SOLUTION

CE 325 Spring 2026 HW#8

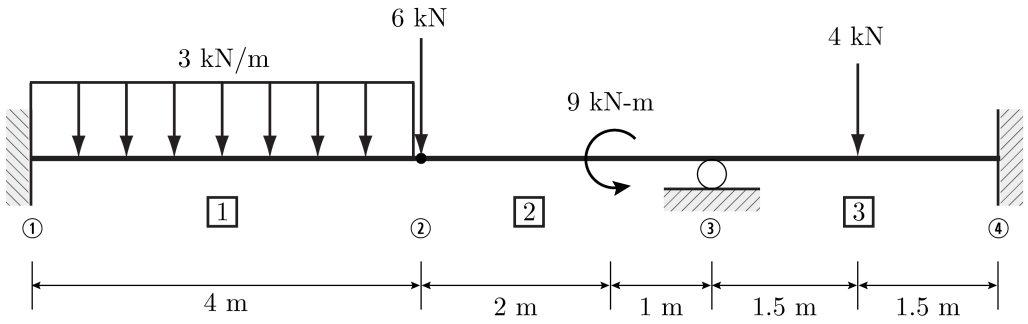
Due Friday, April 17, by 5:00pm ET

1. For the beam shown below determine joint displacements/rotations, support reactions, and shear force and bending moment diagrams, using the Matrix Displacement Method:

a. (10 pts) By hand

b. (10 pts) Using Python

c. (10 pts) Using SAP2000



E = 20 GPa; I = 6.75e7 $mm.^4$ for all members

Problem 1

Executive Summary

Problem Statement

For the beam shown below, determine joint displacements/rotations, support reactions, and shear force and bending moment diagrams, using the Matrix Displacement Method:

a. By hand
b. Using Python
c. Using SAP2000

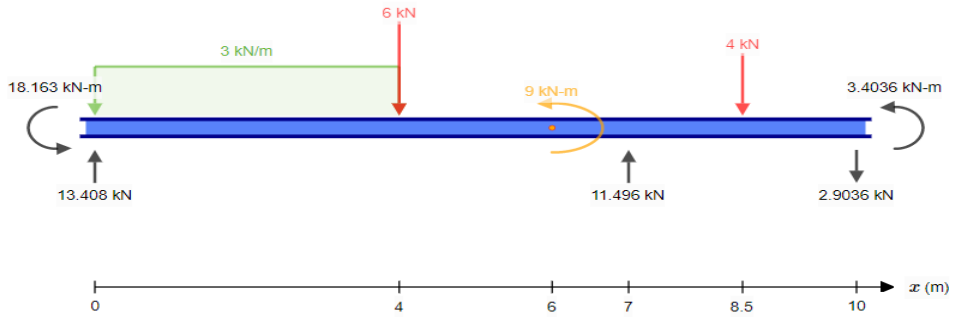
Results

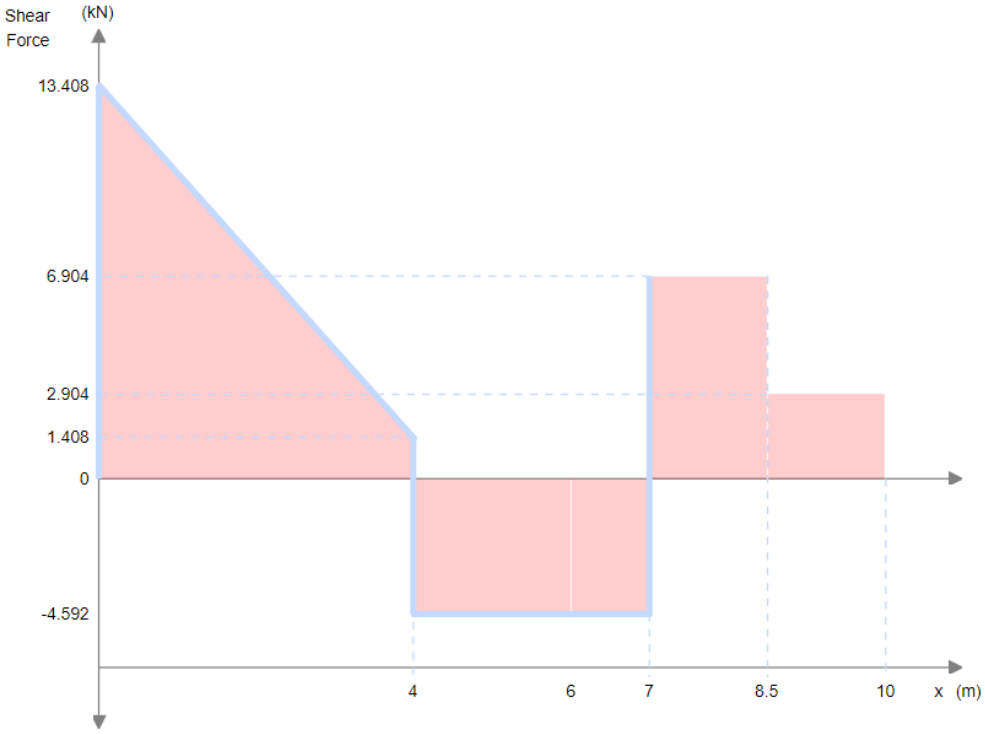
$$\{d\} = \begin{Bmatrix} d_1 \\ d_2 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} -25.4 \\ 0.00194 \\ 0.00545 \end{Bmatrix} \begin{matrix} mm \\ rad \\ rad \end{matrix}$$

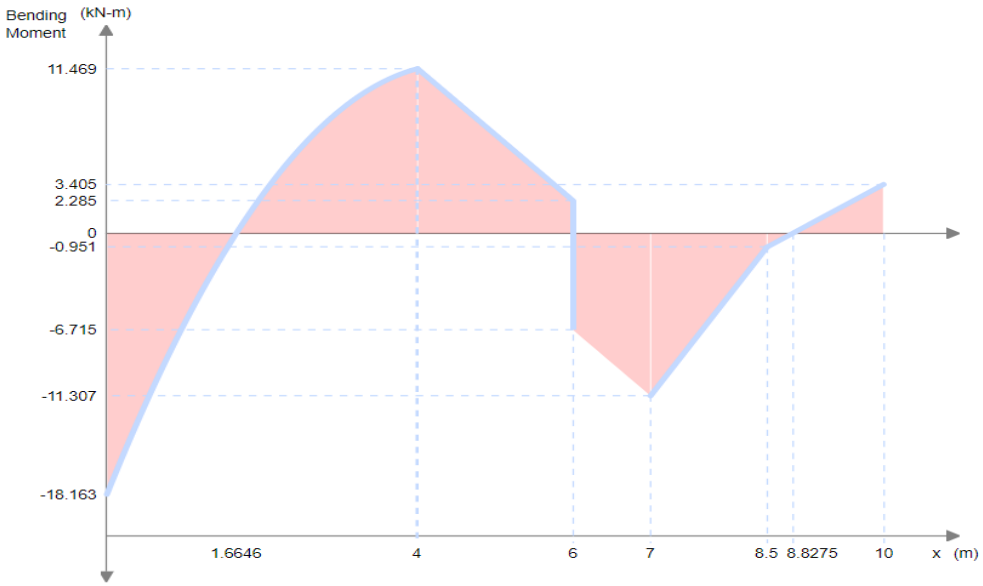
$$R4 = 13.4\ kN$$
$$R5 = 18.2\ kN * m$$
$$R6 = 11.5\ kN$$
$$R7 = -2.9\ kN$$
$$R8 = 3.40\ kN * m$$

Shear and Moment Diagram on next page ->

Shear and Moment Diagram





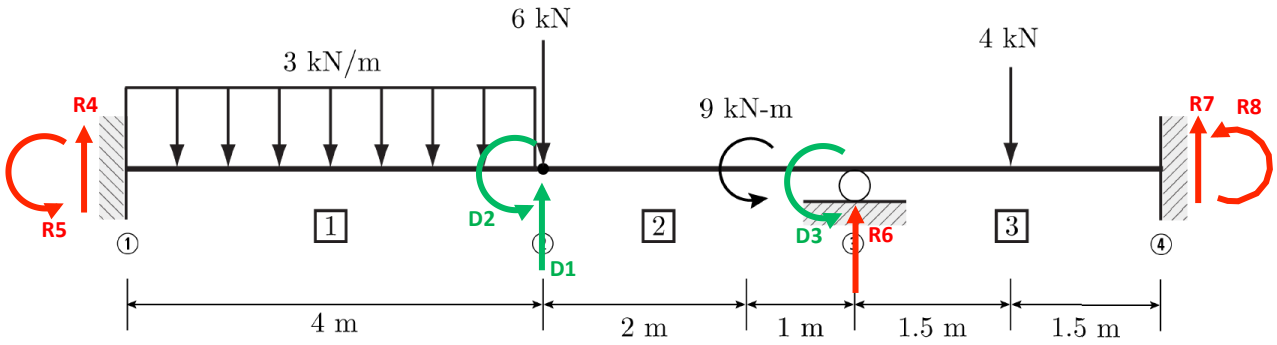


Problem 1

Technical Summary

a. By hand

1. Label structure



2. Create {P}

$$P = \begin{Bmatrix} P_1 \\ P_2 \\ P_3 \end{Bmatrix} = \begin{Bmatrix} -6 \\ 0 \\ 0 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN * mm \end{matrix}$$

3. Code Numbers

Member #	Code Number			
Member 1	4	5	1	2
Member 2	1	2	6	3
Member 3	6	3	7	8

4. Determine {Qf}, [k]

$$\{Q_f\}^1 = \begin{Bmatrix} \frac{wL}{2} \\ \frac{wL^2}{12} \\ \frac{wL}{2} \\ -\frac{wL^2}{12} \end{Bmatrix} = \begin{Bmatrix} \frac{(0.003)(4000)}{2} \\ \frac{(0.003)(4000)^2}{12} \\ \frac{(0.003)(4000)}{2} \\ -\frac{(0.003)(4000)^2}{12} \end{Bmatrix} = \begin{Bmatrix} 6 \\ 4000 \\ 6 \\ -4000 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN \\ kN * mm \end{matrix}$$

$$\{Q_f\}^2 = \begin{Bmatrix} -\frac{6Mab}{L^3} \\ \frac{Mb(b-2a)}{L^2} \\ \frac{6Mab}{L^3} \\ \frac{Ma(a-2b)}{L^2} \end{Bmatrix} = \begin{Bmatrix} -\frac{6(-9000)(2000)(1000)}{3000^3} \\ \frac{(-9000)(1000)(-3000)}{3000^2} \\ \frac{6(-9000)(2000)(1000)}{3000^3} \\ \frac{(-9000)(2000)(0)}{3000^2} \end{Bmatrix} = \begin{Bmatrix} 4 \\ 3000 \\ -4 \\ 0 \end{Bmatrix} \begin{matrix} kN \\ kN*mm \\ kN \\ kN*mm \end{matrix}$$

$$\{Q_f\}^3 = \begin{Bmatrix} \frac{W}{2} \\ \frac{WL}{8} \\ \frac{W}{2} \\ -\frac{WL}{8} \end{Bmatrix} = \begin{Bmatrix} \frac{4}{2} \\ \frac{4(3000)}{8} \\ \frac{4}{2} \\ -\frac{4(3000)}{8} \end{Bmatrix} = \begin{Bmatrix} 2 \\ 1500 \\ 2 \\ -1500 \end{Bmatrix} \begin{matrix} kN \\ kN*mm \\ kN \\ kN*mm \end{matrix}$$

$$[k] = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

Member 1:

$L = 4000\ mm$

$EI = 1.35 * 10^9\ kN * mm^2$

$\frac{12EI}{L^3} = 0.253\ kN/mm \quad \frac{4EI}{L} = 1.35e6\ kN * mm$

$\frac{6EI}{L^2} = 506.25\ kN \quad \frac{2EI}{L} = 6.75e5\ kN * mm$

Members 2, 3:

$L = 3000\ mm$

$EI = 1.35 * 10^9\ kN * mm^2$

$\frac{12EI}{L^3} = 0.600\ kN/mm \quad \frac{4EI}{L} = 1.80e6\ kN * mm$

$\frac{6EI}{L^2} = 900\ kN \quad \frac{2EI}{L} = 9.00e5\ kN * mm$

5. Assemble {Pf} and [S] using Code # Method

$$\{P_f\} = \begin{Bmatrix} {Q_f}_3^1 + {Q_f}_1^2 \\ {Q_f}_4^1 + {Q_f}_2^2 \\ {Q_f}_4^2 + {Q_f}_2^3 \end{Bmatrix} = \begin{Bmatrix} 10 \\ -1000 \\ 1500 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN * mm \end{matrix}$$

$$S = \begin{bmatrix} k_{33}^1 + k_{11}^2 & k_{34}^1 + k_{12}^2 & k_{14}^2 \\ k_{43}^1 + k_{21}^2 & k_{44}^1 + k_{22}^2 & k_{24}^2 \\ k_{41}^2 & k_{42}^2 & k_{44}^2 + k_{22}^3 \end{bmatrix} = \begin{bmatrix} 0.853 & 393.8 & 900 \\ 393.8 & 3.15e6 & 9e5 \\ 900 & 9e5 & 3.6e6 \end{bmatrix}$$

6. Solve {P - P_f} = [S]{d}

$$\{d\} = [S]^{-1}\{P - P_f\}$$

$$\{\boldsymbol{d}\} = \begin{Bmatrix} \boldsymbol{d_1} \\ \boldsymbol{d_2} \\ \boldsymbol{d_3} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{-25.4} \\ \boldsymbol{0.00194} \\ \boldsymbol{0.00545} \end{Bmatrix} \begin{matrix} \boldsymbol{mm} \\ \boldsymbol{rad} \\ \boldsymbol{rad} \end{matrix}$$

7. Compatibility

$$\{u\}^1 = \begin{Bmatrix} 0 \\ 0 \\ d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -25.4 \\ 0.00545 \end{Bmatrix} \begin{matrix} mm \\ rad \\ mm \\ rad \end{matrix}$$

$$\{u\}^2 = \begin{Bmatrix} d_1 \\ d_2 \\ 0 \\ d_3 \end{Bmatrix} = \begin{Bmatrix} -25.4 \\ 0.00194 \\ 0 \\ 0.00545 \end{Bmatrix} \begin{matrix} mm \\ rad \\ mm \\ rad \end{matrix}$$

$$\{u\}^3 = \begin{Bmatrix} 0 \\ d_3 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0.00545 \\ 0 \\ 0 \end{Bmatrix} \begin{matrix} mm \\ rad \\ mm \\ rad \end{matrix}$$

8. Solve {Q}

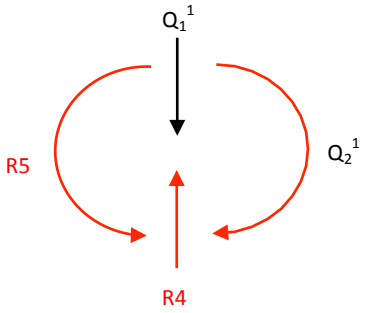
$$\{Q\} = \{Q_f\} + [k]\{u\}$$

$$\{Q\}^1 = \begin{Bmatrix} 13.4 \\ 18{,}163 \\ -1.41 \\ 11{,}469 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN \\ kN * mm \end{matrix}$$

$$\{Q\}^2 = \begin{Bmatrix} -4.59 \\ -11{,}469 \\ 4.59 \\ -11{,}307 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN \\ kN * mm \end{matrix}$$

$$\{Q\}^3 = \begin{Bmatrix} 6.90 \\ -11{,}307 \\ -2.90 \\ 3{,}404 \end{Bmatrix} \begin{matrix} kN \\ kN * mm \\ kN \\ kN * mm \end{matrix}$$

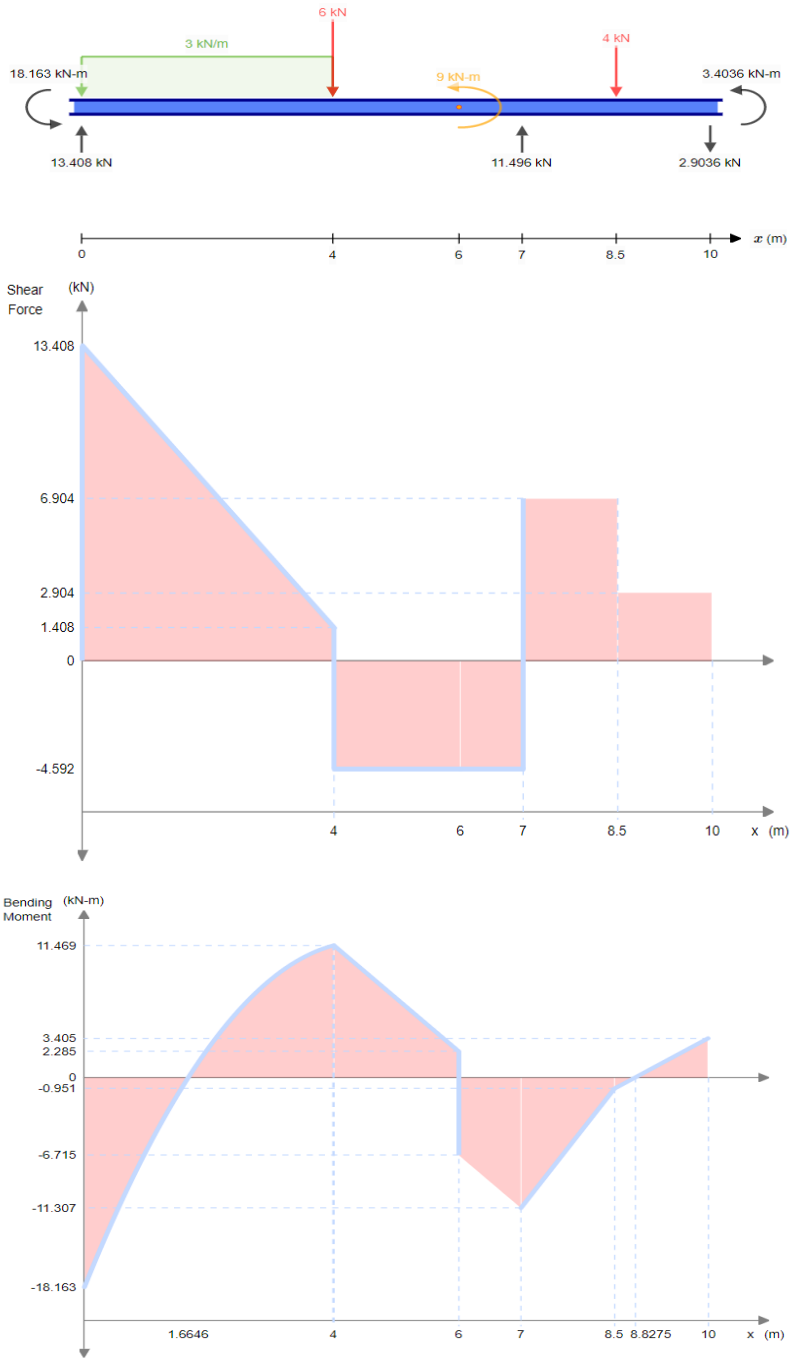
9. Determine Support Reactions from Equilibrium



$$\Sigma F_y = 0, \quad \Sigma M = 0$$

$$\boldsymbol{R4 = Q_1^1 = 13.4\ kN}$$
$$\boldsymbol{R5 = Q_2^1 = 18.2\ kN * m}$$
$$\boldsymbol{R6 = Q_3^2 + Q_1^3 = 11.5\ kN}$$
$$\boldsymbol{R7 = Q_3^3 = -2.9\ kN}$$
$$\boldsymbol{R8 = Q_4^3 = 3.40\ kN * m}$$

10. Shear and Moment Diagram



Appendix

PYTHON CODE INPUT

```
from __future__ import division #fixes floating point and integer division
'''
HW9 Python Code
Units: kN and mm
'''
######## IMPORT MODULES ########
import numpy as np # import matrix operators
float_formatter = lambda x: "%.5f" % x
np.set_printoptions(formatter={'float_kind':float_formatter})

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


######## DEFINE FUNCTIONS ########
###
def Create_k(E,I,L):
   k = (E*I/L**3)*np.array([[12    ,6*L   ,-12   ,6*L   ],
                    [6*L   ,4*L**2,-6*L  ,2*L**2],
                    [-12   ,-6*L  ,12    ,-6*L  ],
                    [6*L   ,2*L**2,-6*L  ,4*L**2]])
   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=4

# Define the number of Structural dofs
NSdof=3

# Define {P} vector
P = np.array([[-6],
          [0],
          [0]])
print 'P ='
print P
print

# Define the code numbers
CodeNum=np.array([[4,5,1,2],
```

```
            [1,2,6,3],
            [6,3,7,8]]) # Here we write out each row explicitly
print 'CodeNum ='
print CodeNum
print

# Initialize S
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]. Be consistent with units.

# Modulus of Elasticity
E = 20


MemNum=1
I = 6.75e7
L = 4e3
w = 3/1000
Qf1 = np.array([[w*L/2],
            [w*L**2/12],
            [w*L/2],
            [-w*L**2/12]])
k1 = Create_k(E,I,L)
S, Pf = Assemble_S_Pf(S,Pf,k1,Qf1,MemNum,NEdof,NSdof,CodeNum)
print 'Qf1 ='
print Qf1
print
print 'k1 ='
print k1
print

MemNum=2
I = 6.75e7
L = 3e3
a = 2e3
b = 1e3
M = -9e3
Qf2 = np.array([[-6*M*a*b/L**3],
            [M*b*(b-2*a)/L**2],
            [6*M*a*b/L**3],
            [M*a*(a-2*b)/L**2]])
k2 = Create_k(E,I,L)
S, Pf = Assemble_S_Pf(S,Pf,k2,Qf2,MemNum,NEdof,NSdof,CodeNum)
print 'Qf2 ='
print Qf2
print
print 'k2 ='
print k2
print
```

```
MemNum=3
I = 6.75e7
L = 3e3
W = 4
Qf3 = np.array([[W/2],
            [W*L/8],
            [W/2],
            [-W*L/8]])
k3 = Create_k(E,I,L)
S, Pf = Assemble_S_Pf(S,Pf,k3,Qf3,MemNum,NEdof,NSdof,CodeNum)
print 'Qf3 ='
print Qf3
print
print 'k3 ='
print k3
print

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],
            [0],
            d[0],
            d[1]])   # Indexing starts with zero (0) in Python

u2 = np.array([d[0],
            d[1],
            [0],
            d[2]])

u3 = np.array([[0],
            d[2],
            [0],
            [0]])



# Calculate the member end forces
Q1 = Qf1 + k1.dot(u1)
Q2 = Qf2 + k2.dot(u2)
```

```
Q3 = Qf3 + k3.dot(u3)

print 'Q1 ='
print Q1
print
print 'Q2 ='
print Q2
print
print 'Q3 ='
print Q3
print

# Calculate the support reactions
R4 = Q1[0]
R5 = Q1[1]
R6 = Q2[2]+Q3[0]
R7 = Q3[2]
R8 = Q3[3]

print 'R4 =', R4
print 'R5 =', R5
print 'R6 =', R6
print 'R7 =', R7
print 'R8 =', R8
```

PYTHON CODE OUTPUT

```
P =
[[-6]
 [ 0]
 [ 0]]

CodeNum =
[[4 5 1 2]
 [1 2 6 3]
 [6 3 7 8]]

Qf1 =
[[6.00000]
 [4000.00000]
 [6.00000]
 [-4000.00000]]

k1 =
[[0.25313 506.25000 -0.25313 506.25000]
 [506.25000 1350000.00000 -506.25000 675000.00000]
 [-0.25313 -506.25000 0.25313 -506.25000]
 [506.25000 675000.00000 -506.25000 1350000.00000]]

Qf2 =
[[4.00000]
 [3000.00000]
 [-4.00000]
 [-0.00000]]

k2 =
[[0.60000 900.00000 -0.60000 900.00000]
 [900.00000 1800000.00000 -900.00000 900000.00000]
 [-0.60000 -900.00000 0.60000 -900.00000]
 [900.00000 900000.00000 -900.00000 1800000.00000]]

Qf3 =
[[2.00000]
 [1500.00000]
 [2.00000]
 [-1500.00000]]

k3 =
[[0.60000 900.00000 -0.60000 900.00000]
 [900.00000 1800000.00000 -900.00000 900000.00000]
 [-0.60000 -900.00000 0.60000 -900.00000]
 [900.00000 900000.00000 -900.00000 1800000.00000]]
```

```
Pf=
[[10.00000]
 [-1000.00000]
 [1500.00000]]

S =
[[0.85313 393.75000 900.00000]
 [393.75000 3150000.00000 900000.00000]
 [900.00000 900000.00000 3600000.00000]]

d =
[[-25.39553]
 [0.00194]
 [0.00545]]

Q1 =
[[13.40794]
 [18162.75510]
 [-1.40794]
 [11469.02332]]

Q2 =
[[-4.59206]
 [-11469.02332]
 [4.59206]
 [-11307.14286]]

Q3 =
[[6.90357]
 [11307.14286]
 [-2.90357]
 [3403.57143]]

R4 = [13.40794]
R5 = [18162.75510]
R6 = [11.49563]
R7 = [-2.90357]
R8 = [3403.57143]
```

SAP OUTPUT

Deflected shape

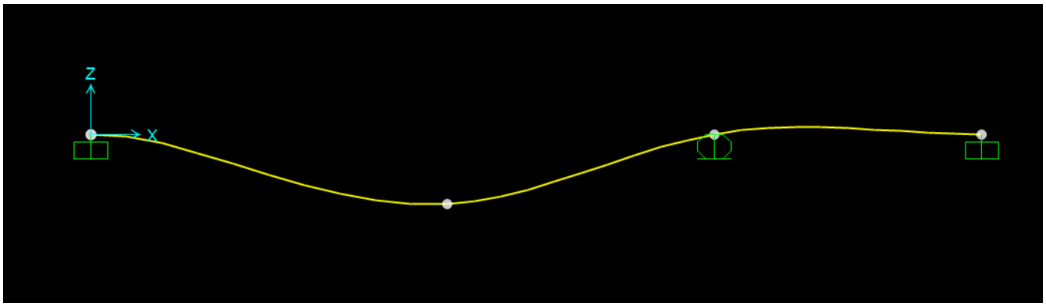
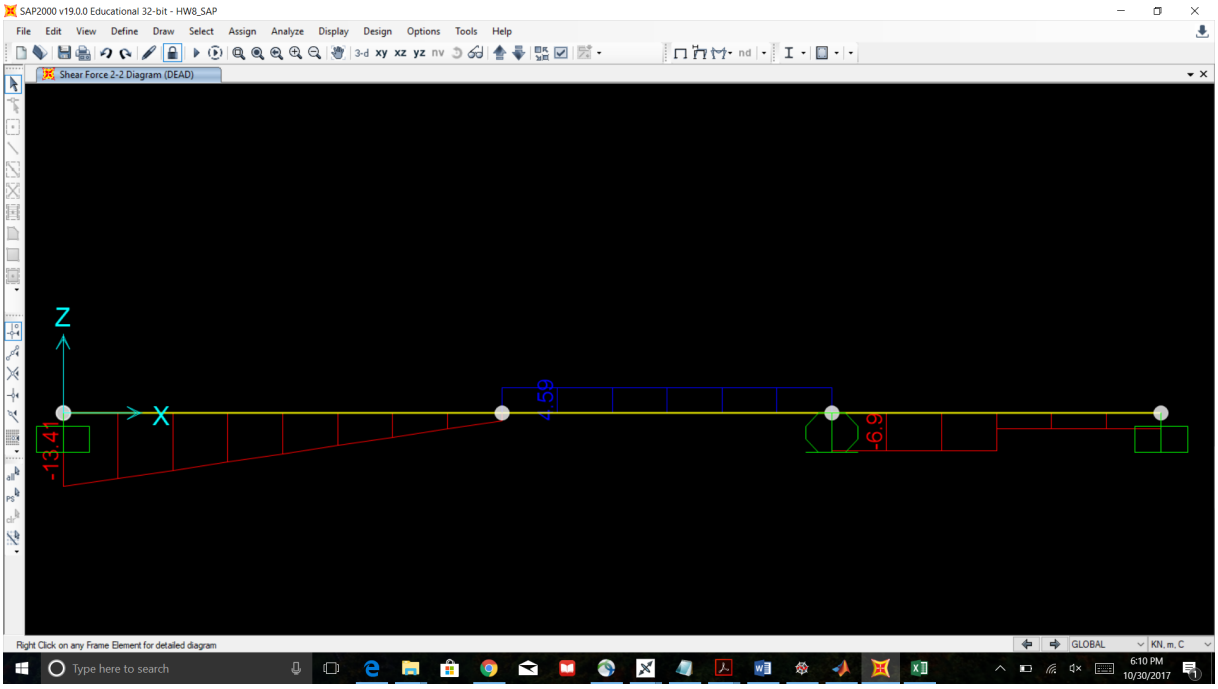


TABLE: Joint Displacements								
Joint	OutputCase	CaseType	U1	U2	U3	R1	R2	R3
Text	Text	Text	m	m	m	Radians	Radians	Radians
1	DEAD	LinStatic	0	0	0	0	0	0
2	DEAD	LinStatic	0	0	-0.025396	0	-0.001935	0
3	DEAD	LinStatic	0	0	0	0	-0.005448	0
4	DEAD	LinStatic	0	0	0	0	0	0

TABLE: Joint Reactions								
Joint	OutputCase	CaseType	F1	F2	F3	M1	M2	M3
Text	Text	Text	KN	KN	KN	KN-m	KN-m	KN-m
1	DEAD	LinStatic	0	0	13.408	0	-18.1628	0
3	DEAD	LinStatic	0	0	11.496	0	0	0
4	DEAD	LinStatic	0	0	-2.904	0	-3.4036	0

Shear force diagram



Bending moment diagram