[redacted]

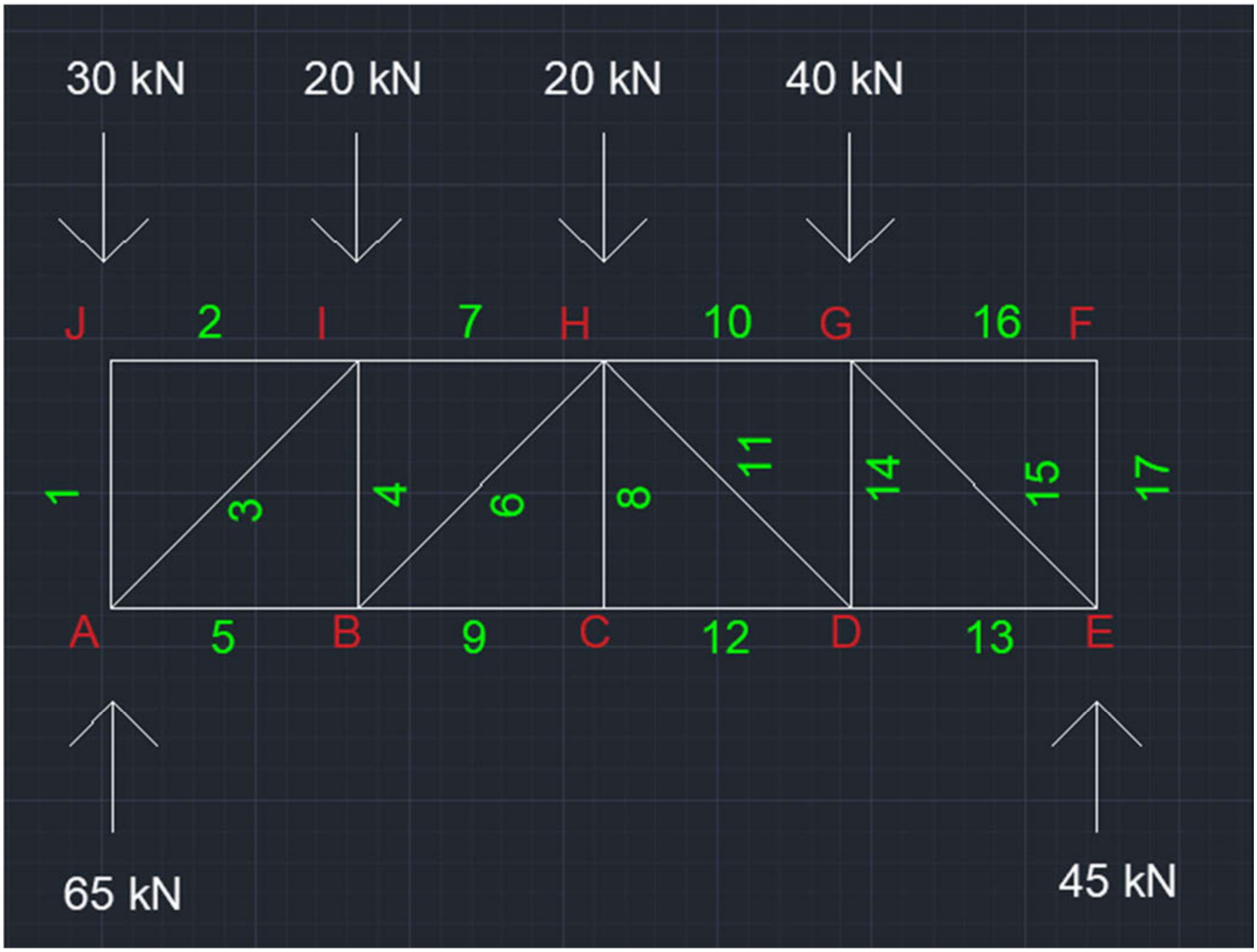
EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21

Introduction

The Howe Truss is a classic statics problem to demonstrate the distribution of the forces acting on the members of a truss when an external load is applied to it, to remain intact in static equilibrium where the sum of the forces in both x and y direction are equal to zero. It is a classic problem to demonstrate how to sum vector forces in the x and y direction, and how it applies in a real life scenario.

Problem Statement

Given the parameters of the truss pictured below, we need to determine the forces acting on each member of the truss, determine whether each member is in tension or compression, and which members are under the greatest stress in order to design a safe truss with materials of adequate strength to withstand the stresses.

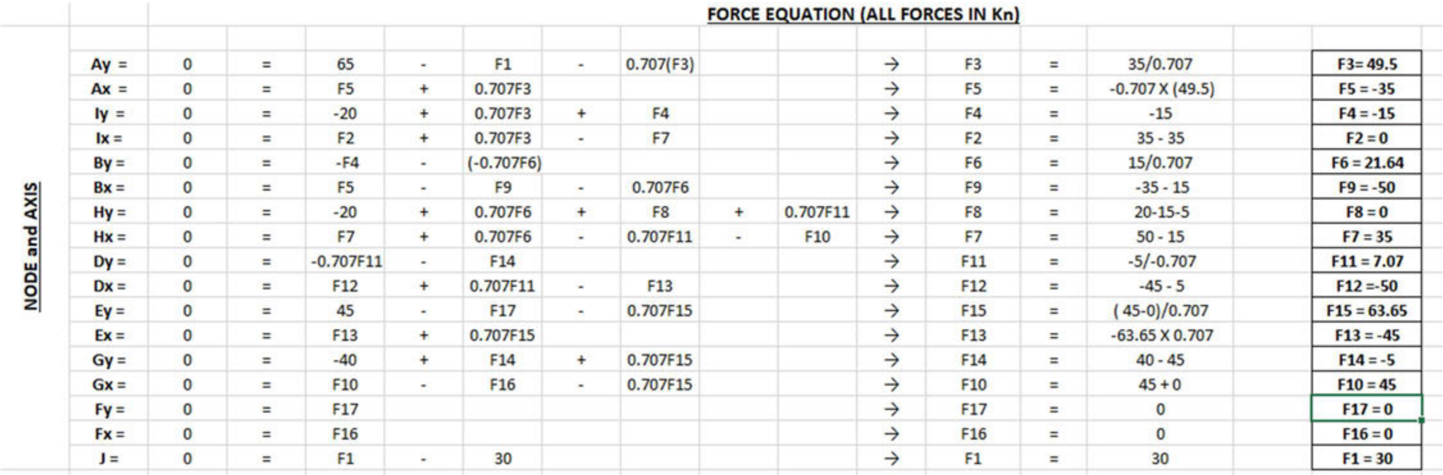


[redacted]

EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21

Methodology

There are various ways to solve this dilemma. As displayed in the diagram above, there are 17 members in the truss, which intersect at 10 different nodes (letters on the diagram). Each node is acted on by one or more member in the x and y direction. Each member represents a variable we need to solve for, which in turn requires 17 equations to eliminate one variable at a time. This can be done by hand by breaking each node into an x and y component, setting the force equal to zero, and solving for the force vectors acting on them one at a time. This method is demonstrated in the photo below. It does not matter the number you assign to each member, only that you keep track of which node each member is acting on, and which direction its force is acting on, and remaining consistent with the +/- sign you utilize for direction. Vertical and horizontal members are multiplied by a factor of 1 and diagonal members by a factor of 0.707, since they are at a 45 degree angle and the respective axial forces are equivalent to sin(45) and cos(45), which both equal 0.707.

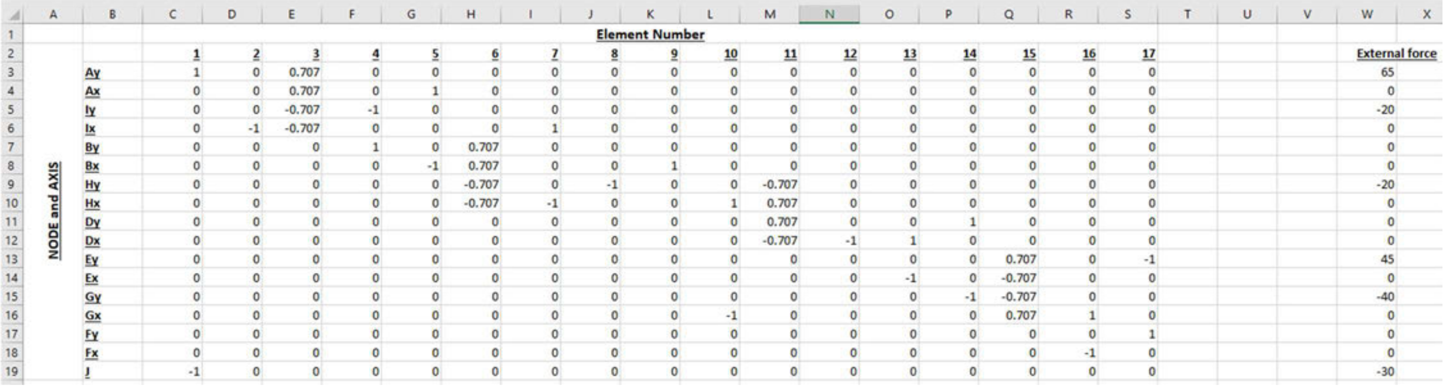


								FORCE EQUATION (ALL FORCES IN Kn)								
NODE and AXIS	**Ay =**	0	=	65	-	F1	-	0.707(F3)			→	F3	=	35/0.707		**F3= 49.5**
	Ax =	0	=	F5	+	0.707F3					→	F5	=	-0.707 X (49.5)		**F5 = -35**
	Iy =	0	=	-20	+	0.707F3	+	F4			→	F4	=	-15		**F4 = -15**
	Ix =	0	=	F2	+	0.707F3	-	F7			→	F2	=	35 - 35		**F2 = 0**
	By =	0	=	-F4	-	(-0.707F6)					→	F6	=	15/0.707		**F6 = 21.64**
	Bx =	0	=	F5	-	F9	-	0.707F6			→	F9	=	-35 - 15		**F9 = -50**
	Hy =	0	=	-20	+	0.707F6	+	F8	+	0.707F11	→	F8	=	20-15-5		**F8 = 0**
	Hx =	0	=	F7	+	0.707F6	-	0.707F11	-	F10	→	F7	=	50 - 15		**F7 = 35**
	Dy =	0	=	-0.707F11	-	F14					→	F11	=	-5/-0.707		**F11 = 7.07**
	Dx =	0	=	F12	+	0.707F11	-	F13			→	F12	=	-45 - 5		**F12 =-50**
	Ey =	0	=	45	-	F17	-	0.707F15			→	F15	=	(45-0)/0.707		**F15 = 63.65**
	Ex =	0	=	F13	+	0.707F15					→	F13	=	-63.65 X 0.707		**F13 = -45**
	Gy =	0	=	-40	+	F14	+	0.707F15			→	F14	=	40 - 45		**F14 = -5**
	Gx =	0	=	F10	-	F16	-	0.707F15			→	F10	=	45 + 0		**F10 = 45**
	Fy =	0	=	F17							→	F17	=	0		**F17 = 0**
	Fx =	0	=	F16							→	F16	=	0		**F16 = 0**
	J =	0	=	F1	-	30					→	F1	=	30		**F1 = 30**

However, utilizing python script and the Numpy library, this problem can be solved much more rapidly utilizing matrix math and dot product functions imbedded in the Numpy. Two separate arrays, one 17x17 representing the truss members and the Node axis, in essence the location of each member acting on a node on each axis, and the other matrix a 1x17 representing the external forces. The trick part about this, which is the same as doing it by hand, is to keep your sign convention consistent in order to get the desired results. I set up a spreadsheet in order to organize my two arrays before entering them into python, then utilized a few simple lines of code to achieve the same result as doing it by hand in a fraction of the time. See photos below.

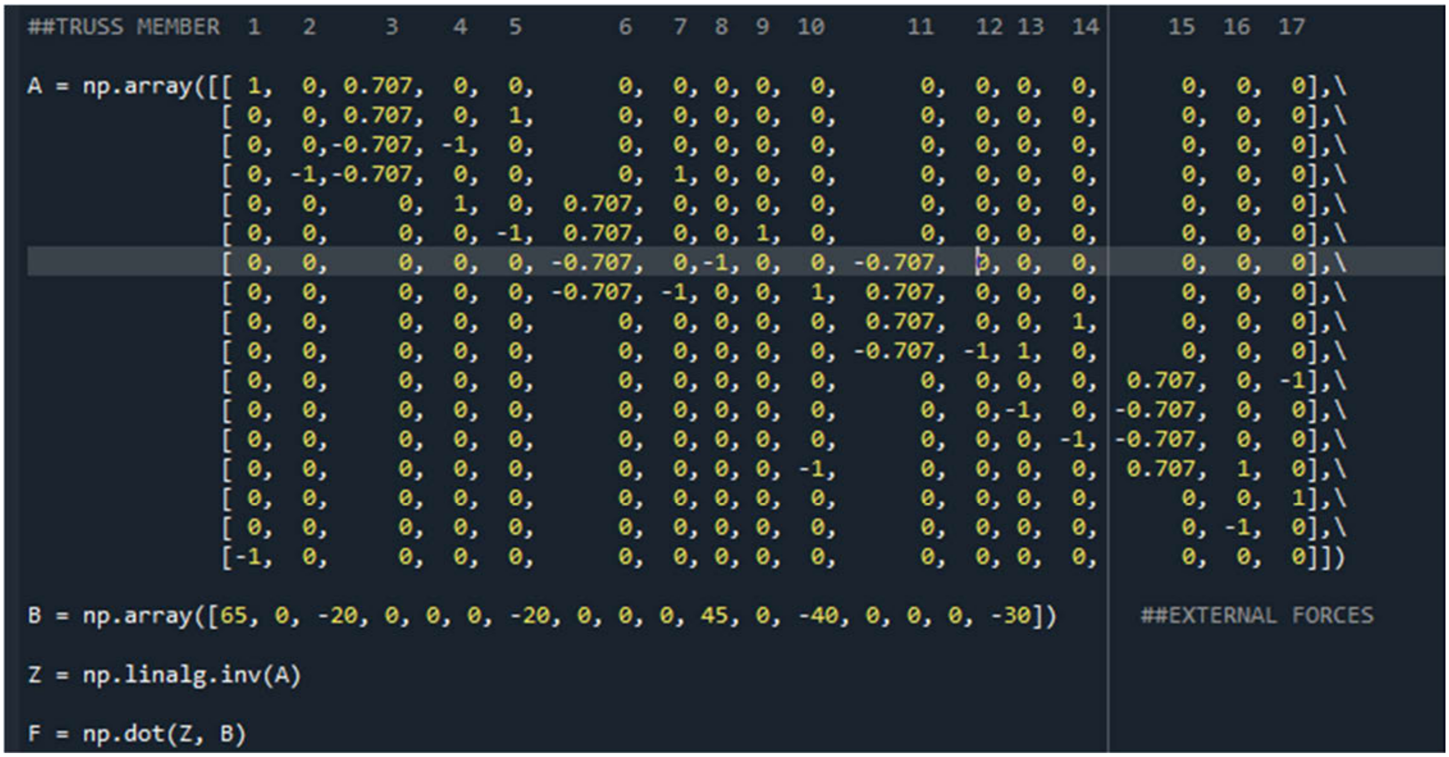
[redacted]

EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21



Element Number

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	External force
NODE and AXIS	Ay	1	0	0.707	0	0	0	0	0	0	0	0	0	0	0	0	0	0	65
	Ax	0	0	0.707	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	Iy	0	0	-0.707	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	-20
	Ix	0	-1	-0.707	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	By	0	0	0	1	0	0.707	0	0	0	0	0	0	0	0	0	0	0	0
	Bx	0	0	0	0	-1	0.707	0	0	1	0	0	0	0	0	0	0	0	0
	Hy	0	0	0	0	0	-0.707	0	-1	0	0	-0.707	0	0	0	0	0	0	-20
	Hx	0	0	0	0	0	-0.707	-1	0	0	1	0.707	0	0	0	0	0	0	0
	Dy	0	0	0	0	0	0	0	0	0	0	0.707	0	0	1	0	0	0	0
	Dx	0	0	0	0	0	0	0	0	0	0	-0.707	-1	1	0	0	0	0	0
	Ey	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.707	0	-1	45
	Ex	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	-0.707	0	0	0
	Gy	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	-0.707	0	0	-40
	Gx	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0.707	1	0	0
	Fy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	Fx	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
	J	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-30



```
##TRUSS MEMBER  1   2     3    4   5       6   7  8  9  10      11    12 13  14      15  16  17

A = np.array([[ 1,  0, 0.707,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0,  0, 0.707,  0,  1,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,-0.707, -1,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0, -1,-0.707,  0,  0,      0,  1, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,     0,  1,  0,  0.707,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,     0,  0, -1,  0.707,  0, 0, 1,  0,      0,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,     0,  0,  0, -0.707,  0,-1, 0,  0, -0.707,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,     0,  0,  0, -0.707, -1, 0, 0,  1,  0.707,  0, 0,  0,      0,  0,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,  0.707,  0, 0,  1,      0,  0,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0, -0.707, -1, 1,  0,      0,  0,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,  0.707,  0, -1],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0,-1,  0, -0.707,  0,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0, -1, -0.707,  0,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0, -1,      0,  0, 0,  0,  0.707,  1,  0],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  1],\
              [ 0,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0, -1,  0],\
              [-1,  0,     0,  0,  0,      0,  0, 0, 0,  0,      0,  0, 0,  0,      0,  0,  0]])

B = np.array([65, 0, -20, 0, 0, 0, -20, 0, 0, 0, 45, 0, -40, 0, 0, 0, -30])      ##EXTERNAL FORCES

Z = np.linalg.inv(A)

F = np.dot(Z, B)
```

[redacted]

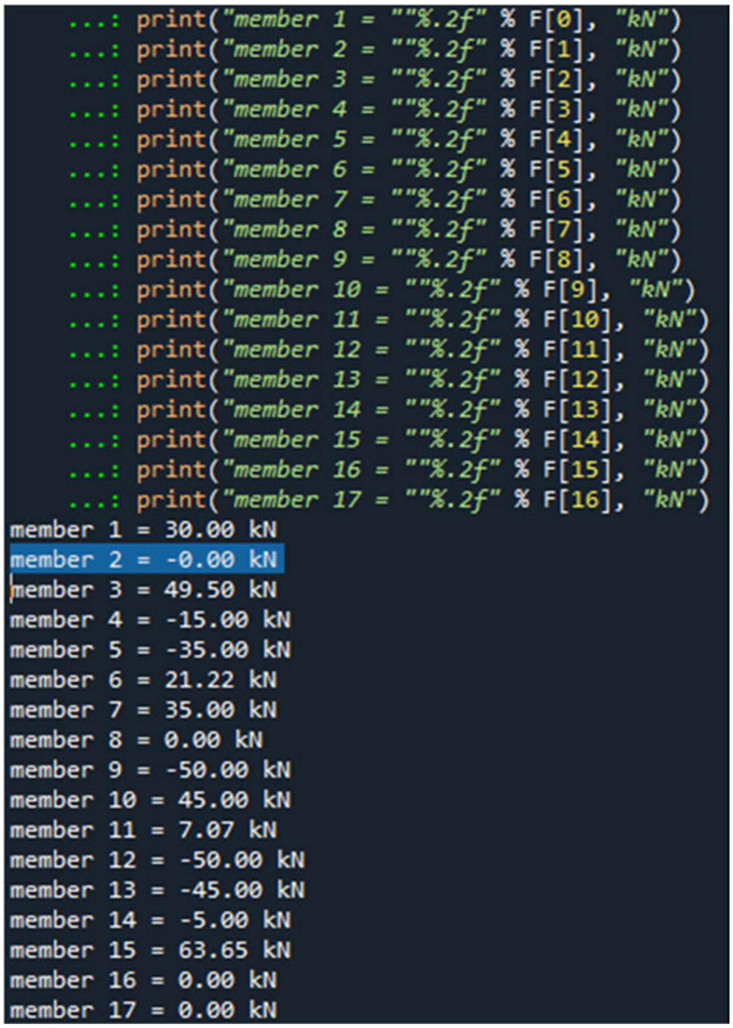
EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21

Solution

By inputting the values into the matrix and executing the function listed, the following results were obtained. Results with positive numbers are under compression, while negative numbers represent members in tension.

Member 15 (EG) is under the most compression, approximately 63.65kN.

Members 9 (BC) and 12 (CD) are under the greatest tension, at approximately -50kN.



```
    ...: print("member 1 = ""%.2f" % F[0], "kN")
    ...: print("member 2 = ""%.2f" % F[1], "kN")
    ...: print("member 3 = ""%.2f" % F[2], "kN")
    ...: print("member 4 = ""%.2f" % F[3], "kN")
    ...: print("member 5 = ""%.2f" % F[4], "kN")
    ...: print("member 6 = ""%.2f" % F[5], "kN")
    ...: print("member 7 = ""%.2f" % F[6], "kN")
    ...: print("member 8 = ""%.2f" % F[7], "kN")
    ...: print("member 9 = ""%.2f" % F[8], "kN")
    ...: print("member 10 = ""%.2f" % F[9], "kN")
    ...: print("member 11 = ""%.2f" % F[10], "kN")
    ...: print("member 12 = ""%.2f" % F[11], "kN")
    ...: print("member 13 = ""%.2f" % F[12], "kN")
    ...: print("member 14 = ""%.2f" % F[13], "kN")
    ...: print("member 15 = ""%.2f" % F[14], "kN")
    ...: print("member 16 = ""%.2f" % F[15], "kN")
    ...: print("member 17 = ""%.2f" % F[16], "kN")
member 1 = 30.00 kN
member 2 = -0.00 kN
member 3 = 49.50 kN
member 4 = -15.00 kN
member 5 = -35.00 kN
member 6 = 21.22 kN
member 7 = 35.00 kN
member 8 = 0.00 kN
member 9 = -50.00 kN
member 10 = 45.00 kN
member 11 = 7.07 kN
member 12 = -50.00 kN
member 13 = -45.00 kN
member 14 = -5.00 kN
member 15 = 63.65 kN
member 16 = 0.00 kN
member 17 = 0.00 kN
```

Photo demonstrates the results from python given the matrix input values.

[redacted]
EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21

Conclusion

Coding and using Python and the Spyder editor takes time to get used to but can be tremendously useful in solving any number of complex calculations. Through use of a few simple lines of code, proper set up of arrays, and the use of Numpy, one of the many build in libraries, a complex and tedious problem can be solved rapidly, greatly reducing work time and increasing efficiency. This method has many real world functions and can be incredibly useful in the engineering and construction design fields. I look forward to further exploring the capabilities of utilizing good coding and the many applications it provides.

Appendices

The following code was used for the program:

```
@author: [redacted]
"""
## This function will demonstrate the classic Howe Truss problem solution
## using the NUMPY library to do matrix math. Columns represent truss members,
## rows represent node values at x and y axis of junctions. A value of +/- 1
## represents a horizontal or vertical member, a value of +/- 0.707 represents a diagonal
## member, since the members are at 45 degree angle, both sin(45) and cos(45)
## are equal to 0.707.


import numpy as np
```

[redacted]

EGN 3214- Programming for Engineers
Assignment 6- Matrix Problem- Howe Truss
10/22/21

```
##TRUSS MEMBER  1   2     3    4   5        6   7  8  9  10       11    12 13  14
15  16  17

A = np.array([[ 1,  0, 0.707,  0,  0,       0,  0, 0, 0,  0,       0,   0, 0, 0,
0,  0,  0],\

              [ 0,  0, 0.707,  0,  1,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,-0.707, -1,  0,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0,  0,  0],\

              [ 0, -1,-0.707,  0,  0,       0,  1, 0, 0,  0,       0,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,      0,  1,  0,  0.707,  0, 0, 0,  0,       0,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,      0,  0, -1,  0.707,  0, 0, 1,  0,       0,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,      0,  0,  0, -0.707,  0,-1, 0,  0, -0.707,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,      0,  0,  0, -0.707, -1, 0, 0,  1,  0.707,  0, 0,  0,
0,  0,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,  0.707,  0, 0,  1,
0,  0,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0, -0.707, -1, 1,  0,
0,  0,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0.707,  0, -1],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0,-1,  0, -
0.707,  0,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0, 0, -1, -
0.707,  0,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0, -1,       0,  0, 0,  0,
0.707,  1,  0],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0,  0,  1],\

              [ 0,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0, -1,  0],\

              [-1,  0,      0,  0,  0,       0,  0, 0, 0,  0,       0,  0, 0,  0,
0,  0,  0]])
```

```
B = np.array([65, 0, -20, 0, 0, 0, -20, 0, 0, 0, 45, 0, -40, 0, 0, 0, -30])
##EXTERNAL FORCES


Z = np.linalg.inv(A)


F = np.dot(Z, B)


print("member 1 = ""%.2f" % F[0], "kN")

print("member 2 = ""%.2f" % F[1], "kN")

print("member 3 = ""%.2f" % F[2], "kN")

print("member 4 = ""%.2f" % F[3], "kN")

print("member 5 = ""%.2f" % F[4], "kN")

print("member 6 = ""%.2f" % F[5], "kN")

print("member 7 = ""%.2f" % F[6], "kN")

print("member 8 = ""%.2f" % F[7], "kN")

print("member 9 = ""%.2f" % F[8], "kN")

print("member 10 = ""%.2f" % F[9], "kN")

print("member 11 = ""%.2f" % F[10], "kN")

print("member 12 = ""%.2f" % F[11], "kN")

print("member 13 = ""%.2f" % F[12], "kN")

print("member 14 = ""%.2f" % F[13], "kN")

print("member 15 = ""%.2f" % F[14], "kN")

print("member 16 = ""%.2f" % F[15], "kN")

print("member 17 = ""%.2f" % F[16], "kN")
```