Simplified Analysis of Trusses with Python

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 353, Fall Semester 2024

September 17, 2024

Overview

Solution of

Linear Matrix Equations

Linear Matrix Equations

Definition. A system of m linear equations with n unknowns may be written

$$\begin{array}{rcrcrcrcl} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \cdots & & \vdots & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & a_{m3}x_3 & + & \cdots & + & a_{mn}x_n & = & b_m \end{array} \tag{1}$$

Points to note:

- The constants a_{11}, a_{21}, a_{31}, $\cdots$ a_{mn} and b_1, b_2, $\cdots$ b_m are called the equation coefficients.
- The variables x_1, x_2 $\cdots$ x_n are the unknowns in the system of equations.

Linear Matrix Equations

Matrix Form. The matrix counterpart of 1 is $[A] \cdot [X] = [B]$, where

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ \vdots \\ b_m \end{bmatrix} \qquad (2)$$

Points to note:

- Matrices A and X have dimensions $(m \times n)$ and $(n \times 1)$, respectively.
- Column vector B has dimensions $(m \times 1)$.

Augmented Matrix Form

Augmented Matrix Form. An augmented matrix for a system of equations is matrix A juxtiposed with matrix B.

Example. The augmented matrix form form of equation 2 is:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & & \vdots & b_2 \\ \vdots & & & \vdots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} & b_m \end{array}\right] \tag{3}$$

The augmented matrix dimensions are $(m \times (n+1))$.

Classification of Solutions

Key results from linear algebra:

- A unique solution $\{X\} = \left[A^{-1}\right] \cdot \{B\}$ exists when $\left[A^{-1}\right]$ exists (i.e., $\det[A] \neq 0$).
- The equations are inconsistent when $[A]$ is singular and $\operatorname{rank}[A|B] \neq \operatorname{rank}[A]$.
- If $\operatorname{rank}[A|B]$ equals $\operatorname{rank}[A]$, then there are an infinite number of solutions.

Numerical Solution of Matrix Equations in Python

Problem Statement: Consider the matrix equations:

$$\begin{bmatrix} 3 & -6 & 7 \\ 9 & 0 & -5 \\ 5 & -8 & 6 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ -4 \end{bmatrix} \tag{4}$$

Theoretical Considerations:

- A unique solution $\{X\} = \left[A^{-1}\right] \cdot \{B\}$ exists when $\left[A^{-1}\right]$ exists (i.e., $\det[A] \neq 0$). Expanding det(A) about the first row gives:

$$\begin{aligned} det(A) &= 3\det\begin{bmatrix} 0 & -5 \\ -8 & 6 \end{bmatrix} + 6\det\begin{bmatrix} 9 & -5 \\ 5 & 6 \end{bmatrix} + 7\det\begin{bmatrix} 9 & 0 \\ 5 & -8 \end{bmatrix}, \\ &= 3(0-40) + 6(54+25) + 7(-72-0) = -150. \end{aligned} \tag{5}$$

Numerical Solution of Matrix Equations in Python

Program Source Code:

```
1   # ================================================================
2   # TestMatrixEquations01.py: Compute solution to matrix equations.
3   #
4   # Written by: Mark Austin                                November 2022
5   # ================================================================
6
7   import numpy as np
8   from numpy.linalg import matrix_rank
9
10  # Function to print two-dimensional matrices ...
11
12  def PrintMatrix(name, a):
13      print("Matrix: {:s} ".format(name) );
14      for row in a:
15          for col in row:
16              print("{:8.3f}".format(col), end=" ")
17          print("")
18
19  # main method ...
20
21  def main():
22      print("--- Enter TestMatrixEquations01.main()      ... ");
23      print("--- ======================================== ... ");
24
25      print("--- Part 1: Create test matrices ... ");
```

Numerical Solution of Matrix Equations in Python

Program Source Code: Continued ...

```
27      A = np.array( [ [ 3, -6,  7],
28                      [ 9,  0, -5],
29                      [ 5, -8,  6] ]);
30      PrintMatrix("A", A);
31
32      B = np.array([ [3], [3], [-4] ]);
33      PrintMatrix("B", B);
34
35      print("--- Part 2: Check properties of matrix A ... ");
36
37      rank = matrix_rank(A)
38      det  = np.linalg.det(A)
39
40      print("--- Matrix A: rank = {:f}, det = {:f}  ...".format(rank, det) );
41
42      print("--- Part 3: Solve A.x = B ... ");
43
44      x = np.linalg.solve(A, B)
45      PrintMatrix("x", x);
46
47      print("--- ======================================== ... ");
48      print("--- Leave TestMatrixEquations01.main()      ... ");
49
50  # call the main method ...
51
52  main()
```

Numerical Solution of Matrix Equations in Python

Abbreviated Output:

```
# Part 1: Create test matrices ...

Matrix: A
   3.000   -6.000    7.000
   9.000    0.000   -5.000
   5.000   -8.000    6.000

Matrix: B
   3.000
   3.000
  -4.000

# Part 2: Check properties of matrix A ...

Matrix A: rank = 3.000000, det = -150.000000  ...
```

```
# Part 3: Solve A.x = B ...

Matrix: x
   2.000
   4.000
   3.000
```

Analysis of Linear Structural Systems

Analysis of Linear Structural Systems

General Computational Procedure:

Let matrix equations $AX = B$ represent behavior of a structural system:

- Matrix A will capture the geometry, material properties, etc.
- Matrix B represents externally applied loads (e.g., dead/live gravity loads).
- Column vector X represents nodal displacements.

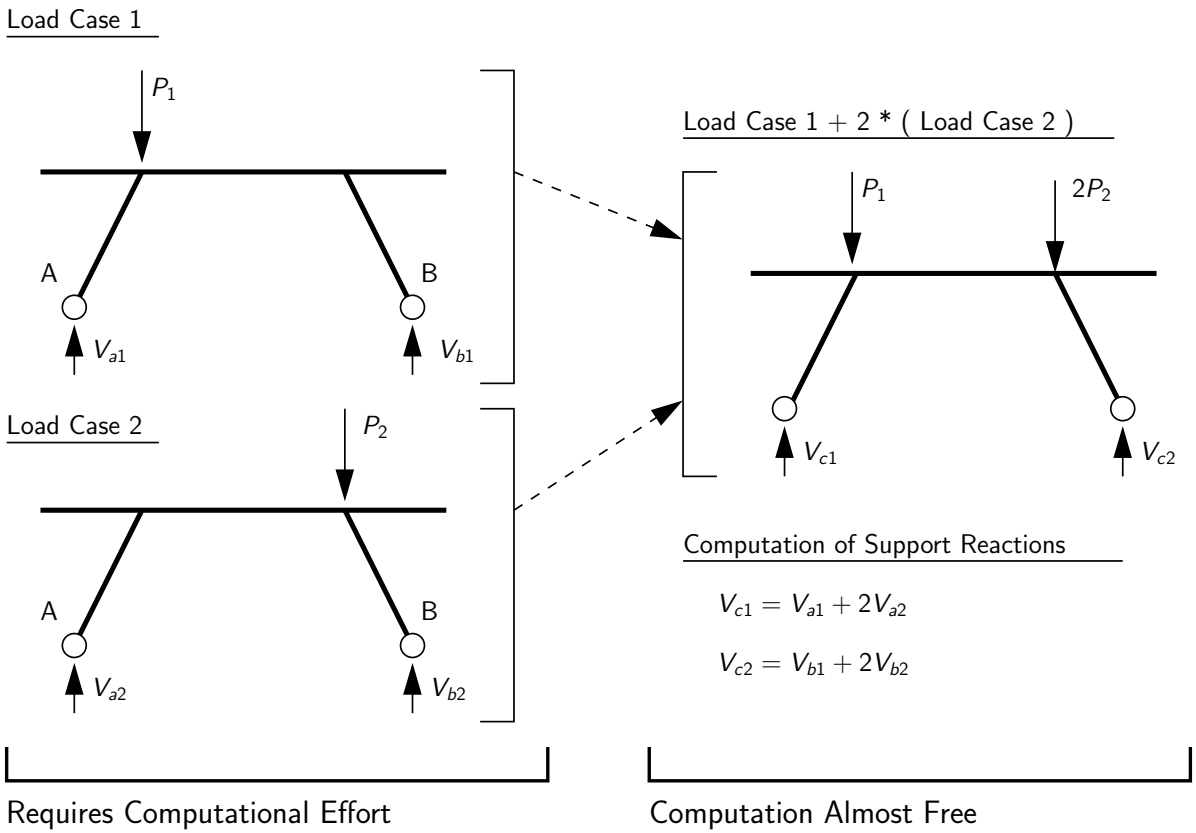
Solving $AX = B$ requires computational work $O(n^3)$.

However, if matrix system is linear, then:

$$AX_1 = B_1, AX_2 = B_2 \rightarrow A(mX_1 + kX_2) = mB_1 + kB_2. \quad (6)$$

Definition of Linear

We can simply add the results of multiple load cases:



Works for support reactions, bending moments, displacements, etc.

Simplified Computational Procedure

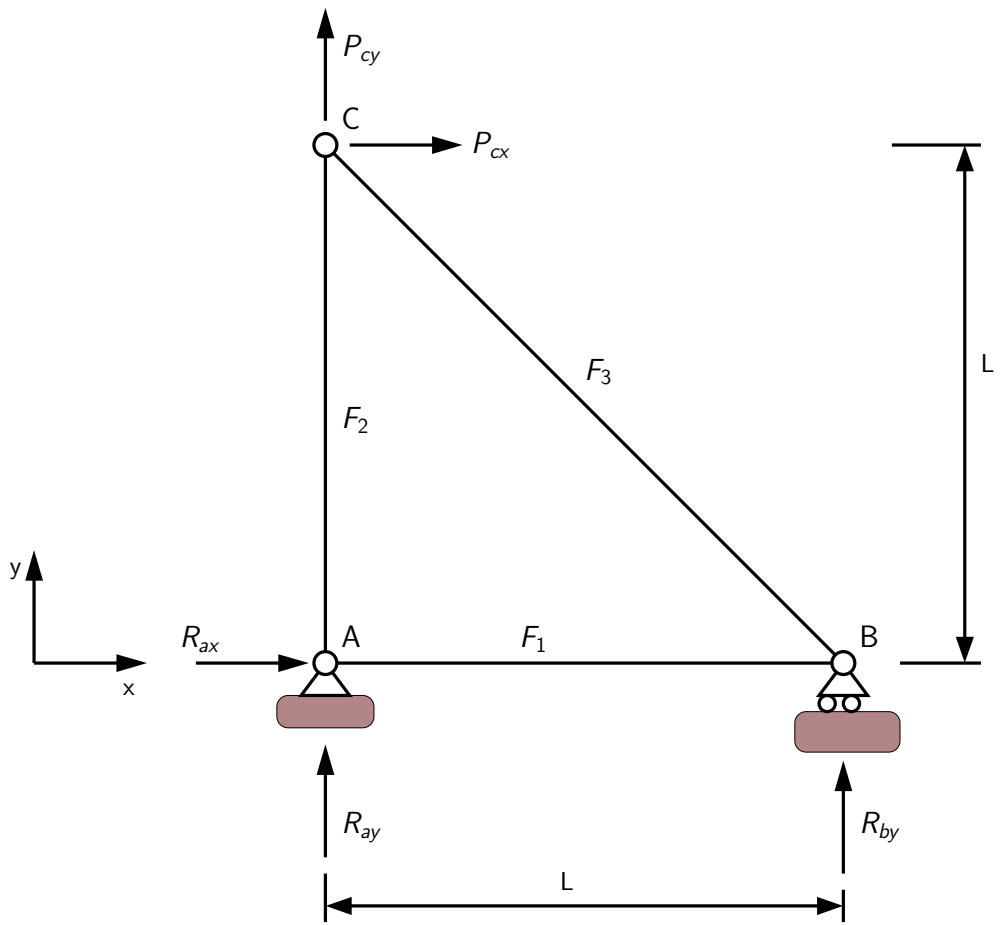
Simplified Computational Procedure:

- Use method of joints to write equations of equilibrium at each node.
- Organize the equations so that unknowns are on the left-hand side of equations; known values on the right.
- Put equations in matrix format.
- Use Python to compute solutions: $[A]\,[X] = [B]$.
- Extract and print reaction forces and element level forces.

Working Examples

Example 1. Analysis of a Three-Bar Truss

Problem Statement: We wish to compute the support reactions and element-level forces in a three-bar truss:



Example 1. Analysis of a Three-Bar Truss

Equations of Equlibrium: Node A:

$$\sum F_{ax} = 0 \rightarrow F_1 + R_{ax} = 0, \sum F_{ay} = 0 \rightarrow F_2 + R_{ay} = 0. \quad (7)$$

Node B:

$$\sum F_{bx} = 0 \rightarrow F_1 + \frac{F_3}{\sqrt{2}} = 0, \sum F_{by} = 0 \rightarrow \frac{F_3}{\sqrt{2}} + R_{by} = 0. \quad (8)$$

Node C:

$$\sum F_{cx} = 0 \rightarrow \frac{-F_3}{\sqrt{2}} = P_{cx}, \sum F_{cy} = 0 \rightarrow F_2 + \frac{F_3}{\sqrt{2}} = P_{cy}. \quad (9)$$

Example 1. Analysis of a Three-Bar Truss

Program Source Code: ...

```
1   # ================================================================
2   # TestTrussAnalysis01.py: Compute distribution of element forces
3   # and support reactions in a small three-bar truss.
4   #
5   # Written by: Mark Austin                                November 2023
6   # ================================================================
7
8   import math
9   import numpy as np
10  from numpy.linalg import matrix_rank
11
12  # ================================================================
13  # Function to print two-dimensional matrices ...
14  # ================================================================
15
16  def PrintMatrix(name, a):
17      print("Matrix: {:s} ".format(name) );
18      for row in a:
19          for col in row:
20              print("{:8.4f}".format(col), end=" ")
21          print("")
22
23  # ================================================================
24  # main method ...
25  # ================================================================
26
27  def main():
```

Example 1. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
28      print("--- Part 1: Initialize coefficients for matrix equations ... ");
29
30      # Node A ...
31
32      a11 =  1       # < --- equilibrium in x direction ...
33      a14 =  1
34      a22 =  1       # < --- equilibrium in y direction ...
35      a25 =  1
36
37      # Node B ...
38
39      a31 = 1                # < --- equilibrium in x direction ...
40      a33 = 1/math.sqrt(2)
41      a43 = 1/math.sqrt(2) # < --- equilibrium in y direction ...
42      a46 = 1
43
44      # Node C ...
45
46      a53 = -1/math.sqrt(2)   # < --- equilibrium in x direction ...
47      a62 =  1                # < --- equilibrium in y direction ...
48      a63 =  1/math.sqrt(2)
49
50      # Load Vector B ...
51
52      print("--- Part 2: Initialize load vector ... ");
53
54      b5 = 10
55      b6 =  0
```

Example 1. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
57      print("--- Part 3: Create test matrices ... ");
58
59      A = np.array([ [ a11,   0,   0, a14,   0,   0 ],
60                     [   0, a22,   0,   0, a25,   0 ],
61                     [ a31,   0, a33,   0,   0,   0 ],
62                     [   0,   0, a43,   0,   0, a46 ],
63                     [   0,   0, a53,   0,   0,   0 ],
64                     [   0, a62, a63,   0,   0,   0 ] ]);
65      PrintMatrix("A", A);
66
67      B = np.array([ [0], [0], [0], [0], [b5], [b6] ]);
68      PrintMatrix("B", B);
69
70      print("--- Part 4: Check properties of matrix A ... ");
71
72      rank = matrix_rank(A)
73      det  = np.linalg.det(A)
74
75      print("--- Matrix A: rank = {:f}, det = {:f}  ...".format(rank, det) );
76
77      print("--- Part 5: Solve A.x = B ... ");
78
79      x = np.linalg.solve(A, B)
80      PrintMatrix("x", x);
```

Example 1. Analysis of a Three-Bar Truss

Program Source Code: Continued ...

```
86        print("--- Part 6: Print support reactions and element-level forces ... ");
87
88        print("--- Support A: R_ax = {:7.2f} ... ".format( x[3][0] ) );
89        print("---           : R_ay = {:7.2f} ... ".format( x[4][0] ) );
90        print("--- Support B: R_by = {:7.2f} ... ".format( x[5][0] ) );
91
92        print("--- Element A-B: F1 = {:7.2f} ... ".format( x[0][0] ) );
93        print("--- Element A-C: F2 = {:7.2f} ... ".format( x[1][0] ) );
94        print("--- Element B-C: F3 = {:7.2f} ... ".format( x[2][0] ) );
95
96    # call the main method ...
97
98    main()
```

Example 1. Analysis of a Three-Bar Truss

Abbreviated Output:

```
--- Part 1: Initialize coefficients for matrix equations ...
--- Part 2: Initialize load vector ...
--- Part 3: Create test matrices ...

Matrix: A
  1.0000   0.0000   0.0000   1.0000   0.0000   0.0000
  0.0000   1.0000   0.0000   0.0000   1.0000   0.0000
  1.0000   0.0000   0.7071   0.0000   0.0000   0.0000
  0.0000   0.0000   0.7071   0.0000   0.0000   1.0000
  0.0000   0.0000  -0.7071   0.0000   0.0000   0.0000
  0.0000   1.0000   0.7071   0.0000   0.0000   0.0000

Matrix: B
  0.0000
  0.0000
  0.0000
  0.0000
 10.0000
  0.0000
```

Example 1. Analysis of a Three-Bar Truss

Abbreviated Output: Continued ...

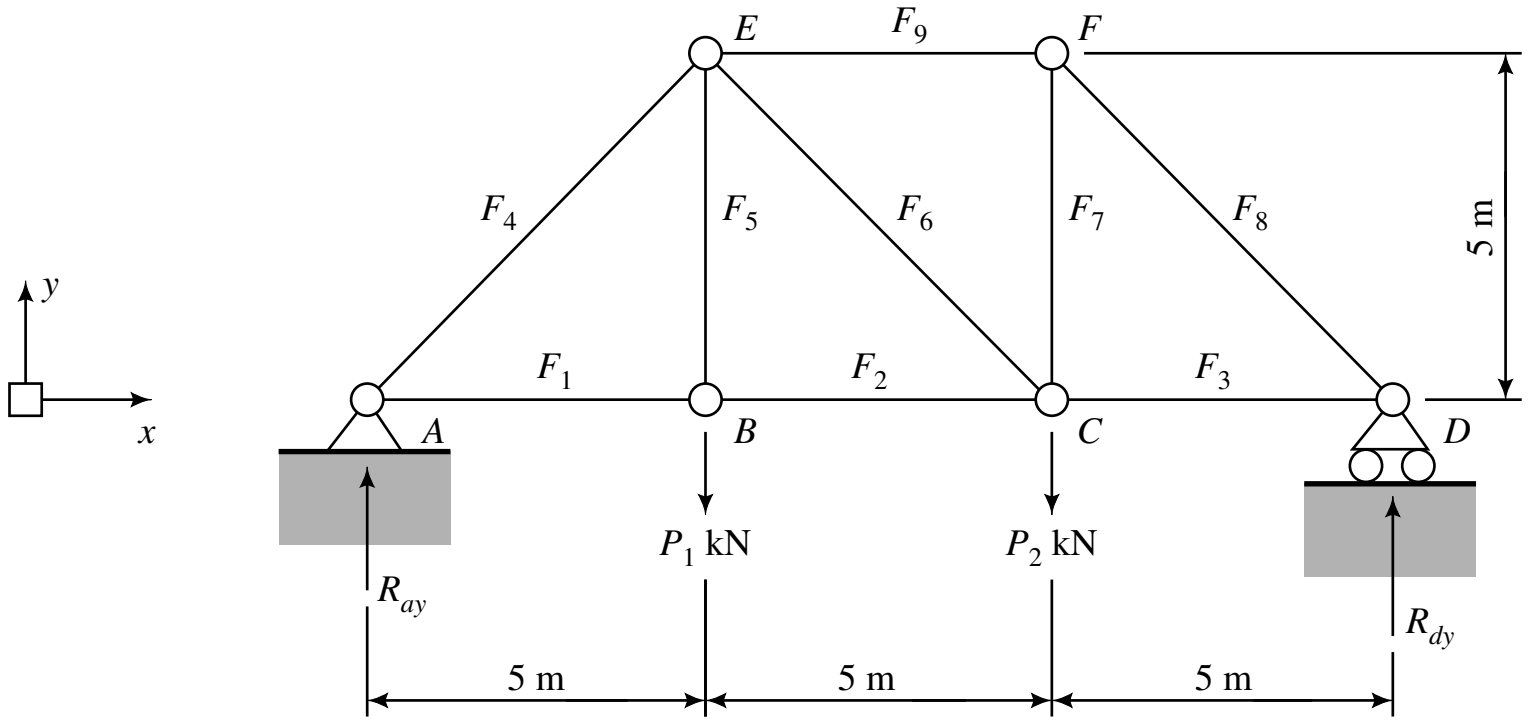
```
--- Part 4: Check properties of matrix A ...
--- Matrix A: rank = 6.000000, det = 0.707107  ...
--- Part 5: Solve A.x = B ...

Matrix: x
 10.0000
 10.0000
-14.1421
-10.0000
-10.0000
 10.0000

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

--- Support A: R_ax =  -10.00       --- Element A-B: F1 =   10.00
---          : R_ay =  -10.00       --- Element A-C: F2 =   10.00
--- Support B: R_by =   10.00       --- Element B-C: F3 =  -14.14
```

Example 2. Pin-Jointed Bridge Truss



External Load Vectors:

$$\begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \quad \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix} \qquad (10)$$

Example 2. Pin-Jointed Bridge Truss

Equations of Equilibrium:

At Joint A:

$$\sum F_x = 0, \qquad \frac{F_4}{\sqrt{2}} + F_1 = 0. \tag{11}$$

$$\sum F_y = 0, \qquad \frac{F_4}{\sqrt{2}} + R_{ay} = 0. \tag{12}$$

At Joint B:

$$\sum F_x = 0, \qquad F_1 - F_2 = 0. \tag{13}$$

$$\sum F_y = 0, \qquad F_5 - P_1 = 0. \tag{14}$$

Example 2. Pin-Jointed Bridge Truss

Equations of Equilibrium:

At Joint C:

$$\sum F_x = 0, \qquad F_2 - F_3 + \frac{F_6}{\sqrt{2}} = 0. \tag{15}$$

$$\sum F_y = 0, \qquad \frac{F_6}{\sqrt{2}} + F_7 - P_2 = 0. \tag{16}$$

At Joint D:

$$\sum F_x = 0, \qquad F_3 + \frac{F_8}{\sqrt{2}} = 0. \tag{17}$$

$$\sum F_y = 0, \qquad \frac{F_8}{\sqrt{2}} + R_{dy} = 0. \tag{18}$$

Example 2. Pin-Jointed Bridge Truss

Equations of Equilibrium:

At Joint E:

$$\sum F_x = 0, \qquad -\frac{F_4}{\sqrt{2}} + \frac{F_6}{\sqrt{2}} + F_9 = 0. \tag{19}$$

$$\sum F_y = 0, \qquad -\frac{F_4}{\sqrt{2}} - F_5 - \frac{F_6}{\sqrt{2}} = 0. \tag{20}$$

At Joint F:

$$\sum F_x = 0, \qquad \frac{F_8}{\sqrt{2}} - F_9 = 0. \tag{21}$$

$$\sum F_y = 0, \qquad -\frac{F_7}{\sqrt{2}} - \frac{F_8}{\sqrt{2}} = 0. \tag{22}$$

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
1   # ================================================================
2   # TestTrussAnalysis02.py: Compute distribution of element forces
3   # and support reactions in a nine-bar truss.
4   #
5   # Written by: Mark Austin                                November 2023
6   # ================================================================
7
8   import math
9   import numpy as np
10  from numpy.linalg import matrix_rank
11
12  # ================================================================
13  # Function to print one- and two-dimensional matrices ...
14  # ================================================================
15
16  def PrintMatrix(name, matrix):
17      NoColumns = 6;
18
19      ... details of PrintMatrix removed ...
20
21  # ================================================================
22  # Compute maximum and minumun of three numbers ...
23  # ================================================================
24
25  def maximum(a, b, c):
26      list = [a, b, c]
27      return max(list)
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
28
29  def minimum(a, b, c):
30      list = [a, b, c]
31      return min(list)
32
33  # ================================================================
34  # Print element forces ...
35  # ================================================================
36
37  def printElementForces(name, minF, maxF):
38      if( minF < 0):
39          print("---     Minimum {:s} = {:7.2f} (C) ... ".format( name, minF ) )
40      else:
41          print("---     Minimum {:s} = {:7.2f} (T) ... ".format( name, minF ) )
42
43      if( maxF < 0):
44          print("---     Maximum {:s} = {:7.2f} (C) ... ".format( name, maxF ) )
45      else:
46          print("---     Maximum {:s} = {:7.2f} (T) ... ".format( name, maxF ) )
47
48  # ================================================================
49  # main method ...
50  # ================================================================
51
52  def main():
53      print("--- Part 1: Initialize coefficients for matrix equations ... ");
54
55      # Node A ...
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
57      a11  = 1                # <--- equilibrium in x direction ...
58      a14  = 1/math.sqrt(2)
59      a110 = 1
60      a24  = 1/math.sqrt(2)   # <--- equilibrium in y direction ...
61      a211 = 1
62
63      # Node B ...
64
65      a31  =  1               # <--- equilibrium in x direction ...
66      a32  = -1
67      a45  = -1               # <--- equilibrium in y direction ...
68
69      # Node C ...
70
71      ... details of element matrix initialization removed ...
72
73      # Node D ...
74
75      ... details of element matrix initialization removed ...
76
77      # Node E ...
78
79      ... details of element matrix initialization removed ...
80
81      # Node F ...
82
83      ... details of element matrix initialization removed ...
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
85      print("--- Part 2: Create test matrix A ... ");
86
87      A = np.array([ [ a11,   0,   0,  a14,    0,   ... a110,    0,    0 ],
88                     [   0,   0,   0,  a24,    0,   ...    0, a211,    0 ],
89                     [ a31, a32,   0,    0,    0,   ...    0,    0,    0 ],
90                     [   0,   0,   0,    0,  a45,   ...    0,    0,    0 ],
91                     [   0, a52, a53,    0,    0,   ...    0,    0,    0 ],
92                     [   0,   0,   0,    0,    0,   ...    0,    0,    0 ],
93                     [   0,   0, a73,    0,    0,   ...    0,    0,    0 ],
94                     [   0,   0,   0,    0,    0,   ...    0,    0, a812 ],
95                     [   0,   0,   0,  a94,    0,   ...    0,    0,    0 ],
96                     [   0,   0,   0, a104, a105,   ...    0,    0,    0 ],
97                     [   0,   0,   0,    0,    0,   ...    0,    0,    0 ],
98                     [   0,   0,   0,    0,    0,   ...    0,    0,    0 ] ]);
99      PrintMatrix("A", A);
100
101     print("--- Part 2: Initialize load vectors ... ");
102     print("--- Load Case 1: b4 = -10, b6 = 0 ...");
103
104     b4 = -10.0; b6 = 0.0
105     B1 = np.array([ [0], [0], [0], [b4],  [0], [b6], [0], [0], [0], [0], [0], [0] ]);
106     PrintMatrix("Load Case 1: B", B1);
107
108     print("--- Load Case 2: b4 = -5, b6 = -5 ...");
109
110     ... details removed ...
111
112     print("--- Load Case 3: b4 = 0, b6 = -10 ...");
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
113
114     ... details removed ...
115
116     print("--- Part 4: Check properties of matrix A ... ");
117
118     rank = matrix_rank(A)
119     det  = np.linalg.det(A)
120
121     print("--- Matrix A: rank = {:f}, det = {:f}  ...".format(rank, det) );
122
123     print("--- Part 5: Solve A.F = B for three load cases ... ");
124
125     F1 = np.linalg.solve(A, B1)
126     PrintMatrix("Load Case 1: Forces ...", F1);
127
128     ... details removed ...
129
130     print("--- Part 6: Print support reactions and element-level forces ... ");
131
132     print("--- ");
133     print("--- Support Reactions and Element-Level Forces: Load Case 1:");
134     print("--- ");
135     print("---   Reaction A: R_ax = {:7.2f} ... ".format( F1[9][0] ) );
136     print("---              : R_ay = {:7.2f} ... ".format( F1[10][0] ) );
137     print("---   Reaction D: R_dy = {:7.2f} ... ".format( F1[11][0] ) );
138     print("--- ");
139     print("--- Element Level Forces:");
140     print("--- ");
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
141     print("---   Element A-B: F1 = {:7.2f} ... ".format( F1[0][0] ) );
142     print("---   Element B-C: F2 = {:7.2f} ... ".format( F1[1][0] ) );
143     print("---   Element C-D: F3 = {:7.2f} ... ".format( F1[2][0] ) );
144     print("---   Element A-E: F4 = {:7.2f} ... ".format( F1[3][0] ) );
145     print("---   Element B-E: F5 = {:7.2f} ... ".format( F1[4][0] ) );
146     print("---   Element C-E: F6 = {:7.2f} ... ".format( F1[5][0] ) );
147     print("---   Element C-F: F7 = {:7.2f} ... ".format( F1[6][0] ) );
148     print("---   Element D-F: F8 = {:7.2f} ... ".format( F1[7][0] ) );
149     print("---   Element E-F: F9 = {:7.2f} ... ".format( F1[8][0] ) );
150
151     print("--- Support Reactions and Element-Level Forces: Load Case 2:");
152
153     ... details removed ...
154
155     print("--- Support Reactions and Element-Level Forces: Load Case 3:");
156
157     ... details removed ...
158
159     print("--- Summary of Max/Min Reactions and Element-Level Forces ...");
160
161     MinRax = minimum( F1[9][0], F2[9][0], F3[9][0] )
162     MaxRax = maximum( F1[9][0], F2[9][0], F3[9][0] )
163
164     ... details removed ...
165
166     print("---   Reaction A: Minimum R_ax = {:7.2f}, Maximum R_ax = {:7.2f} ... ".format
167     print("---               : Minimum R_ay = {:7.2f}, Maximum R_ay = {:7.2f} ... ".format
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Python Source Code:

```
168         print("--- ");
169         print("---   Reaction D: Minimum R_dy = {:7.2f}, Maximum R_dy = {:7.2f} ... ".format
170
171         MinF1 = minimum( F1[0][0], F2[0][0], F3[0][0] )
172
173         ... details removed ...
174
175         print("---   Element A-B: ")
176         printElementForces( "F1", MinF1, MaxF1)
177
178         print("---   Element B-C: ")
179         printElementForces( "F2", MinF2, MaxF2)
180
181         ... details removed ...
182
183         print("--- ======================================== ... ");
184         print("--- Leave TestTrussAnalysis02.main()         ... ");
185
186     # call the main method ...
187
188     main()
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Program Output:

```
Matrix: A
row/col          1             2             3             4             5             6
   1   1.0000e+00  0.0000e+00 0.0000e+00 7.0710e-01 0.0000e+00 0.0000e+00
   2   0.0000e+00  0.0000e+00 0.0000e+00 7.0710e-01 0.0000e+00 0.0000e+00
   3   1.0000e+00 -1.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00

... output removed ...

  12   0.0000e+00  0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00

Matrix: A
row/col          7             8             9            10            11            12
   1   0.0000e+00  0.0000e+00 0.0000e+00 1.0000e+00 0.0000e+00 0.0000e+00
   2   0.0000e+00  0.0000e+00 0.0000e+00 0.0000e+00 1.0000e+00 0.0000e+00
   3   0.0000e+00  0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00

... output removed ...

  12   1.0000e+00  7.0710e-01 0.0000e+00 0.0000e+00 0.0000e+00 0.0000e+00
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Program Output:

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

Load Case 1:  b4 = -10    Load Case 2:  b4 =  -5   Load Case 3:  b4 =   0
              b6 =   0                  b6 =  -5                 b6 = -10

Matrix: Load Case 1: B    Matrix: Load Case 2: B   Matrix: Load Case 3: B
row/col              1    row/col              1   row/col              1
   1       0.00000e+00       1       0.00000e+00      1       0.00000e+00
   2       0.00000e+00       2       0.00000e+00      2       0.00000e+00
   3       0.00000e+00       3       0.00000e+00      3       0.00000e+00
   4      -1.00000e+01       4      -5.00000e+00      4      -0.00000e+00
   5       0.00000e+00       5       0.00000e+00      5       0.00000e+00
   6       0.00000e+00       6      -5.00000e+00      6      -1.00000e+01
   7       0.00000e+00       7       0.00000e+00      7       0.00000e+00
   8       0.00000e+00       8       0.00000e+00      8       0.00000e+00
   9       0.00000e+00       9       0.00000e+00      9       0.00000e+00
  10       0.00000e+00      10       0.00000e+00     10       0.00000e+00
  11       0.00000e+00      11       0.00000e+00     11       0.00000e+00
  12       0.00000e+00      12       0.00000e+00     12       0.00000e+00
```

Example 2. Pin-Jointed Bridge Truss

Abbreviated Program Output:

```
--- 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: Case 1: Forces   Matrix: Case 2: Forces    Matrix: Case 3: Forces
row/col                1 row/col                1 row/col                1
   1       6.66667e+00      1       5.00000e+00      1       3.33333e+00
   2       6.66667e+00      2       5.00000e+00      2       3.33333e+00

... output removed ...   ... output removed ...    ... output removed ...

  11       6.66667e+00     11       5.00000e+00     11       3.33333e+00
  12       3.33333e+00     12       5.00000e+00     12       6.66667e+00
```

Example 2. Pin-Jointed Bridge Truss

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