SOLUTION

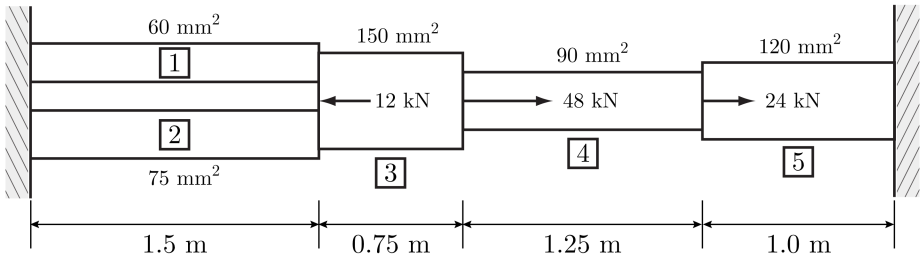
CE 325 Spring 2026 HW#5

Due Thursday, March 12, at the beginning of class

1. For the uniaxial structure shown below determine joint displacements, member end forces along with axial force diagram, and external reactions using:

a. (5 pts) Python

b. (5 pts) SAP2000



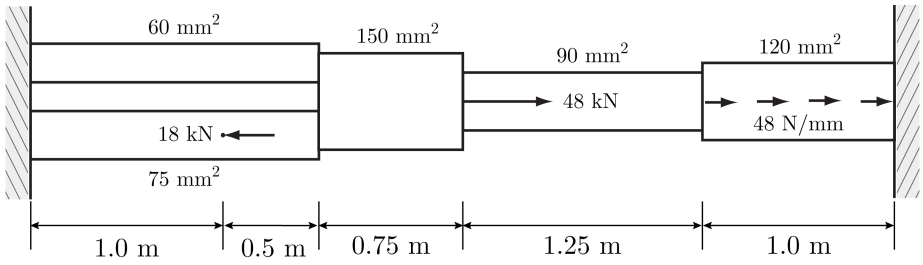
E = 20 GPa for all members

2. For the uniaxial structure shown below determine joint displacements, member end forces along with axial force diagram, and external reactions using:

a. (10 pts) By Hand

b. (5 pts) Python

c. (5 pts) SAP2000



E = 20 GPa for all members

Problem 1

Executive Summary

Problem Statement

For the uniaxial structure shown in Problem 1, determine joint displacements, member end forces along with axial force diagram, and external reactions using:

a. Python
b. SA2000

Results

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 13.623 \\ 22.754 \\ 14.783 \end{Bmatrix} mm$$

$$\begin{Bmatrix} Q_1^1 \\ Q_2^1 \end{Bmatrix} = \begin{Bmatrix} -10.9 \\ 10.9 \end{Bmatrix} kN$$

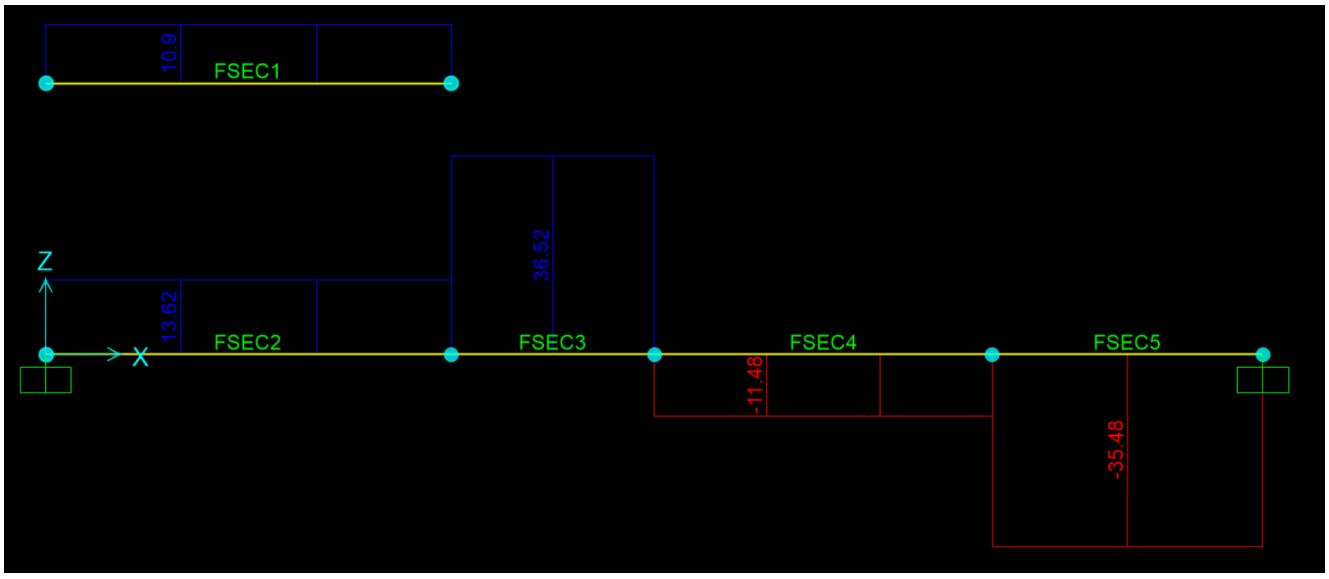
$$\begin{Bmatrix} Q_1^2 \\ Q_2^2 \end{Bmatrix} = \begin{Bmatrix} -13.6 \\ 13.6 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^3 \\ Q_2^3 \end{Bmatrix} = \begin{Bmatrix} -36.5 \\ 36.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^4 \\ Q_2^4 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -11.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^5 \\ Q_2^5 \end{Bmatrix} = \begin{Bmatrix} 35.5 \\ -35.5 \end{Bmatrix} kN$$

$$R_4 = -24.5\ kN$$
$$R_5 = -35.5\ kN$$



Problem 1

Technical Summary **see appendix for code and SAP results*

$$[k] = \frac{AE}{L}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix}$$

$$\{P\} = \begin{Bmatrix} P_1 \\ P_2 \\ P_3 \end{Bmatrix} = \begin{Bmatrix} -12 \\ 48 \\ 24 \end{Bmatrix} kN$$

$$\{S\} = \begin{bmatrix} (k_{22}^1 + k_{22}^2 + k_{11}^3) & k_{12}^3 & 0 \\ k_{21}^3 & k_{22}^3 + k_{11}^4 & k_{12}^4 \\ 0 & k_{21}^4 & k_{22}^4 + k_{11}^5 \end{bmatrix} = \begin{bmatrix} 5.80 & -4.00 & 0 \\ -4.00 & 5.44 & -1.44 \\ 0 & -1.44 & 3.84 \end{bmatrix} kN/mm$$

$$\{P\} = [S]\{d\}$$

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 13.623 \\ 22.754 \\ 14.783 \end{Bmatrix} mm$$

$$\{Q\} = [k]\{u\}$$

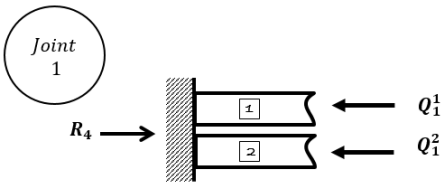
$$\begin{Bmatrix} Q_1^1 \\ Q_2^1 \end{Bmatrix} = \begin{Bmatrix} -10.9 \\ 10.9 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^2 \\ Q_2^2 \end{Bmatrix} = \begin{Bmatrix} -13.6 \\ 13.6 \end{Bmatrix} kN$$

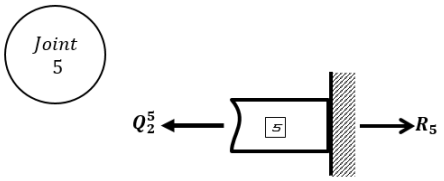
$$\begin{Bmatrix} Q_1^3 \\ Q_2^3 \end{Bmatrix} = \begin{Bmatrix} -36.5 \\ 36.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^4 \\ Q_2^4 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -11.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^5 \\ Q_2^5 \end{Bmatrix} = \begin{Bmatrix} 35.5 \\ -35.5 \end{Bmatrix} kN$$



$$R_4 = Q_1^1 + Q_1^2 = -24.5\ kN$$



$$R_5 = Q_2^5 = -35.5\ kN$$

Problem 2

Executive Summary

Problem Statement

For the uniaxial structure shown in Problem 2, determine joint displacements, member end forces along with axial force diagram, and external reactions using:

a. MATLAB
b. Python
c. SA2000

Results

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 13.623 \\ 22.754 \\ 14.783 \end{Bmatrix} mm$$

$$\begin{Bmatrix} Q_1^1 \\ Q_2^1 \end{Bmatrix} = \begin{Bmatrix} -10.9 \\ 10.9 \end{Bmatrix} kN$$

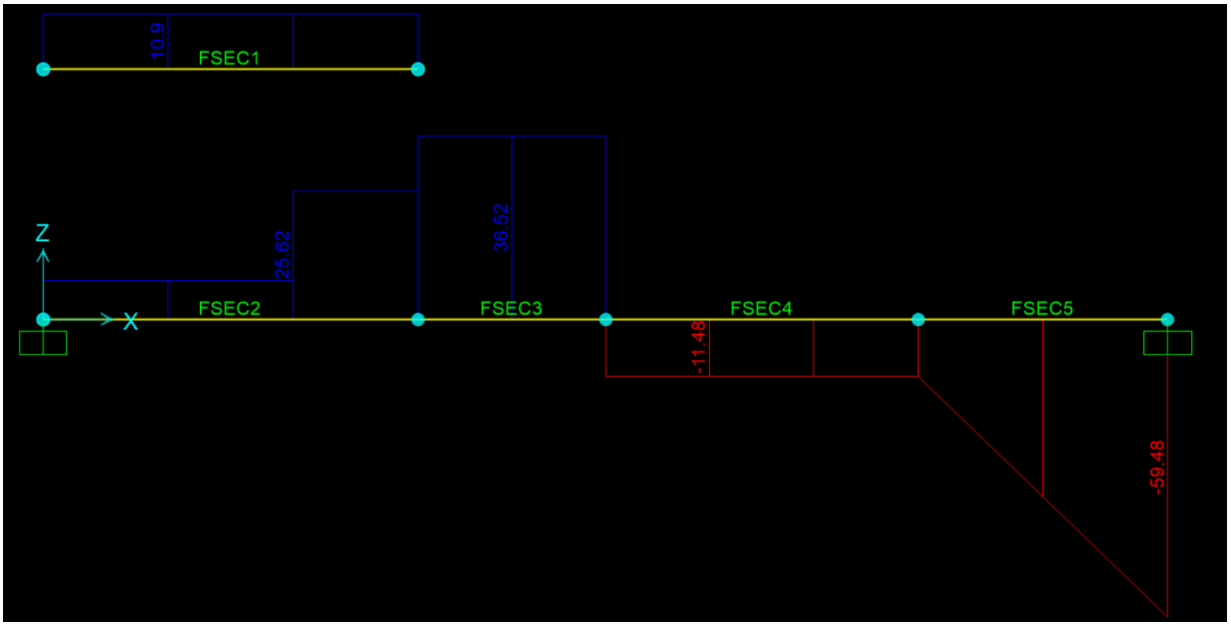
$$\begin{Bmatrix} Q_1^2 \\ Q_2^2 \end{Bmatrix} = \begin{Bmatrix} -7.62 \\ 25.6 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^3 \\ Q_2^3 \end{Bmatrix} = \begin{Bmatrix} -36.5 \\ 36.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^4 \\ Q_2^4 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -11.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^5 \\ Q_2^5 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -59.5 \end{Bmatrix} kN$$

$$R_4 = -18.5\ kN$$
$$R_5 = -59.5\ kN$$



Problem 2

Technical Summary *see appendix for code and SAP results

$$[k] = \frac{AE}{L}\begin{bmatrix} +1 & -1 \\ -1 & +1 \end{bmatrix} \qquad \{Q_f\} = W\begin{Bmatrix} -\frac{b}{L} \\ -\frac{a}{L} \end{Bmatrix} \qquad \{Q_f\} = \frac{wL}{2}\begin{Bmatrix} -1 \\ -1 \end{Bmatrix}$$

$$\{P\} = \begin{Bmatrix} P_1 \\ P_2 \\ P_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 48 \\ 0 \end{Bmatrix} kN$$

$$\{P_f\} = \begin{Bmatrix} Qf_2^1 + Qf_2^2 + Qf_1^3 \\ Qf_2^3 + Qf_1^4 \\ Qf_2^4 + Qf_1^5 \end{Bmatrix} = \begin{Bmatrix} 12 \\ 0 \\ -24 \end{Bmatrix} kN$$

$$\{S\} = \begin{bmatrix} (k_{22}^1 + k_{22}^2 + k_{11}^3) & k_{12}^3 & 0 \\ k_{21}^3 & k_{22}^3 + k_{11}^4 & k_{12}^4 \\ 0 & k_{21}^4 & k_{22}^4 + k_{11}^5 \end{bmatrix} = \begin{bmatrix} 5.80 & -4.00 & 0 \\ -4.00 & 5.44 & -1.44 \\ 0 & -1.44 & 3.84 \end{bmatrix} kN/mm$$

$$\{P\} = \{P_f\} + [S]\{d\}$$

$$\begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} 13.623 \\ 22.754 \\ 14.783 \end{Bmatrix} mm$$

$$\{Q\} = \{Q_f\} + [k]\{u\}$$

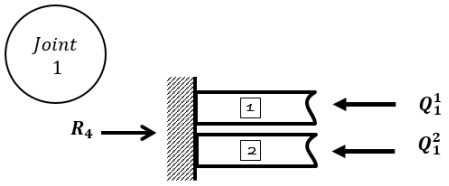
$$\begin{Bmatrix} Q_1^1 \\ Q_2^1 \end{Bmatrix} = \begin{Bmatrix} -10.9 \\ 10.9 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^2 \\ Q_2^2 \end{Bmatrix} = \begin{Bmatrix} -7.62 \\ 25.6 \end{Bmatrix} kN$$

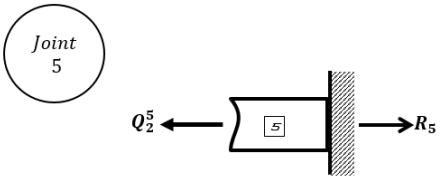
$$\begin{Bmatrix} Q_1^3 \\ Q_2^3 \end{Bmatrix} = \begin{Bmatrix} -36.5 \\ 36.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^4 \\ Q_2^4 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -11.5 \end{Bmatrix} kN$$

$$\begin{Bmatrix} Q_1^5 \\ Q_2^5 \end{Bmatrix} = \begin{Bmatrix} 11.5 \\ -59.5 \end{Bmatrix} kN$$



$$R_4 = Q_1^1 + Q_1^2 = -18.5\ kN$$



$$R_5 = Q_2^5 = -59.5\ kN$$

Appendix

Problem 1

PYTHON INPUT

```
'''
HW5_Prob1_Python
units: kN and mm
'''
######## IMPORT MODULES ########

import numpy as np # import matrix operators

solve = np.linalg.solve             # set up a shorthand for factorization

######## DEFINE FUNCTIONS ########
###
def Create_k(E,A,L):
   k = (A*E/L)*np.array([[+1,-1],
                         [-1,+1]])
   return k
###
def Assemble_S(S,k,MemNum,NEdof,NSdof,CodeNum):
   for KRow in range(0,NEdof,1):
      SRow=CodeNum[MemNum-1,KRow]
      if SRow <= NSdof:
         for KCol in range(0,NEdof,1):
            SCol=CodeNum[MemNum-1,KCol]
            if SCol <= NSdof:
               S[SRow-1,SCol-1]=S[SRow-1,SCol-1]+k[KRow,KCol]

   return S
###

######## DEFINE SYSTEM FOR ANALYSIS ########

# Define the number of element dofs (always 2 for the uniaxial element)
NEdof=2

# Define the number of Structural dofs
NSdof=3

# Define {P} vector
P = np.array([[-12],
              [48],
              [24]])
print 'P = '
print P
print

# Define the code numbers
CodeNum=np.array([[4,1],
                  [4,1],
                  [1,2],
                  [2,3],
                  [3,5]]) # Here we write out each row explicitly
print 'CodeNum = '
print CodeNum
print
```

```
# Initialize S
S = np.zeros((3,3),dtype=float)

# Define geometry and loading for each member. Create the member stiffness
# and then assemble [S]. Note that no units are shown, you just need to be
# be consistent.

# Modulus of Elasticity
E = 20

MemNum=1
A=60
L=1.5e3
k1 = Create_k(E,A,L)
S = Assemble_S(S,k1,MemNum,NEdof,NSdof,CodeNum)

MemNum=2
A=75
L=1.5e3
k2 = Create_k(E,A,L)
S = Assemble_S(S,k2,MemNum,NEdof,NSdof,CodeNum)

MemNum=3
A=150
L=0.75e3
k3 = Create_k(E,A,L)
S = Assemble_S(S,k3,MemNum,NEdof,NSdof,CodeNum)

MemNum=4
A=90
L=1.25e3
k4 = Create_k(E,A,L)
S = Assemble_S(S,k4,MemNum,NEdof,NSdof,CodeNum)

MemNum=5
A=120
L=1e3
k5 = Create_k(E,A,L)
S = Assemble_S(S,k5,MemNum,NEdof,NSdof,CodeNum)

print 'S ='
print S
print

######## Solve ########

d = solve(S,P)

print 'd ='
print d
print

######## POST-PROCESS ########

# Use compatibility to create the {u} vectors for each element
u1 = np.array([[0],
               d[0]])
```

```
u2 = np.array([[0],
               d[0]])
u3 = np.array([d[0],
               d[1]])
u4 = np.array([d[1],
               d[2]])
u5 = np.array([d[2],
               [0]])
# Calculate the member end forces
Q1 = k1.dot(u1)
Q2 = k2.dot(u2)
Q3 = k3.dot(u3)
Q4 = k4.dot(u4)
Q5 = k5.dot(u5)


print 'Q1 ='
print Q1
print
print 'Q2 ='
print Q2
print
print 'Q3 ='
print  Q3
print
print 'Q4 ='
print Q4
print
print 'Q5 ='
print Q5
print

# Calculate the support reactions
R4 = Q1[0]+Q2[0]
R5 = Q5[1]

print 'R4 =', R4[0]
print
print 'R5 =', R5[0]
```

PYTHON OUTPUT

```
P =
[[-12]
 [ 48]
 [ 24]]

CodeNum =
[[4 1]
 [4 1]
 [1 2]
 [2 3]
 [3 5]]

S =
[[ 5.8  -4.    0.  ]
 [-4.    5.44 -1.44]
 [ 0.   -1.44  3.84]]

d =
[[ 13.62318841]
 [ 22.75362319]
 [ 14.7826087 ]]

Q1 =
[[-10.89855072]
 [ 10.89855072]]

Q2 =
[[-13.62318841]
 [ 13.62318841]]

Q3 =
[[-36.52173913]
 [ 36.52173913]]

Q4 =
[[ 11.47826087]
 [-11.47826087]]

Q5 =
[[ 35.47826087]
 [-35.47826087]]

R4 = -24.5217391304

R5 = -35.4782608696
```

SAP RESULTS

TABLE: Element Forces - Frames

Frame	Station	OutputCase	CaseType	P	V2	V3	T	M2	M3	FrameElem	ElemStation
Text	mm	Text	Text	KN	KN	KN	KN-mm	KN-mm	KN-mm	Text	mm
1	0	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	0
1	500	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	500
1	1000	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	1000
1	1500	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	1500
2	0	DEAD	LinStatic	13.623	0	0	0	0	0	2-1	0
2	500	DEAD	LinStatic	13.623	0	0	0	0	0	2-1	500
2	1000	DEAD	LinStatic	13.623	0	0	0	0	0	2-1	1000
2	1500	DEAD	LinStatic	13.623	0	0	0	0	0	2-1	1500
3	0	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	0
3	375	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	375
3	750	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	750
4	0	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	0
4	416.67	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	416.67
4	833.33	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	833.33
4	1250	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	1250
5	0	DEAD	LinStatic	-35.478	0	0	0	0	0	5-1	0
5	500	DEAD	LinStatic	-35.478	0	0	0	0	0	5-1	500
5	1000	DEAD	LinStatic	-35.478	0	0	0	0	0	5-1	1000

TABLE: Joint Displacements

Joint	OutputCase	CaseType	U1	U2	U3	R1	R2	R3
Text	Text	Text	mm	mm	mm	Radians	Radians	Radians
1	DEAD	LinStatic	0	0	0	0	0	0
2	DEAD	LinStatic	13.623188	0	0	0	0	0
3	DEAD	LinStatic	0	0	0	0	0	0
4	DEAD	LinStatic	13.623188	0	0	0	0	0
5	DEAD	LinStatic	22.753623	0	0	0	0	0
6	DEAD	LinStatic	14.782609	0	0	0	0	0
7	DEAD	LinStatic	0	0	0	0	0	0

TABLE: Joint Reactions

Joint	OutputCase	CaseType	F1	F2	F3	M1	M2	M3
Text	Text	Text	KN	KN	KN	KN-mm	KN-mm	KN-mm
3	DEAD	LinStatic	-24.522	0	0	0	-10898.55	0
7	DEAD	LinStatic	-35.478	0	0	0	0	0

Problem 2

PYTHON INPUT

```
'''
HW5_Prob2_Python
units: kN and mm
'''
######## IMPORT MODULES ########

import numpy as np # import matrix operators

solve = np.linalg.solve             # set up a shorthand for factorization

######## DEFINE FUNCTIONS ########
###
def Create_k(E,A,L):
   k = (E*A/L)*np.array([[+1,-1],
                         [-1,+1]])
   return k
###
def Assemble_S_Pf(S,Pf,k,Qf,MemNum,NEdof,NSdof,CodeNum):
   for KRow in range(0,NEdof,1):
      SRow=CodeNum[MemNum-1,KRow]
      if SRow <= NSdof:
         Pf[SRow-1]=Pf[SRow-1]+Qf[KRow]
         for KCol in range(0,NEdof,1):
            SCol=CodeNum[MemNum-1,KCol]
            if SCol <= NSdof:
               S[SRow-1,SCol-1]=S[SRow-1,SCol-1]+k[KRow,KCol]
   return S, Pf
###

######## DEFINE SYSTEM FOR ANALYSIS ########

# Define the number of element dofs (always 2 for the uniaxial element)
NEdof=2

# Define the number of Structural dofs
NSdof=3

# Define {P} vector
P = np.array([[0],
              [48],
              [0]])
print 'P = '
print P
print
# Define the code numbers
CodeNum=np.array([[4,1],
                  [4,1],
                  [1,2],
                  [2,3],
                  [3,5]]) # Here we write out each row explicitly
print 'CodeNum = '
print CodeNum
print

# Initialize S,Pf
```

```
S = np.zeros((NSdof,NSdof),dtype=float)
Pf = np.zeros((NSdof,1),dtype=float)

# Define geometry and loading for each member. Create the member stiffness
# and then assemble [S]. Note that no units are shown, you just need to be
# be consistent.

# Modulus of Elasticity
E = 20

MemNum=1
A=60
L=1.5e3
Qf1 = np.zeros((NEdof,1),dtype=float)
k1 = Create_k(E,A,L)
S, Pf = Assemble_S_Pf(S,Pf,k1,Qf1,MemNum,NEdof,NSdof,CodeNum)

MemNum=2
A=75
L=1.5e3
a = 1e3
b = 0.5e3
W = -18
Qf2 = W*np.array([[-b/L],
                  [-a/L]])
print 'Qf2 = '
print Qf2
print
k2 = Create_k(E,A,L)
S, Pf = Assemble_S_Pf(S,Pf,k2,Qf2,MemNum,NEdof,NSdof,CodeNum)

MemNum=3
A=150
L=0.75e3
Qf3 = np.zeros((NEdof,1),dtype=float)
k3 = Create_k(E,A,L)
S, Pf = Assemble_S_Pf(S,Pf,k3,Qf3,MemNum,NEdof,NSdof,CodeNum)

MemNum=4
A=90
L=1.25e3
Qf4 = np.zeros((NEdof,1),dtype=float)
k4 = Create_k(E,A,L)
S, Pf = Assemble_S_Pf(S,Pf,k4,Qf4,MemNum,NEdof,NSdof,CodeNum)

MemNum=5
A=120
L=1.00e3
w=48e-3
Qf5 = w*L/2*np.array([[-1],
                      [-1]])
print 'Qf5 = '
print Qf5
print
k5 = Create_k(E,A,L)
S, Pf = Assemble_S_Pf(S,Pf,k5,Qf5,MemNum,NEdof,NSdof,CodeNum)
print 'Pf = '
print Pf
```

```
print

print 'S ='
print S
print

######## Solve ########

d = solve(S,(P-Pf))
print 'd ='
print d
print

######## POST-PROCESS ########
# Use compatibility to create the {u} vectors for each element
u1 = np.array([[0],
          d[0]])
u2 = np.array([[0],
          d[0]])
u3 = np.array([d[0],
          d[1]])
u4 = np.array([d[1],
          d[2]])
u5 = np.array([d[2],
           [0]])

# Calculate the member end forces
Q1 = Qf1 + k1.dot(u1)
Q2 = Qf2 + k2.dot(u2)
Q3 = Qf3 + k3.dot(u3)
Q4 = Qf4 + k4.dot(u4)
Q5 = Qf5 + k5.dot(u5)


print 'Q1 ='
print Q1
print
print 'Q2 ='
print Q2
print
print 'Q3 ='
print  Q3
print
print 'Q4 ='
print Q4
print
print 'Q5 ='
print Q5
print


# Calculate the support reactions
R4 = Q1[0]+Q2[0]
R5 = Q5[1]

print 'R4 =', R4[0]
print
print 'R5 =', R5[0]
```

PYTHON OUTPUT

```
P =
[[ 0]
 [48]
 [ 0]]
CodeNum =
[[4 1]
 [4 1]
 [1 2]
 [2 3]
 [3 5]]

Qf2 =
[[  6.]
 [ 12.]]

Qf5 =
[[-24.]
 [-24.]]

Pf =
[[ 12.]
 [  0.]
 [-24.]]

S =
[[ 5.8  -4.    0.  ]
 [-4.    5.44 -1.44]
 [ 0.   -1.44  3.84]]

d =
[[ 13.62318841]
 [ 22.75362319]
 [ 14.7826087 ]]

Q1 =
[[-10.89855072]
 [ 10.89855072]]

Q2 =
[[ -7.62318841]
 [ 25.62318841]]

Q3 =
[[-36.52173913]
 [ 36.52173913]]

Q4 =
[[ 11.47826087]
 [-11.47826087]]
Q5 =
[[ 11.47826087]
 [-59.47826087]]

R4 = -18.5217391304
R5 = -59.4782608696
```

SAP RESULTS

TABLE: Element Forces - Frames

Frame	Station	OutputCase	CaseType	P	V2	V3	T	M2	M3	FrameElem	ElemStation
Text	mm	Text	Text	KN	KN	KN	KN-mm	KN-mm	KN-mm	Text	mm
1	0	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	0
1	500	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	500
1	1000	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	1000
1	1500	DEAD	LinStatic	10.899	0	0	0	0	0	1-1	1500
2	0	DEAD	LinStatic	7.623	0	0	0	0	0	2-1	0
2	500	DEAD	LinStatic	7.623	0	0	0	0	0	2-1	500
2	1000	DEAD	LinStatic	7.623	0	0	0	0	0	2-1	1000
2	1000	DEAD	LinStatic	25.623	0	0	0	0	0	2-1	1000
2	1500	DEAD	LinStatic	25.623	0	0	0	0	0	2-1	1500
3	0	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	0
3	375	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	375
3	750	DEAD	LinStatic	36.522	0	0	0	0	0	3-1	750
4	0	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	0
4	416.67	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	416.67
4	833.33	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	833.33
4	1250	DEAD	LinStatic	-11.478	0	0	0	0	0	4-1	1250
5	0	DEAD	LinStatic	-11.478	0	0	0	0	0	5-1	0
5	500	DEAD	LinStatic	-35.478	0	0	0	0	0	5-1	500
5	1000	DEAD	LinStatic	-59.478	0	0	0	0	0	5-1	1000

TABLE: Joint Displacements

Joint	OutputCase	CaseType	U1	U2	U3	R1	R2	R3
Text	Text	Text	mm	mm	mm	Radians	Radians	Radians
1	DEAD	LinStatic	0	0	0	0	0	0
2	DEAD	LinStatic	13.623188	0	0	0	0	0
3	DEAD	LinStatic	0	0	0	0	0	0
4	DEAD	LinStatic	13.623188	0	0	0	0	0
5	DEAD	LinStatic	22.753623	0	0	0	0	0
6	DEAD	LinStatic	14.782609	0	0	0	0	0
7	DEAD	LinStatic	0	0	0	0	0	0

TABLE: Joint Reactions

Joint	OutputCase	CaseType	F1	F2	F3	M1	M2	M3
Text	Text	Text	KN	KN	KN	KN-mm	KN-mm	KN-mm
3	DEAD	LinStatic	-18.522	0	0	0	-10898.55	0
7	DEAD	LinStatic	-59.478	0	0	0	0	0