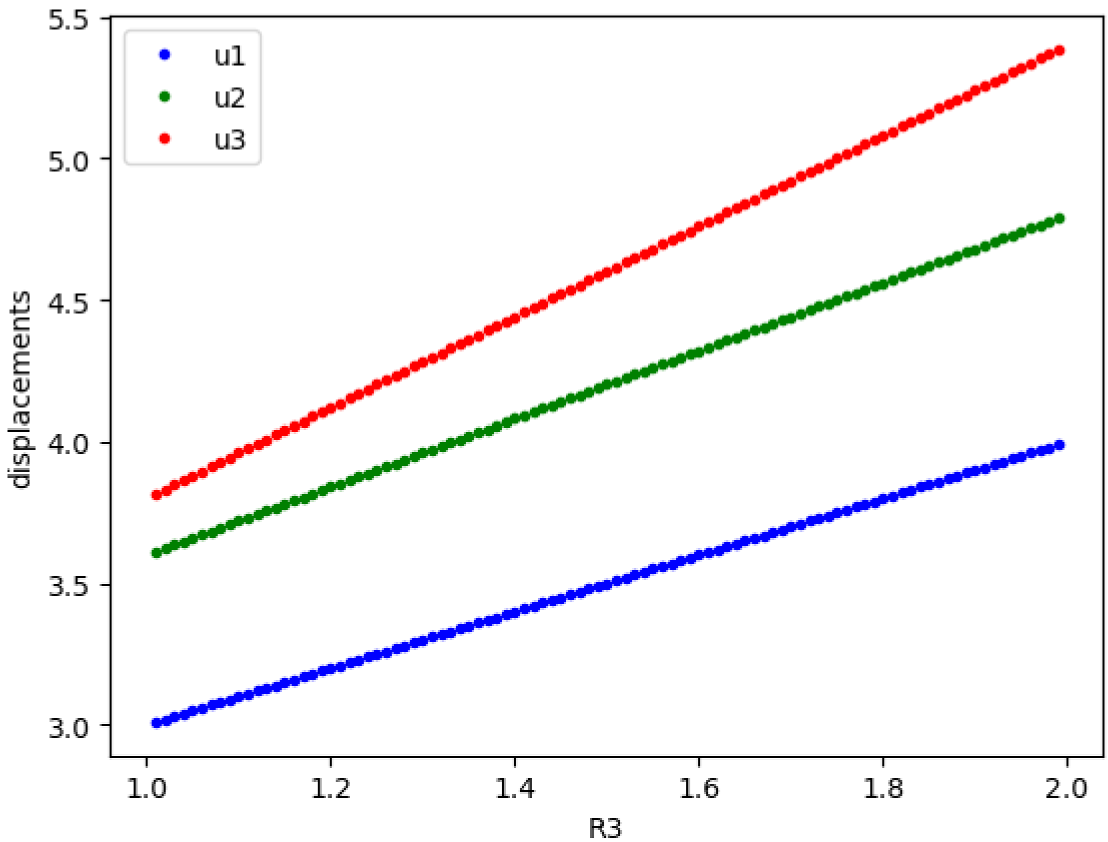
In [13]:
```
import matplotlib.pyplot as plt
```

In [14]:
```
varying_force=reduced_force
accuracy=100
for i in range(1,accuracy):
    varying_force[2,0]=varying_force[2,0]+1/accuracy
    displacements1=np.linalg.solve(reduced_stiffness, varying_force)
    plt.plot(varying_force[2,0],displacements1[0],'b.')
```

```
    plt.plot(varying_force[2,0],displacements1[1],'g.')
    plt.plot(varying_force[2,0],displacements1[2],'r.')
plt.xlabel("R3")
plt.ylabel("displacements")
plt.legend(["u1", "u2","u3"], loc=0, frameon=True)
```

Out[14]:

```
<matplotlib.legend.Legend at 0x26080ce2a60>
```




In [15]:

```
displacements1 # values at R3=2 for u2 u3 and u4 respectively
```

Out[15]:

```
array([[3.99 ],
       [4.788],
       [5.384]])
```

In [ ]: