

Solutions to Homework 2

Question 1: 10 points.

The cantilevered frame structure carries vertical loads P kN, $2P$ kN, $3P$ kN and $4P$ kN at nodes D, F, H and I. Use the **method of sections** to determine the forces in members a, b and c, as a function of the problem parameters L and P .

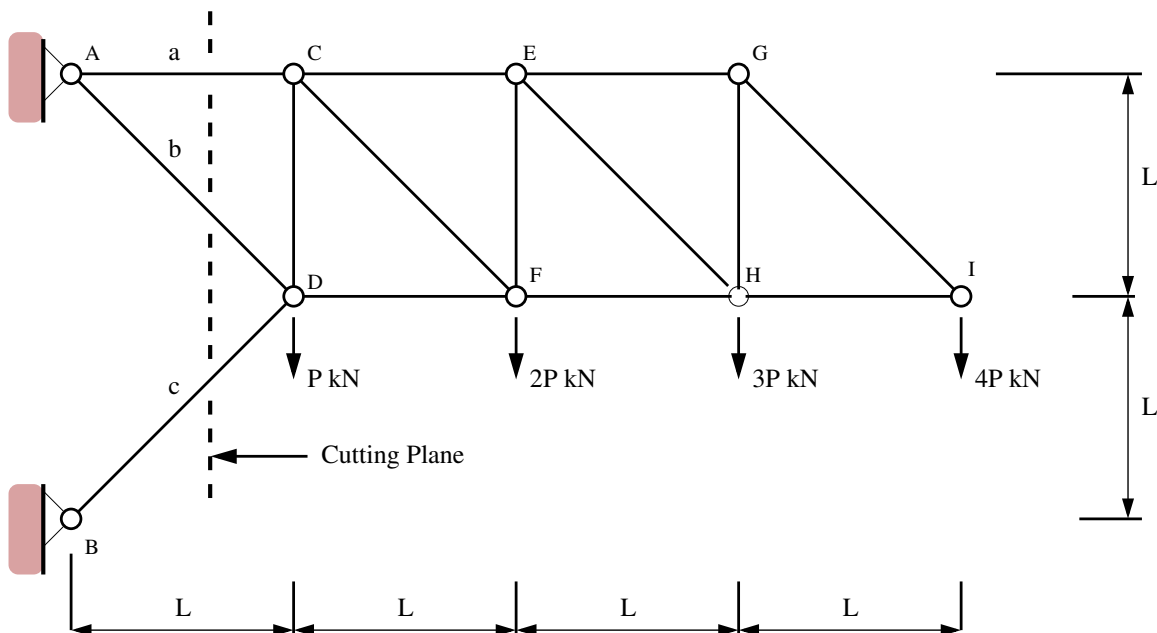


Figure 1: Cantilevered frame structure.

State if the members are in tension or compression. Show all of your working.

Solution: We begin by cutting the structure as shown in Figure 1. Taking moments about D for the right-hand substructure gives:

$$\sum M_D = 0 \quad 2PL + 6PL + 12PL = LF_{ac} \rightarrow F_{ac} = 20P (T). \quad (1)$$

Taking moments about A for the complete structure gives:

$$\sum M_A = 0 \quad PL + 4PL + 9PL + 16PL = 2LH_b \longrightarrow H_b = 15P. \quad (2)$$

Next, observe that the orientation of element B-D is $\pi/4$ radians. Hence, the horizontal and vertical components of reaction force at B need to be equal, i.e., $V_b = 15P$ and $F_{bd} = -15\sqrt{2}P$ (C). Looking at the equilibrium of the structure at A, the horizontal and vertical reaction forces are: $V_a = -5P$ and $H_a = -15P$. Summing forces in the vertical direction at A, member force $F_{ad} = -5\sqrt{2}P$ (C).

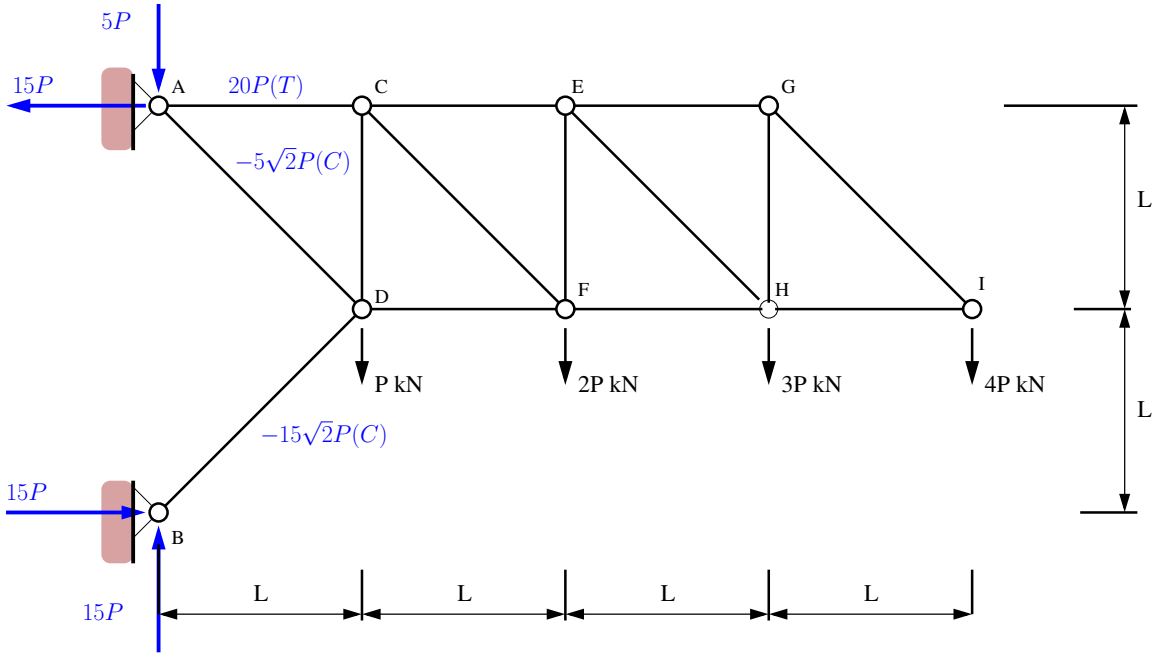


Figure 2: Critical forces in cantilevered frame structure.

Summary of Results: $F_{ac} = 20P$ (T), $F_{bd} = -15\sqrt{2}P$ (C), $F_{ad} = -5\sqrt{2}P$ (C).

Question 2: 10 points.

If the maximum tensile force that any member in Figure 1 can carry is 50 kN, and the maximum compressive force is 30 kN (before buckling), what is the maximum value of loads P that can be carried.

Soln's. Maximum tensile and compressive forces are $20P$ (kN) and $-15\sqrt{2}P$ (kN), respectively.

Tension constraint: $P_{max} = 50/20$ kN = 2.5 kN.

Compression constraint: $P_{max} = \frac{-30}{-15\sqrt{2}}$ kN = 1.41 kN.

Therefore, maximum allowable load is $P_{max} = 1.41$ kN.

Question 3: 10 points.

Figure 5 shows a multi-span beam. The cantilever is fully-fixed to the wall at Point A. Points B, D, E are hinges. Beam elements A-B and D-E have flexural stiffness EI . Beam elements B-C-D and E-F-G-H have flexural stiffness $2EI$. Vertical point loads $6P$ (kN) and $3P$ (kN) are applied mid-way between D and E, and at node H.

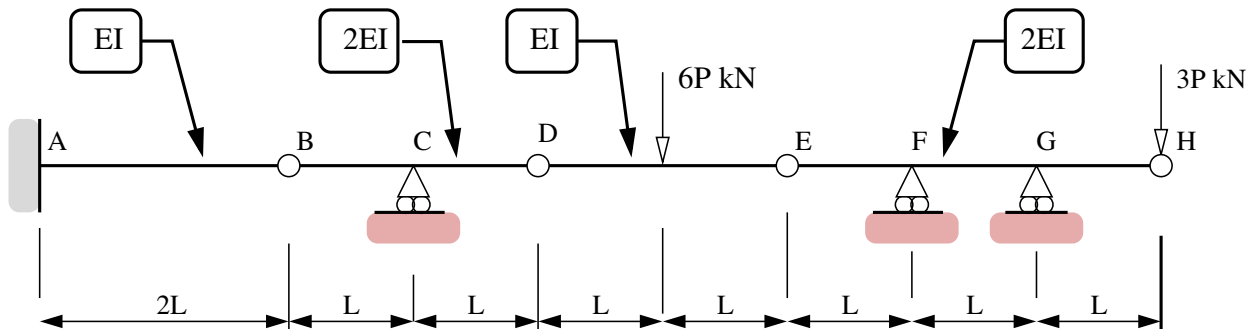
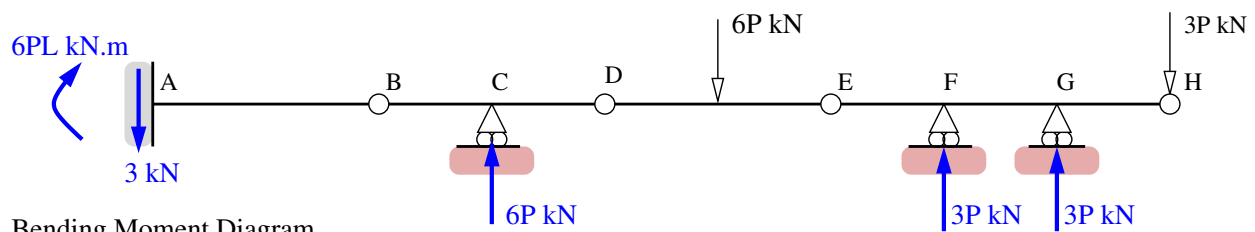


Figure 3: Three-span beam.

Draw and label the distributions of bending moment and curvature throughout the beam as a function of P and L . Qualitatively sketch the deflected shape of the beam.

Soln's. The support reactions are as follows:

Support Reactions



Bending Moment Diagram

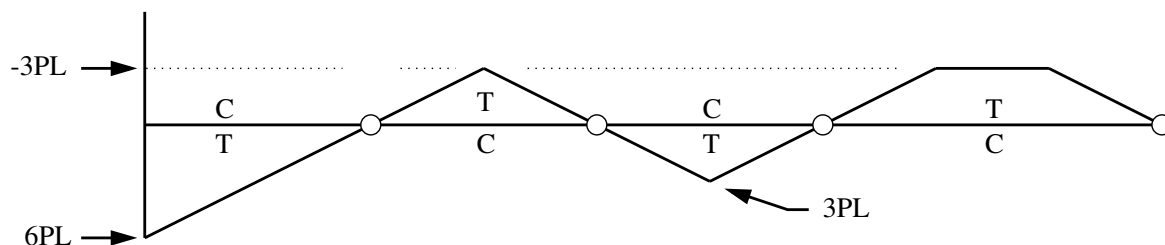
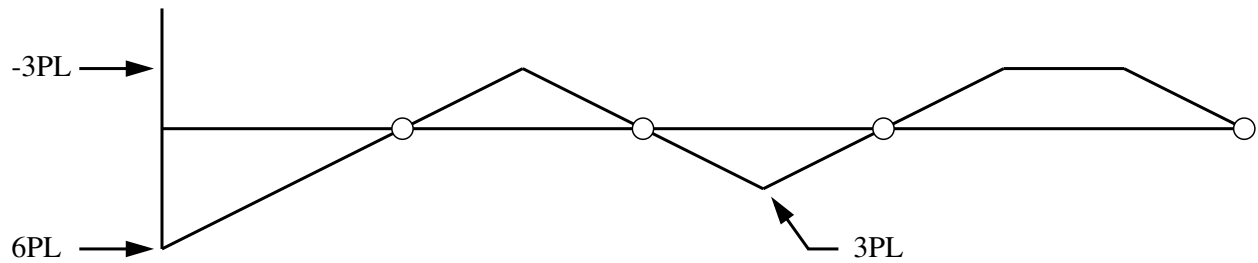


Figure 4: Three-span beam + support reactions + bending moments.

Note: Here we are drawing the bending moment on the tension side of the element.

The bending moment and curvature diagrams are as follows:

Bending Moment Diagram



Curvature Diagram

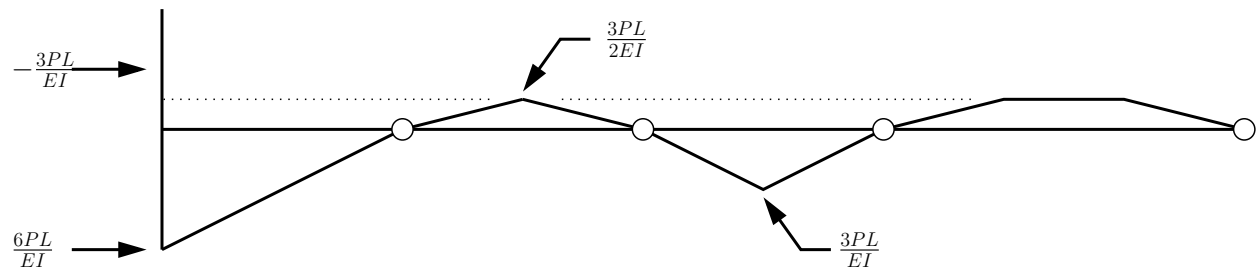


Figure 5: Bending moments + curvature for three-span beam.

Question 4: 20 points. In the design of large frame structures, engineers are often required to compute the maximum and minimum member forces and support reactions due to a variety of loading conditions.

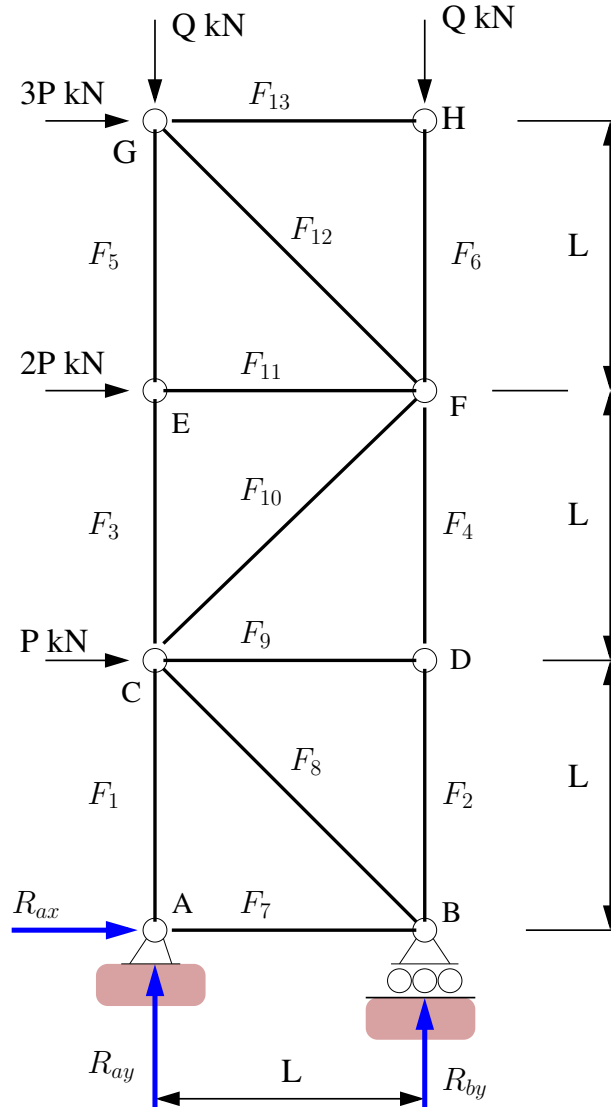


Figure 6: Front elevation of pin-jointed tower.

Figure 6 shows a thirteen bar pin-jointed truss that carries a variety of horizontal loads, P kN, and vertical loads Q kN. The symbols $F_1, F_2, \dots, F_{13}$ represent the axial forces in truss members 1 through 13, and R_{ax}, R_{ay} , and R_{by} are the horizontal and vertical support reactions at joints A and B.

Write down the equations of equilibrium for joints A through H and put the equations in matrix form. Now suppose that the frame supports a variety of payloads:

$$\begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} = \begin{bmatrix} 250 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} P_2 \\ Q_2 \end{bmatrix} = \begin{bmatrix} 200 \\ 200 \end{bmatrix}, \quad \begin{bmatrix} P_3 \\ Q_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 500 \end{bmatrix} kN. \quad (3)$$

Develop a Python program that will solve the matrix equations for each of the external load conditions, and compute and print the minimum and maximum support reactions at nodes A and B, and axial forces in each of the truss members.

Equations of Equilibrium:

At Joint A:

$$\sum F_x = 0, \quad R_{ax} + F_7 = 0. \quad (4)$$

$$\sum F_y = 0, \quad R_{ay} + F_1 = 0. \quad (5)$$

At Joint B:

$$\sum F_x = 0, \quad F_7 + \frac{1}{\sqrt{2}}F_8 = 0. \quad (6)$$

$$\sum F_y = 0, \quad F_2 + \frac{1}{\sqrt{2}}F_9 + R_{by} = 0. \quad (7)$$

At Joint C:

$$\sum F_x = 0, \quad \frac{1}{\sqrt{2}}F_8 + F_9 + \frac{1}{\sqrt{2}}F_{10} + P = 0. \quad (8)$$

$$\sum F_y = 0, \quad F_1 - F_3 + \frac{1}{\sqrt{2}}F_8 - \frac{1}{\sqrt{2}}F_{10} = 0. \quad (9)$$

At Joint D:

$$\sum F_x = 0, \quad F_9 = 0. \quad (10)$$

$$\sum F_y = 0, \quad F_2 - F_4 = 0. \quad (11)$$

At Joint E:

$$\sum F_x = 0, \quad F_{11} + 2P = 0. \quad (12)$$

$$\sum F_y = 0, \quad F_5 - F_3 = 0. \quad (13)$$

At Joint F:

$$\sum F_x = 0, \quad \frac{1}{\sqrt{2}}F_{10} + F_{11} + \frac{1}{\sqrt{2}}F_{12} = 0. \quad (14)$$

$$\sum F_y = 0, \quad F_6 - F_4 - \frac{1}{\sqrt{2}}F_{10} + \frac{1}{\sqrt{2}}F_{12} = 0. \quad (15)$$

At Joint G:

$$\sum F_x = 0, \quad \frac{1}{\sqrt{2}}F_{12} + F_{13} + 3P = 0. \quad (16)$$

$$\sum F_y = 0, \quad F_5 + \frac{1}{\sqrt{2}}F_{12} + Q = 0. \quad (17)$$

At Joint H:

$$\sum F_x = 0, \quad F_{13} = 0. \quad (18)$$

$$\sum F_y = 0, \quad F_6 + Q = 0. \quad (19)$$

Python Source Code:

```
# =====
# TestTrussTowerAnalysis01.py: Compute distribution of element forces
# and support reactions in a 13-bar truss tower.
#
# Written by: Mark Austin                                October 2026
# =====

import math
import numpy as np
from numpy.linalg import matrix_rank

# =====
# Function to print one- and two-dimensional matrices ...
# =====
```

```

def PrintMatrix(name, matrix):
    NoColumns = 6;

    # Compute no of blocks of rows to be printed .....

    if matrix.ndim == 1:
        noMatrixRows = matrix.shape[0]
        noMatrixCols = 1

    if matrix.ndim == 2:
        noMatrixRows = matrix.shape[0]
        noMatrixCols = matrix.shape[1]

    # Compute number of blocks to be printed ...

    if noMatrixCols % NoColumns == 0:
        iNoBlocks = noMatrixCols/NoColumns;
    else:
        iNoBlocks = noMatrixCols/NoColumns + 1;

    # Loop over the number of blocks ...

    for ib in range( int(iNoBlocks) ):
        iFirstColumn = ib*NoColumns + 1
        iLastColumn = min ( (ib+1)*NoColumns, noMatrixCols )

        # Print title of matrix at top of each block ....

        print("Matrix: {:s} ".format(name) );

        # Label row and column nos */

        print("row/col      ", end="")
        colList = range(iFirstColumn, iLastColumn + 1)
        for col in [ *colList ]:
            print("          {:3d} ".format(col), end="")
        print("")

        # Loop over rows and print matrix elements ....

        ii = 1
        for row in matrix:
            print(" {:3d} ".format(ii), end="")
            colList = range( iFirstColumn, iLastColumn + 1)
            for col in [ *colList ]:
                if matrix.ndim == 1:
                    print(" {:12.5e} ".format( matrix[ii-1] ), end="")
                else:
                    print(" {:12.5e} ".format(matrix[ii-1][col-1]), end="")
            print("")
            ii = ii + 1
        print("")

# =====
# Compute maximum and minumun of three numbers ...
# =====

def maximum(a, b, c):
    list = [a, b, c]
    return max(list)

def minimum(a, b, c):
    list = [a, b, c]
    return min(list)

# =====
# Print element forces ...
# =====

def printElementForces(name, minF, maxF):
    if( minF < 0):
        print("----      Minimum {:s} = {:10.2f} (C) ... ".format( name, minF ) )
    else:
        print("----      Minimum {:s} = {:10.2f} (T) ... ".format( name, minF ) )

    if( maxF < 0):
        print("----      Maximum {:s} = {:10.2f} (C) ... ".format( name, maxF ) )
    else:
        print("----      Maximum {:s} = {:10.2f} (T) ... ".format( name, maxF ) )

# =====
# main method ...
# =====

def main():
    print("--- Enter TestTrussAnalysisTower01.main()      ... ");

```

```

print("--- ===== ... ");

print("--- ");
print("--- Part 1: Initialize coefficients for matrix equations ... ");

nonodes = 8; # <--- no of nodes in crane structure ...
maxterms = 4; # <--- max no terms in an equation of equilibrium ...

# Equilibrium and destination arrays ...

equilibrium = np.zeros(( 2*nonodes, maxterms ))
destination = np.zeros(( 2*nonodes, maxterms ))

# Node A ...

equilibrium[0][0] = 1 # < --- equilibrium in x direction ...
equilibrium[0][1] = 1

destination[0][0] = 7;
destination[0][1] = 14;

equilibrium[1][0] = 1 # < --- equilibrium in y direction ...
equilibrium[1][1] = 1

destination[1][0] = 1;
destination[1][1] = 15;

# Node B ...

equilibrium[2][0] = 1 # < --- equilibrium in x direction ...
equilibrium[2][1] = 1/math.sqrt(2)

destination[2][0] = 7;
destination[2][1] = 8;

equilibrium[3][0] = 1 # < --- equilibrium in y direction ...
equilibrium[3][1] = 1/math.sqrt(2)
equilibrium[3][2] = 1

destination[3][0] = 2;
destination[3][1] = 8;
destination[3][2] = 16;

# Node C ...

equilibrium[4][0] = 1/math.sqrt(2) # <--- equilibrium in x direction ...
equilibrium[4][1] = 1
equilibrium[4][2] = 1/math.sqrt(2)

destination[4][0] = 8;
destination[4][1] = 9;
destination[4][2] = 10;

equilibrium[5][0] = 1 # <--- equilibrium in y direction ...
equilibrium[5][1] = -1
equilibrium[5][2] = 1/math.sqrt(2)
equilibrium[5][3] = -1/math.sqrt(2)

destination[5][0] = 1;
destination[5][1] = 3;
destination[5][2] = 8;
destination[5][3] = 10;

# Node D ...

equilibrium[6][0] = 1; # <--- equilibrium in x direction ...
destination[6][0] = 9;

equilibrium[7][0] = 1; # <--- equilibrium in y direction ...
equilibrium[7][1] = -1;

destination[7][0] = 2;
destination[7][1] = 4;

# Node E ...

equilibrium[8][0] = 1; # <--- equilibrium in x direction ...
destination[8][0] = 11;

equilibrium[9][0] = -1 # <--- equilibrium in y direction ...
equilibrium[9][1] = 1

destination[9][0] = 3;
destination[9][1] = 5;

# Node F ...

```

```

equilibrium[10][0] = 1/math.sqrt(2) # <--- equilibrium in x direction ...
equilibrium[10][1] = 1
equilibrium[10][2] = 1/math.sqrt(2)

destination[10][0] = 10;
destination[10][1] = 11;
destination[10][2] = 12;

equilibrium[11][0] = 1 # <--- equilibrium in y direction ...
equilibrium[11][1] = -1
equilibrium[11][2] = -1/math.sqrt(2)
equilibrium[11][3] = 1/math.sqrt(2)

destination[11][0] = 6;
destination[11][1] = 4;
destination[11][2] = 10;
destination[11][3] = 12;

# Node G ...

equilibrium[12][0] = 1/math.sqrt(2) # <--- equilibrium in x direction ...
equilibrium[12][1] = 1

destination[12][0] = 12;
destination[12][1] = 13;

equilibrium[13][0] = 1; # <--- equilibrium in y direction ...
equilibrium[13][1] = 1/math.sqrt(2);

destination[13][0] = 5;
destination[13][1] = 12;

# Node H ...

equilibrium[14][0] = 1 # <--- equilibrium in x direction ...
destination[14][0] = 13;

equilibrium[15][0] = 1 # <--- equilibrium in y direction ...
destination[15][0] = 6;

PrintMatrix("Equilibrium", equilibrium);
PrintMatrix("Destination", destination);

print("--- ");
print("--- Part 2: Systematically Assemble Matrix A ... ");
print("--- ");

# Allocate memory for A matrix ...

A = np.zeros(( 2*nonodes, 2*nonodes ))

# Systematically assemble A matrix from equilibrium and destination arrays ...

for row in range(2*nonodes):
    print("--- ");
    for col in range(maxterms):
        eitem = equilibrium[row][col];
        ditem = int( destination[row][col] );
        if eitem != 0 or ditem != 0:
            print("---- (row,col) --> {:2d}, {:d}: eitem --> {:9.5f}, ditem --> {:3d} ...".format( row, col, eitem, ditem ));
            A[row][ditem-1] = eitem;

PrintMatrix("A", A);

print("--- ");
print("--- Part 2: Initialize 16x1 load vectors ... ");
print("--- ");

P = 250; Q = 0;
print("---- Load Case 1: P = {:.2f}, Q = {:.2f} ...".format(P,Q));

B1 = np.zeros(( 2*nonodes, 1 ))
B1[ 0][0] = 0.0;
B1[ 1][0] = 0.0;
B1[ 2][0] = 0.0;
B1[ 3][0] = 0.0;
B1[ 4][0] = -P;
B1[ 5][0] = 0.0;
B1[ 6][0] = 0.0;
B1[ 7][0] = 0.0;
B1[ 8][0] = -2*P;
B1[ 9][0] = 0.0;
B1[10][0] = 0.0;
B1[11][0] = 0.0;
B1[12][0] = -3*P;
B1[13][0] = -Q;
B1[14][0] = 0.0;

```

```

B1[15][0] = -Q;

PrintMatrix("Load Case 1:", B1);

P = 200; Q = 200;
print("---- Load Case 2: P = {:.2f}, Q = {:.2f} ...".format(P,Q));

B2 = np.zeros(( 2*nonodes, 1 ))
B2[ 0][0] = 0.0;
B2[ 1][0] = 0.0;
B2[ 2][0] = 0.0;
B2[ 3][0] = 0.0;
B2[ 4][0] = -P;
B2[ 5][0] = 0.0;
B2[ 6][0] = 0.0;
B2[ 7][0] = 0.0;
B2[ 8][0] = -2*P;
B2[ 9][0] = 0.0;
B2[10][0] = 0.0;
B2[11][0] = 0.0;
B2[12][0] = -3*P;
B2[13][0] = -Q;
B2[14][0] = 0.0;
B2[15][0] = -Q;

PrintMatrix("Load Case 2:", B2);

P = 0; Q = 500;
print("---- Load Case 3: P = {:.2f}, Q = {:.2f} ...".format(P,Q));

B3 = np.zeros(( 2*nonodes, 1 ))
B3[ 0][0] = 0.0;
B3[ 1][0] = 0.0;
B3[ 2][0] = 0.0;
B3[ 3][0] = 0.0;
B3[ 4][0] = -P;
B3[ 5][0] = 0.0;
B3[ 6][0] = 0.0;
B3[ 7][0] = 0.0;
B3[ 8][0] = -2*P;
B3[ 9][0] = 0.0;
B3[10][0] = 0.0;
B3[11][0] = 0.0;
B3[12][0] = -3*P;
B3[13][0] = -Q;
B3[14][0] = 0.0;
B3[15][0] = -Q;

PrintMatrix("Load Case 3:", B3);

print("---- ");
print("---- Part 4: Check properties of matrix A ... ");
print("---- -----");
print("---- ");

rank = matrix_rank(A)
det = np.linalg.det(A)

print("---- Matrix A: rank = {:f}, det = {:f} ...".format(rank, det) );

print("---- ");
print("---- Part 5: Solve A.F = B for three load cases ... ");
print("---- ");

F1 = np.linalg.solve(A, B1)
PrintMatrix("Load Case 1: Forces ...", F1);

F2 = np.linalg.solve(A, B2)
PrintMatrix("Load Case 2: Forces ...", F2);

F3 = np.linalg.solve(A, B3)
PrintMatrix("Load Case 3: Forces ...", F3);

print("---- ");
print("---- Part 6: Print support reactions and element-level forces ... ");

print("---- ");
print("---- Support Reactions and Element-Level Forces: Load Case 1: ...");
print("---- -----");
print("---- ");
print("---- Reaction A: R_ax = {:.2f} ... ".format( F1[13][0] ) );
print("---- : R_ay = {:.2f} ... ".format( F1[14][0] ) );
print("---- Reaction B: R_by = {:.2f} ... ".format( F1[15][0] ) );
print("---- ");
print("---- Element Level Forces:");
print("---- ");
print("---- Element A-C: F1 = {:.2f} ... ".format( F1[0][0] ) );

```

```

print("---- Element B-D: F2 = {:10.2f} ... ".format( F1[1][0] ) );

print("---- Element C-E: F3 = {:10.2f} ... ".format( F1[2][0] ) );
print("---- Element D-F: F4 = {:10.2f} ... ".format( F1[3][0] ) );

print("---- Element F-G: F5 = {:10.2f} ... ".format( F1[4][0] ) );
print("---- Element F-H: F6 = {:10.2f} ... ".format( F1[5][0] ) );

print("---- Element A-B: F7 = {:10.2f} ... ".format( F1[6][0] ) );
print("---- Element B-C: F8 = {:10.2f} ... ".format( F1[7][0] ) );
print("---- Element C-D: F09 = {:10.2f} ... ".format( F1[8][0] ) );
print("---- Element C-F: F10 = {:10.2f} ... ".format( F1[9][0] ) );
print("---- Element E-F: F11 = {:10.2f} ... ".format( F1[10][0] ) );
print("---- Element F-G: F12 = {:10.2f} ... ".format( F1[11][0] ) );
print("---- Element G-H: F13 = {:10.2f} ... ".format( F1[12][0] ) );

print("---- ");
print("---- Support Reactions and Element-Level Forces: Load Case 2: ...");
print("---- -----");
print("---- ");
print("---- Reaction A: R_ax = {:10.2f} ... ".format( F2[13][0] ) );
print("---- : R_ay = {:10.2f} ... ".format( F2[14][0] ) );
print("---- Reaction B: R_by = {:10.2f} ... ".format( F2[15][0] ) );
print("---- ");
print("---- Element Level Forces:");
print("---- ");
print("---- Element A-C: F1 = {:10.2f} ... ".format( F2[0][0] ) );
print("---- Element B-D: F2 = {:10.2f} ... ".format( F2[1][0] ) );

print("---- Element C-E: F3 = {:10.2f} ... ".format( F2[2][0] ) );
print("---- Element D-F: F4 = {:10.2f} ... ".format( F2[3][0] ) );

print("---- Element F-G: F5 = {:10.2f} ... ".format( F2[4][0] ) );
print("---- Element F-H: F6 = {:10.2f} ... ".format( F2[5][0] ) );

print("---- Element A-B: F7 = {:10.2f} ... ".format( F2[6][0] ) );
print("---- Element B-C: F8 = {:10.2f} ... ".format( F2[7][0] ) );
print("---- Element C-D: F09 = {:10.2f} ... ".format( F2[8][0] ) );
print("---- Element C-F: F10 = {:10.2f} ... ".format( F2[9][0] ) );
print("---- Element E-F: F11 = {:10.2f} ... ".format( F2[10][0] ) );
print("---- Element F-G: F12 = {:10.2f} ... ".format( F2[11][0] ) );
print("---- Element G-H: F13 = {:10.2f} ... ".format( F2[12][0] ) );

print("---- ");
print("---- Support Reactions and Element-Level Forces: Load Case 3: ...");
print("---- -----");
print("---- ");

print("---- Reaction A: R_ax = {:10.2f} ... ".format( F3[13][0] ) );
print("---- : R_ay = {:10.2f} ... ".format( F3[14][0] ) );
print("---- Reaction B: R_by = {:10.2f} ... ".format( F3[15][0] ) );
print("---- ");
print("---- Element Level Forces:");
print("---- ");
print("---- Element A-C: F1 = {:10.2f} ... ".format( F3[0][0] ) );
print("---- Element B-D: F2 = {:10.2f} ... ".format( F3[1][0] ) );

print("---- Element C-E: F3 = {:10.2f} ... ".format( F3[2][0] ) );
print("---- Element D-F: F4 = {:10.2f} ... ".format( F3[3][0] ) );

print("---- Element F-G: F5 = {:10.2f} ... ".format( F3[4][0] ) );
print("---- Element F-H: F6 = {:10.2f} ... ".format( F3[5][0] ) );

print("---- Element A-B: F7 = {:10.2f} ... ".format( F3[6][0] ) );
print("---- Element B-C: F8 = {:10.2f} ... ".format( F3[7][0] ) );
print("---- Element C-D: F09 = {:10.2f} ... ".format( F3[8][0] ) );
print("---- Element C-F: F10 = {:10.2f} ... ".format( F3[9][0] ) );
print("---- Element E-F: F11 = {:10.2f} ... ".format( F3[10][0] ) );
print("---- Element F-G: F12 = {:10.2f} ... ".format( F3[11][0] ) );
print("---- Element G-H: F13 = {:10.2f} ... ".format( F3[12][0] ) );

print("---- ");
print("---- Summary of Max/Min Reactions and Element-Level Forces ...");
print("---- -----");

MinRax = minimum( F1[13][0], F2[13][0], F3[13][0] )
MaxRax = maximum( F1[13][0], F2[13][0], F3[13][0] )

MinRay = minimum( F1[14][0], F2[14][0], F3[14][0] )
MaxRay = maximum( F1[14][0], F2[14][0], F3[14][0] )

MinRby = minimum( F1[15][0], F2[15][0], F3[15][0] )
MaxRby = maximum( F1[15][0], F2[15][0], F3[15][0] )

print("---- ");
print("---- Reaction A: Minimum_R_ax = {:7.2f}, Maximum_R_ax = {:7.2f} ... ".format( MinRax, MaxRax ) )
print("---- : Minimum_R_ay = {:7.2f}, Maximum_R_ay = {:7.2f} ... ".format( MinRay, MaxRay ) )

```

```

print("--- ");
print("--- Reaction B: Minimum R_by = {:7.2f}, Maximum R_by = {:7.2f} ... ".format( MinRby, MaxRby) )

MinF1 = minimum( F1[0][0], F2[0][0], F3[0][0] )
MaxF1 = maximum( F1[0][0], F2[0][0], F3[0][0] )

MinF2 = minimum( F1[1][0], F2[1][0], F3[1][0] )
MaxF2 = maximum( F1[1][0], F2[1][0], F3[1][0] )

MinF3 = minimum( F1[2][0], F2[2][0], F3[2][0] )
MaxF3 = maximum( F1[2][0], F2[2][0], F3[2][0] )

MinF4 = minimum( F1[3][0], F2[3][0], F3[3][0] )
MaxF4 = maximum( F1[3][0], F2[3][0], F3[3][0] )

MinF5 = minimum( F1[4][0], F2[4][0], F3[4][0] )
MaxF5 = maximum( F1[4][0], F2[4][0], F3[4][0] )

MinF6 = minimum( F1[5][0], F2[5][0], F3[5][0] )
MaxF6 = maximum( F1[5][0], F2[5][0], F3[5][0] )

MinF7 = minimum( F1[6][0], F2[6][0], F3[6][0] )
MaxF7 = maximum( F1[6][0], F2[6][0], F3[6][0] )

MinF8 = minimum( F1[7][0], F2[7][0], F3[7][0] )
MaxF8 = maximum( F1[7][0], F2[7][0], F3[7][0] )

MinF9 = minimum( F1[8][0], F2[8][0], F3[8][0] )
MaxF9 = maximum( F1[8][0], F2[8][0], F3[8][0] )

MinF10 = minimum( F1[9][0], F2[9][0], F3[9][0] )
MaxF10 = maximum( F1[9][0], F2[9][0], F3[9][0] )

MinF11 = minimum( F1[10][0], F2[10][0], F3[10][0] )
MaxF11 = maximum( F1[10][0], F2[10][0], F3[10][0] )

MinF12 = minimum( F1[11][0], F2[11][0], F3[11][0] )
MaxF12 = maximum( F1[11][0], F2[11][0], F3[11][0] )

MinF13 = minimum( F1[12][0], F2[12][0], F3[12][0] )
MaxF13 = maximum( F1[12][0], F2[12][0], F3[12][0] )

print("--- ");
print("--- Element A-C: ")
printElementForces( "F1", MinF1, MaxF1)

print("--- Element B-D: ")
printElementForces( "F2", MinF2, MaxF2)

print("--- Element C-G: ")
printElementForces( "F3", MinF3, MaxF3)

print("--- Element D-H: ")
printElementForces( "F4", MinF4, MaxF4)

print("--- Element F-I: ")
printElementForces( "F5", MinF5, MaxF5)

print("--- Element G-J: ")
printElementForces( "F6", MinF6, MaxF6)

print("--- Element H-K: ")
printElementForces( "F7", MinF7, MaxF7)

print("--- Element A-B: ")
printElementForces( "F8", MinF8, MaxF8)

print("--- Element B-C: ")
printElementForces( "F9", MinF9, MaxF9)

print("--- Element C-D: ")
printElementForces( "F10", MinF10, MaxF10)

print("--- Element C-H: ")
printElementForces( "F11", MinF11, MaxF11)

print("--- Element E-F: ")
printElementForces( "F12", MinF12, MaxF12)

print("--- Element F-G: ")
printElementForces( "F13", MinF13, MaxF13)

print("--- ===== ... ");
print("--- Leave TestTrussAnalysisTower01.main() ... ");

# call the main method ...

```

```
main()
```

Abbreviated Output:

```
--- Enter TestTrussAnalysisTower01.main()    ...
--- ===== ...
---
--- Part 1: Initialize coefficients for matrix equations ...

Matrix: Equilibrium
row/col      1      2      3      4
  1      1.00000e+00      1.00000e+00      0.00000e+00      0.00000e+00
  2      1.00000e+00      1.00000e+00      0.00000e+00      0.00000e+00
  3      1.00000e+00      7.07107e-01      0.00000e+00      0.00000e+00
  4      1.00000e+00      7.07107e-01      1.00000e+00      0.00000e+00
  5      7.07107e-01      1.00000e+00      7.07107e-01      0.00000e+00
  6      1.00000e+00      -1.00000e+00      7.07107e-01      -7.07107e-01
  7      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  8      1.00000e+00      -1.00000e+00      0.00000e+00      0.00000e+00
  9      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 10      -1.00000e+00      1.00000e+00      0.00000e+00      0.00000e+00
 11      7.07107e-01      1.00000e+00      7.07107e-01      0.00000e+00
 12      1.00000e+00      -1.00000e+00      -7.07107e-01      7.07107e-01
 13      7.07107e-01      1.00000e+00      0.00000e+00      0.00000e+00
 14      1.00000e+00      7.07107e-01      0.00000e+00      0.00000e+00
 15      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 16      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00

Matrix: Destination
row/col      1      2      3      4
  1      7.00000e+00      1.40000e+01      0.00000e+00      0.00000e+00
  2      1.00000e+00      1.50000e+01      0.00000e+00      0.00000e+00
  3      7.00000e+00      8.00000e+00      0.00000e+00      0.00000e+00
  4      2.00000e+00      8.00000e+00      1.60000e+01      0.00000e+00
  5      8.00000e+00      9.00000e+00      1.00000e+01      0.00000e+00
  6      1.00000e+00      3.00000e+00      8.00000e+00      1.00000e+01
  7      9.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  8      2.00000e+00      4.00000e+00      0.00000e+00      0.00000e+00
  9      1.10000e+01      0.00000e+00      0.00000e+00      0.00000e+00
 10      3.00000e+00      5.00000e+00      0.00000e+00      0.00000e+00
 11      1.00000e+01      1.10000e+01      1.20000e+01      0.00000e+00
 12      6.00000e+00      4.00000e+00      1.00000e+01      1.20000e+01
 13      1.20000e+01      1.30000e+01      0.00000e+00      0.00000e+00
 14      5.00000e+00      1.20000e+01      0.00000e+00      0.00000e+00
 15      1.30000e+01      0.00000e+00      0.00000e+00      0.00000e+00
 16      6.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00

---
--- Part 2: Systematically Assemble Matrix A ...
---
---
--- (row,col) -->  0, 0: eitem -->  1.00000, ditem -->  7 ...
--- (row,col) -->  0, 1: eitem -->  1.00000, ditem --> 14 ...

.... output removed ...

--- (row,col) --> 14, 0: eitem -->  1.00000, ditem --> 13 ...
---
--- (row,col) --> 15, 0: eitem -->  1.00000, ditem -->  6 ...

Matrix: A
row/col      1      2      3      4      5      6
  1      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  2      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  3      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  4      0.00000e+00      1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  5      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  6      1.00000e+00      0.00000e+00      -1.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  7      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
  8      0.00000e+00      1.00000e+00      0.00000e+00      -1.00000e+00      0.00000e+00      0.00000e+00
  9      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 10      0.00000e+00      0.00000e+00      -1.00000e+00      0.00000e+00      1.00000e+00      0.00000e+00
 11      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 12      0.00000e+00      0.00000e+00      0.00000e+00      -1.00000e+00      0.00000e+00      1.00000e+00
 13      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 14      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      1.00000e+00      0.00000e+00
 15      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00
 16      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      0.00000e+00      1.00000e+00

Matrix: A
row/col      7      8      9      10      11      12
```

1	1.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
2	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
3	1.00000e+00	7.07107e-01	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
4	0.00000e+00	7.07107e-01	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
5	0.00000e+00	7.07107e-01	1.00000e+00	7.07107e-01	0.00000e+00	0.00000e+00
6	0.00000e+00	7.07107e-01	0.00000e+00	-7.07107e-01	0.00000e+00	0.00000e+00
7	0.00000e+00	0.00000e+00	1.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
8	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
9	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	1.00000e+00	0.00000e+00
10	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
11	0.00000e+00	0.00000e+00	0.00000e+00	7.07107e-01	1.00000e+00	7.07107e-01
12	0.00000e+00	0.00000e+00	0.00000e+00	-7.07107e-01	0.00000e+00	7.07107e-01
13	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	7.07107e-01
14	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	7.07107e-01
15	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
16	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00

Matrix: A

row/col	13	14	15	16
1	0.00000e+00	1.00000e+00	0.00000e+00	0.00000e+00
2	0.00000e+00	0.00000e+00	1.00000e+00	0.00000e+00
3	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
4	0.00000e+00	0.00000e+00	0.00000e+00	1.00000e+00
5	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
6	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
7	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
8	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
9	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
10	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
11	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
12	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
13	1.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
14	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
15	1.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00
16	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00

--- Part 2: Initialize 16x1 load vectors ...

--- Load Case 1: P = 250.00, Q = 0.00 ...

Matrix: Load Case 1:

row/col	1
1	0.00000e+00
2	0.00000e+00
3	0.00000e+00
4	0.00000e+00
5	-2.50000e+02
6	0.00000e+00
7	0.00000e+00
8	0.00000e+00
9	-5.00000e+02
10	0.00000e+00
11	0.00000e+00
12	0.00000e+00
13	-7.50000e+02
14	0.00000e+00
15	0.00000e+00
16	0.00000e+00

--- Load Case 2: P = 200.00, Q = 200.00 ...

Matrix: Load Case 2:

row/col	1
1	0.00000e+00
2	0.00000e+00
3	0.00000e+00
4	0.00000e+00
5	-2.00000e+02
6	0.00000e+00
7	0.00000e+00
8	0.00000e+00
9	-4.00000e+02
10	0.00000e+00
11	0.00000e+00
12	0.00000e+00
13	-6.00000e+02
14	-2.00000e+02
15	0.00000e+00
16	-2.00000e+02

--- Load Case 3: P = 0.00, Q = 500.00 ...

Matrix: Load Case 3:

row/col	1
1	0.00000e+00
2	0.00000e+00
3	0.00000e+00
4	0.00000e+00

```

5      0.00000e+00
6      0.00000e+00
7      0.00000e+00
8      0.00000e+00
9      0.00000e+00
10     0.00000e+00
11     0.00000e+00
12     0.00000e+00
13     0.00000e+00
14     -5.00000e+02
15     0.00000e+00
16     -5.00000e+02

---
--- Part 4: Check properties of matrix A ...
--- ----- ...
---

--- Matrix A: rank = 16.000000, det = 0.353553 ...

---
--- Part 5: Solve A.F = B for three load cases ...
---

Matrix: Load Case 1: Forces ...
row/col      1
1      3.50000e+03
2     -2.00000e+03
3      7.50000e+02
4     -2.00000e+03
5      7.50000e+02
6     -2.38320e-13
7      1.50000e+03
8     -2.12132e+03
9      0.00000e+00
10     1.76777e+03
11     -5.00000e+02
12     -1.06066e+03
13     -6.47630e-14
14     -1.50000e+03
15     -3.50000e+03
16      3.50000e+03

Matrix: Load Case 2: Forces ...
row/col      1
1      2.60000e+03
2     -1.80000e+03
3      4.00000e+02
4     -1.80000e+03
5      4.00000e+02
6     -2.00000e+02
7      1.20000e+03
8     -1.69706e+03
9      0.00000e+00
10     1.41421e+03
11     -4.00000e+02
12     -8.48528e+02
13     -1.02141e-14
14     -1.20000e+03
15     -2.60000e+03
16      3.00000e+03

Matrix: Load Case 3: Forces ...
row/col      1
1     -5.00000e+02
2     -5.00000e+02
3     -5.00000e+02
4     -5.00000e+02
5     -5.00000e+02
6     -5.00000e+02
7     -2.79776e-14
8      0.00000e+00
9      0.00000e+00
10     -0.00000e+00
11      0.00000e+00
12     -0.00000e+00
13     -2.22045e-16
14     2.79776e-14
15      5.00000e+02
16      5.00000e+02

---
--- Part 6: Print support reactions and element-level forces ...
---
--- Support Reactions and Element-Level Forces: Load Case 1: ...
--- ----- ...
---

```

```

--- Reaction A: R_ax = -1500.00 ...
--- : R_ay = -3500.00 ...
--- Reaction B: R_by = 3500.00 ...
---
--- Element Level Forces:
---
--- Element A-C: F1 = 3500.00 ...
--- Element B-D: F2 = -2000.00 ...
--- Element C-E: F3 = 750.00 ...
--- Element D-F: F4 = -2000.00 ...
--- Element F-G: F5 = 750.00 ...
--- Element F-H: F6 = -0.00 ...
--- Element A-B: F7 = 1500.00 ...
--- Element B-C: F8 = -2121.32 ...
--- Element C-D: F09 = 0.00 ...
--- Element C-F: F10 = 1767.77 ...
--- Element E-F: F11 = -500.00 ...
--- Element F-G: F12 = -1060.66 ...
--- Element G-H: F13 = -0.00 ...
---
--- Support Reactions and Element-Level Forces: Load Case 2: ...
-----
---
--- Reaction A: R_ax = -1200.00 ...
--- : R_ay = -2600.00 ...
--- Reaction B: R_by = 3000.00 ...
---
--- Element Level Forces:
---
--- Element A-C: F1 = 2600.00 ...
--- Element B-D: F2 = -1800.00 ...
--- Element C-E: F3 = 400.00 ...
--- Element D-F: F4 = -1800.00 ...
--- Element F-G: F5 = 400.00 ...
--- Element F-H: F6 = -200.00 ...
--- Element A-B: F7 = 1200.00 ...
--- Element B-C: F8 = -1697.06 ...
--- Element C-D: F09 = 0.00 ...
--- Element C-F: F10 = 1414.21 ...
--- Element E-F: F11 = -400.00 ...
--- Element F-G: F12 = -848.53 ...
--- Element G-H: F13 = -0.00 ...
---
--- Support Reactions and Element-Level Forces: Load Case 3: ...
-----
---
--- Reaction A: R_ax = 0.00 ...
--- : R_ay = 500.00 ...
--- Reaction B: R_by = 500.00 ...
---
--- Element Level Forces:
---
--- Element A-C: F1 = -500.00 ...
--- Element B-D: F2 = -500.00 ...
--- Element C-E: F3 = -500.00 ...
--- Element D-F: F4 = -500.00 ...
--- Element F-G: F5 = -500.00 ...
--- Element F-H: F6 = -500.00 ...
--- Element A-B: F7 = -0.00 ...
--- Element B-C: F8 = 0.00 ...
--- Element C-D: F09 = 0.00 ...
--- Element C-F: F10 = -0.00 ...
--- Element E-F: F11 = 0.00 ...
--- Element F-G: F12 = -0.00 ...
--- Element G-H: F13 = -0.00 ...
---
--- Summary of Max/Min Reactions and Element-Level Forces ...
-----
---
--- Reaction A: Minimum R_ax = -1500.00, Maximum R_ax = 0.00 ...
--- : Minimum R_ay = -3500.00, Maximum R_ay = 500.00 ...
---
--- Reaction B: Minimum R_by = 500.00, Maximum R_by = 3500.00 ...
---
--- Element A-C:
--- Minimum F1 = -500.00 (C) ...
--- Maximum F1 = 3500.00 (T) ...
--- Element B-D:
--- Minimum F2 = -2000.00 (C) ...
--- Maximum F2 = -500.00 (C) ...
--- Element C-G:
--- Minimum F3 = -500.00 (C) ...
--- Maximum F3 = 750.00 (T) ...
--- Element D-H:
--- Minimum F4 = -2000.00 (C) ...
--- Maximum F4 = -500.00 (C) ...
--- Element F-I:

```

```

---      Minimum F5 =    -500.00 (C) ...
---      Maximum F5 =     750.00 (T) ...
--- Element G-J:
---      Minimum F6 =    -500.00 (C) ...
---      Maximum F6 =     -0.00 (C) ...
--- Element H-K:
---      Minimum F7 =     -0.00 (C) ...
---      Maximum F7 =    1500.00 (T) ...
--- Element A-B:
---      Minimum F8 =   -2121.32 (C) ...
---      Maximum F8 =      0.00 (T) ...
--- Element B-C:
---      Minimum F9 =      0.00 (T) ...
---      Maximum F9 =      0.00 (T) ...
--- Element C-D:
---      Minimum F10 =    -0.00 (T) ...
---      Maximum F10 =    1767.77 (T) ...
--- Element C-H:
---      Minimum F11 =   -500.00 (C) ...
---      Maximum F11 =      0.00 (T) ...
--- Element E-F:
---      Minimum F12 =  -1060.66 (C) ...
---      Maximum F12 =    -0.00 (T) ...
--- Element F-G:
---      Minimum F13 =    -0.00 (C) ...
---      Maximum F13 =    -0.00 (C) ...
--- ===== ...
--- Leave TestTrussAnalysisTower01.main()    ...

```