Question 4: 10 points. In the design of highway bridge structures and crane structures, engineers are often required to compute the maximum and minimum member forces and support reactions due to a variety of loading conditions.

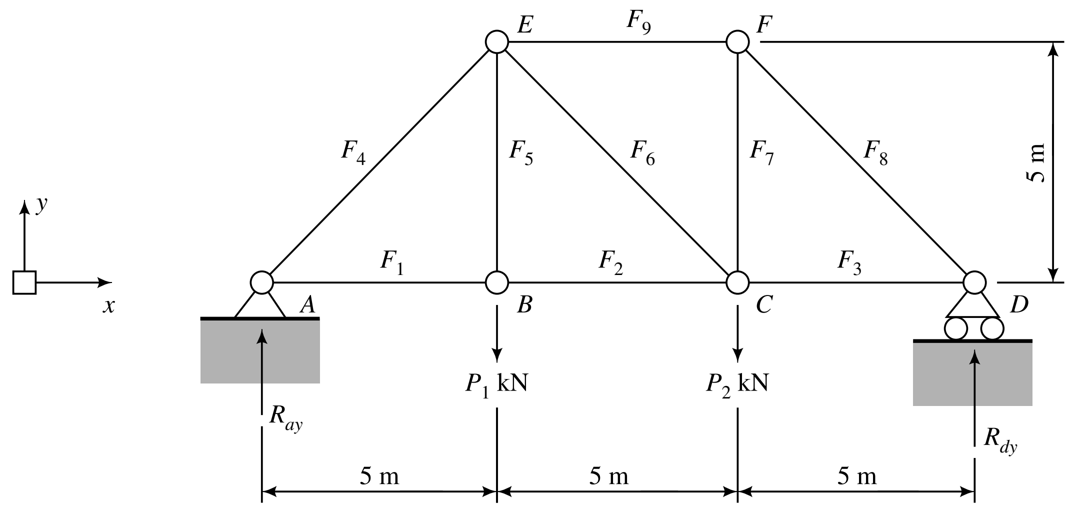

Figure 3: Front elevation of pin-jointed bridge truss.

Figure 3 shows a nine bar pin-jointed bridge truss carrying vertical loads P_1 kN and P_2 kN at joints `B` and `C`. The symbols F_1, F_2, $\cdots$ F_9 represent the axial forces in truss members 1 through 9, and R_{ay} and R_{dy} are the support reactions at joints `A` and `D`. (Notice that because support D is on a roller and there are no horizontal components of external loads, horizontal reactions will be zero.)

Write down the equations of equilibrium for joints `A` through `F` and put the equations in matrix form. Now suppose that a heavy load moves across the bridge and that, for engineering purposes, it can be represented by the sequence of external load vectors

$$\begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix} \tag{5}$$

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 D, and axial forces in each of the truss members.

Equations of Equilibrium:

At Joint A:

$$\sum F_x = 0, \qquad \frac{F_4}{\sqrt{2}} + F_1 = 0. \tag{6}$$

$$\sum F_y = 0, \qquad \frac{F_4}{\sqrt{2}} + R_{ay} = 0. \tag{7}$$

At Joint B:

$$\sum F_x = 0, \qquad F_1 - F_2 = 0. \tag{8}$$

$$\sum F_y = 0, \qquad F_5 - P_1 = 0. \tag{9}$$

At Joint C:

$$\sum F_x = 0, \qquad F_2 - F_3 + \frac{F_6}{\sqrt{2}} = 0. \tag{10}$$

$$\sum F_y = 0, \qquad \frac{F_6}{\sqrt{2}} + F_7 - P_2 = 0. \tag{11}$$

At Joint D:

$$\sum F_x = 0, \qquad F_3 + \frac{F_8}{\sqrt{2}} = 0. \tag{12}$$

$$\sum F_y = 0, \qquad \frac{F_8}{\sqrt{2}} + R_{dy} = 0. \tag{13}$$

At Joint E:

$$\sum F_x = 0, \qquad -\frac{F_4}{\sqrt{2}} + \frac{F_6}{\sqrt{2}} + F_9 = 0. \tag{14}$$

$$\sum F_y = 0, \qquad -\frac{F_4}{\sqrt{2}} - F_5 - \frac{F_6}{\sqrt{2}} = 0. \tag{15}$$

At Joint F:

$$\sum F_x = 0, \qquad \frac{F_8}{\sqrt{2}} - F_9 = 0. \tag{16}$$

$$\sum F_y = 0, \qquad -\frac{F_7}{\sqrt{2}} - \frac{F_8}{\sqrt{2}} = 0. \tag{17}$$

System of Matrix Equations: Collecting equations 6 through 17 and writing in matrix form: ...

Python Source Code:

```
# ===============================================================
# TestTrussAnalysis02.py: Compute distribution of element forces
# and support reactions in a nine-bar truss.
#
# Written by: Mark Austin                          November 2023
# ===============================================================

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} = {:7.2f} (C) ... ".format( name, minF ) )
    else:
        print("---     Minimum {:s} = {:7.2f} (T) ... ".format( name, minF ) )

    if( maxF < 0):
        print("---     Maximum {:s} = {:7.2f} (C) ... ".format( name, maxF ) )
    else:
        print("---     Maximum {:s} = {:7.2f} (T) ... ".format( name, maxF ) )

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

def main():
```

```
print("--- Enter TestTrussAnalysis02.main()         ... ");
print("--- ======================================== ... ");

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

# Node A ...

a11  = 1               # < --- equilibrium in x direction ...
a14  = 1/math.sqrt(2)
a110 = 1
a24  = 1/math.sqrt(2)  # < --- equilibrium in y direction ...
a211 = 1

# Node B ...

a31  =  1              # < --- equilibrium in x direction ...
a32  = -1
a45  = -1              # < --- equilibrium in y direction ...

# Node C ...

a52  =  1              # < --- equilibrium in x direction ...
a53  = -1
a56  =  1/math.sqrt(2)

a66  = -1/math.sqrt(2) # < --- equilibrium in y direction ...
a67  = -1

# Node D ...

a73  = 1               # < --- equilibrium in x direction ...
a78  = 1/math.sqrt(2)
a88  = 1/math.sqrt(2)  # < --- equilibrium in y direction ...
a812 = 1

# Node E ...

a99  =  1              # < --- equilibrium in x direction ...
a96  =  1/math.sqrt(2)
a94  = -1/math.sqrt(2)

a104 =  1/math.sqrt(2) # < --- equilibrium in y direction ...
a105 =  1
a106 =  1/math.sqrt(2)

# Node F ...

a118 = -1/math.sqrt(2)  # < --- equilibrium in x direction ...
a119 =  1
a127 =  1               # < --- equilibrium in y direction ...
a128 =  1/math.sqrt(2)

print("--- ");
print("--- Part 2: Create test matrix A ... ");
```

```
print("--- ");

A = np.array([ [ a11,    0,    0,  a14,    0,    0,    0,    0,    0, a110,    0,    0 ],
               [   0,    0,    0,  a24,    0,    0,    0,    0,    0,    0, a211,    0 ],
               [ a31,  a32,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0 ],
               [   0,    0,    0,    0,  a45,    0,    0,    0,    0,    0,    0,    0 ],
               [   0,  a52,  a53,    0,    0,  a56,    0,    0,    0,    0,    0,    0 ],
               [   0,    0,    0,    0,    0,  a66,  a67,    0,    0,    0,    0,    0 ],
               [   0,    0,  a73,    0,    0,    0,    0,  a78,    0,    0,    0,    0 ],
               [   0,    0,    0,    0,    0,    0,    0,  a88,    0,    0,    0, a812 ],
               [   0,    0,    0,  a94,    0,  a96,    0,    0,  a99,    0,    0,    0 ],
               [   0,    0,    0, a104, a105, a106,    0,    0,    0,    0,    0,    0 ],
               [   0,    0,    0,    0,    0,    0,    0, a118, a119,    0,    0,    0 ],
               [   0,    0,    0,    0,    0,    0, a127, a128,    0,    0,    0,    0 ] ]);
PrintMatrix("A", A);

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

print("--- Load Case 1: b4 = -10, b6 = 0 ...");

b4 = -10.0; b6 = 0.0

B1 = np.array([ [0], [0], [0], [b4],  [0], [b6], [0], [0], [0], [0], [0], [0] ]);
PrintMatrix("Load Case 1: B", B1);

print("--- Load Case 2: b4 = -5, b6 = -5 ...");

b4 = -5.0; b6 = -5.0

B2 = np.array([ [0], [0], [0], [b4],  [0], [b6], [0], [0], [0], [0], [0], [0] ]);
PrintMatrix("Load Case 2: B", B2);

print("--- Load Case 3: b4 = 0, b6 = -10 ...");

b4 = -0.0; b6 = -10.0

B3 = np.array([ [0], [0], [0], [b4],  [0], [b6], [0], [0], [0], [0], [0], [0] ]);
PrintMatrix("Load Case 3: B", B3);

print("--- ");
print("--- Part 4: Check properties of matrix A ... ");
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("---   Reaction A: R_ax = {:7.2f} ... ".format( F1[9][0] ) );
print("---             : R_ay = {:7.2f} ... ".format( F1[10][0] ) );
print("---   Reaction D: R_dy = {:7.2f} ... ".format( F1[11][0] ) );
print("--- ");
print("--- Element Level Forces:");
print("--- ");
print("---   Element A-B: F1 = {:7.2f} ... ".format( F1[0][0] ) );
print("---   Element B-C: F2 = {:7.2f} ... ".format( F1[1][0] ) );
print("---   Element C-D: F3 = {:7.2f} ... ".format( F1[2][0] ) );
print("---   Element A-E: F4 = {:7.2f} ... ".format( F1[3][0] ) );
print("---   Element B-E: F5 = {:7.2f} ... ".format( F1[4][0] ) );
print("---   Element C-E: F6 = {:7.2f} ... ".format( F1[5][0] ) );
print("---   Element C-F: F7 = {:7.2f} ... ".format( F1[6][0] ) );
print("---   Element D-F: F8 = {:7.2f} ... ".format( F1[7][0] ) );
print("---   Element E-F: F9 = {:7.2f} ... ".format( F1[8][0] ) );

print("--- ");
print("--- Support Reactions and Element-Level Forces: Load Case 2:");
print("--- ");
print("---   Reaction A: R_ax = {:7.2f} ... ".format( F2[9][0] ) );
print("---             : R_ay = {:7.2f} ... ".format( F2[10][0] ) );
print("---   Reaction D: R_dy = {:7.2f} ... ".format( F2[11][0] ) );
print("--- ");
print("--- Element Level Forces:");
print("--- ");
print("---   Element A-B: F1 = {:7.2f} ... ".format( F2[0][0] ) );
print("---   Element B-C: F2 = {:7.2f} ... ".format( F2[1][0] ) );
print("---   Element C-D: F3 = {:7.2f} ... ".format( F2[2][0] ) );
print("---   Element A-E: F4 = {:7.2f} ... ".format( F2[3][0] ) );
print("---   Element B-E: F5 = {:7.2f} ... ".format( F2[4][0] ) );
print("---   Element C-E: F6 = {:7.2f} ... ".format( F2[5][0] ) );
print("---   Element C-F: F7 = {:7.2f} ... ".format( F2[6][0] ) );
print("---   Element D-F: F8 = {:7.2f} ... ".format( F2[7][0] ) );
print("---   Element E-F: F9 = {:7.2f} ... ".format( F2[8][0] ) );

print("--- ");
print("--- Support Reactions and Element-Level Forces: Load Case 3:");
print("--- ");
print("---   Reaction A: R_ax = {:7.2f} ... ".format( F3[9][0] ) );
print("---             : R_ay = {:7.2f} ... ".format( F3[10][0] ) );
print("---   Reaction D: R_dy = {:7.2f} ... ".format( F3[11][0] ) );
```

```
print("--- ");
print("--- Element Level Forces:");
print("--- ");
print("---   Element A-B: F1 = {:7.2f} ... ".format( F3[0][0] ) );
print("---   Element B-C: F2 = {:7.2f} ... ".format( F3[1][0] ) );
print("---   Element C-D: F3 = {:7.2f} ... ".format( F3[2][0] ) );
print("---   Element A-E: F4 = {:7.2f} ... ".format( F3[3][0] ) );
print("---   Element B-E: F5 = {:7.2f} ... ".format( F3[4][0] ) );
print("---   Element C-E: F6 = {:7.2f} ... ".format( F3[5][0] ) );
print("---   Element C-F: F7 = {:7.2f} ... ".format( F3[6][0] ) );
print("---   Element D-F: F8 = {:7.2f} ... ".format( F3[7][0] ) );
print("---   Element E-F: F9 = {:7.2f} ... ".format( F3[8][0] ) );

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

MinRax = minimum( F1[9][0], F2[9][0], F3[9][0] )
MaxRax = maximum( F1[9][0], F2[9][0], F3[9][0] )

MinRay = minimum( F1[10][0], F2[10][0], F3[10][0] )
MaxRay = maximum( F1[10][0], F2[10][0], F3[10][0] )

MinRdy = minimum( F1[11][0], F2[11][0], F3[11][0] )
MaxRdy = maximum( F1[11][0], F2[11][0], F3[11][0] )

print("--- ");
print("---   Reaction A: Minimum R_ax = {:7.2f}, Maximum R_ax = {:7.2f} ... ".format( MinRax, Max
print("---             : Minimum R_ay = {:7.2f}, Maximum R_ay = {:7.2f} ... ".format( MinRay, Max
print("--- ");
print("---   Reaction D: Minimum R_dy = {:7.2f}, Maximum R_dy = {:7.2f} ... ".format( MinRdy, Max

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] )

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

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

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

    print("---   Element A-E: ")
    printElementForces( "F4", MinF4, MaxF4)

    print("---   Element B-E: ")
    printElementForces( "F5", MinF5, MaxF5)

    print("---   Element C-E: ")
    printElementForces( "F6", MinF6, MaxF6)

    print("---   Element C-F: ")
    printElementForces( "F7", MinF7, MaxF7)

    print("---   Element D-F: ")
    printElementForces( "F8", MinF8, MaxF8)

    print("---   Element E-F: ")
    printElementForces( "F9", MinF9, MaxF9)

    print("--- ======================================== ... ");
    print("--- Leave TestTrussAnalysis02.main()          ... ");

# call the main method ...

main()
```

Abbreviated Program Output:

```
---
--- Part 1: Initialize coefficients for matrix equations ...
---
--- Part 2: Create test matrix A ...
---

Matrix: A
row/col             1             2             3             4             5             6
   1      1.00000e+00   0.00000e+00   0.00000e+00   7.07107e-01   0.00000e+00   0.00000e+00
   2      0.00000e+00   0.00000e+00   0.00000e+00   7.07107e-01   0.00000e+00   0.00000e+00
   3      1.00000e+00  -1.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
   4      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00  -1.00000e+00   0.00000e+00
   5      0.00000e+00   1.00000e+00  -1.00000e+00   0.00000e+00   0.00000e+00   7.07107e-01
   6      0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00  -7.07107e-01
   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  -7.07107e-01   0.00000e+00   7.07107e-01
  10      0.00000e+00   0.00000e+00   0.00000e+00   7.07107e-01   1.00000e+00   7.07107e-01
  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   0.00000e+00   0.00000e+00   0.00000e+00

Matrix: A
row/col              7             8             9            10            11            12
   1       0.00000e+00   0.00000e+00   0.00000e+00   1.00000e+00   0.00000e+00   0.00000e+00
   2       0.00000e+00   0.00000e+00   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   0.00000e+00   0.00000e+00
   4       0.00000e+00   0.00000e+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   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
   7       0.00000e+00   7.07107e-01   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
   8       0.00000e+00   7.07107e-01   0.00000e+00   0.00000e+00   0.00000e+00   1.00000e+00
   9       0.00000e+00   0.00000e+00   1.00000e+00   0.00000e+00   0.00000e+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  -7.07107e-01   1.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00
  12       1.00000e+00   7.07107e-01   0.00000e+00   0.00000e+00   0.00000e+00   0.00000e+00

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

---
--- Load Case 1: b4 = -10, b6 = 0 ...

Matrix: Load Case 1: B
row/col              1
   1       0.00000e+00
   2       0.00000e+00
   3       0.00000e+00
   4      -1.00000e+01
   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

--- Load Case 2: b4 = -5, b6 = -5 ...
Matrix: Load Case 2: B
row/col              1
   1       0.00000e+00
   2       0.00000e+00
   3       0.00000e+00
   4      -5.00000e+00
   5       0.00000e+00
   6      -5.00000e+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

--- Load Case 3: b4 = 0, b6 = -10 ...
```

```
Matrix: Load Case 3: B
row/col             1
   1       0.00000e+00
   2       0.00000e+00
   3       0.00000e+00
   4      -0.00000e+00
   5       0.00000e+00
   6      -1.00000e+01
   7       0.00000e+00
   8       0.00000e+00
   9       0.00000e+00
  10       0.00000e+00
  11       0.00000e+00
  12       0.00000e+00

---
--- Part 4: Check properties of matrix A ...
---
--- Matrix A: rank = 12.000000, det = 1.060660  ...
---
--- Part 5: Solve A.F = B for three load cases ...
---

Matrix: Load Case 1: Forces ...
row/col             1
   1       6.66667e+00
   2       6.66667e+00

   ... lines of output removed ...

  11       6.66667e+00
  12       3.33333e+00

Matrix: Load Case 2: Forces ...
row/col             1
   1       5.00000e+00
   2       5.00000e+00

   ... lines of output removed ...

  11       5.00000e+00
  12       5.00000e+00

Matrix: Load Case 3: Forces ...
row/col             1
   1       3.33333e+00
   2       3.33333e+00

   ... lines of output removed ...

  11       3.33333e+00
  12       6.66667e+00

---
--- Part 6: Print support reactions and element-level forces ...
```

```
---

--- Support Reactions and Element-Level Forces: Load Case 1:
---
---   Reaction A: R_ax =   -0.00 ...
---             : R_ay =    6.67 ...
---   Reaction D: R_dy =    3.33 ...
---
--- Element Level Forces:
---
---   Element A-B: F1 =    6.67 ...
---   Element B-C: F2 =    6.67 ...
---   Element C-D: F3 =    3.33 ...
---   Element A-E: F4 =   -9.43 ...
---   Element B-E: F5 =   10.00 ...
---   Element C-E: F6 =   -4.71 ...
---   Element C-F: F7 =    3.33 ...
---   Element D-F: F8 =   -4.71 ...
---   Element E-F: F9 =   -3.33 ...
---
--- Support Reactions and Element-Level Forces: Load Case 2:
---
---   Reaction A: R_ax =   -0.00 ...
---             : R_ay =    5.00 ...
---   Reaction D: R_dy =    5.00 ...
---
--- Element Level Forces:
---
---   Element A-B: F1 =    5.00 ...
---   Element B-C: F2 =    5.00 ...
---   Element C-D: F3 =    5.00 ...
---   Element A-E: F4 =   -7.07 ...
---   Element B-E: F5 =    5.00 ...
---   Element C-E: F6 =   -0.00 ...
---   Element C-F: F7 =    5.00 ...
---   Element D-F: F8 =   -7.07 ...
---   Element E-F: F9 =   -5.00 ...
---
--- Support Reactions and Element-Level Forces: Load Case 3:
---
---   Reaction A: R_ax =   -0.00 ...
---             : R_ay =    3.33 ...
---   Reaction D: R_dy =    6.67 ...
---
--- Element Level Forces:
---
---   Element A-B: F1 =    3.33 ...
---   Element B-C: F2 =    3.33 ...
---   Element C-D: F3 =    6.67 ...
---   Element A-E: F4 =   -4.71 ...
---   Element B-E: F5 =   -0.00 ...
---   Element C-E: F6 =    4.71 ...
---   Element C-F: F7 =    6.67 ...
---   Element D-F: F8 =   -9.43 ...
---   Element E-F: F9 =   -6.67 ...
```

```
---
--- Summary of Max/Min Reactions and Element-Level Forces ...
--- ----------------------------------------------------- ...
---
---   Reaction A: Minimum R_ax =   -0.00, Maximum R_ax =   -0.00 ...
---             : Minimum R_ay =    3.33, Maximum R_ay =    6.67 ...
---
---   Reaction D: Minimum R_dy =    3.33, Maximum R_dy =    6.67 ...
---
---   Element A-B:
---     Minimum F1 =    3.33 (T) ...
---     Maximum F1 =    6.67 (T) ...
---   Element B-C:
---     Minimum F2 =    3.33 (T) ...
---     Maximum F2 =    6.67 (T) ...
---   Element C-D:
---     Minimum F3 =    3.33 (T) ...
---     Maximum F3 =    6.67 (T) ...
---   Element A-E:
---     Minimum F4 =   -9.43 (C) ...
---     Maximum F4 =   -4.71 (C) ...
---   Element B-E:
---     Minimum F5 =   -0.00 (T) ...
---     Maximum F5 =   10.00 (T) ...
---   Element C-E:
---     Minimum F6 =   -4.71 (C) ...
---     Maximum F6 =    4.71 (T) ...
---   Element C-F:
---     Minimum F7 =    3.33 (T) ...
---     Maximum F7 =    6.67 (T) ...
---   Element D-F:
---     Minimum F8 =   -9.43 (C) ...
---     Maximum F8 =   -4.71 (C) ...
---   Element E-F:
---     Minimum F9 =   -6.67 (C) ...
---     Maximum F9 =   -3.33 (C) ...
```