Howe Truss Problem

# Abstract

The purpose of this exercise is to calculate a classic engineering problem; The Howe Truss Problem.

# Problem Statement

Many times throughout or engineering career we will be asked to calculate classic problems over and over again. Simply because they still apply to modern applications. We are tasked to calculate the force vectors on all major nodes of the following problem, fig 1. The equation to do this is listed below.

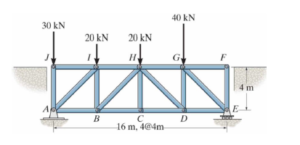


Figure : Howe Truss Problem

# Methodology

Utilizing excel, python, and a few libraries for python. Namely pandas and numpy we can make short work of the problem. First we separate all the truss members into nodes. We then create a matrix (x) of all the nodes with either: 0, 1, -1, 0.7071, -0.7071. We will utilize these across all nodes. We then take the forces and make a separate array (y). With numpy and the linear algebra function we take the main array (x) and apply the function turning x into x\_inv. Then using matrix calculation we apply our main array (x\_inv) and our forces array(y). Thus calculating all force vectors on all the different nodes getting the answers (z).

'''

Chris Oakley

Arrays and Matrices

10/3/2020

EGN3214 - Assignment 6

Variables:

a - Force Vector Names

x - Main Array before inverse

x\_inv - Inverse Main Array

y - Force vectors on Nodes

z - Final Output with Force Vectors and Nodes

'''

import numpy as np

import matplotlib.pyplot as plt

y = np.array([65, 0, -20, 0, 0, 0, -20, 0, 0, 0, 45, 0, -40 , 0, 0, 0, -30])

a = np.array(['AJ', 'JI', 'AI', 'BI', 'AB', 'BH', 'HI', 'CH', 'BC', 'HG', 'HD', 'CD', 'DE', 'DG', 'EG', 'GF', 'EF'])

x = 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],

]

)

x\_inv = np.linalg.inv(x)

z = np.dot(x\_inv,y)

# print(z)

for a, z in zip(a,z):

print(f'{a}: {z:.1f}')

# Solution

The following assignment variables are provided below within the table. Also the most compression comes from node EG at -63.7 kN, and the Most Tension comes from nodes BC and CD at 50.00 kN.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AJ | JI | AI | BI | AB | BH | HI | CH | BC | HG | HD | CD | DE | DG | EG | GF | EF | Forces | Force Vectors  (nodes) |
| AJ | -1 | 0 | -0.707 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 65 | -30.0 |
| JI | 0 | 0 | -0.707 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| AI | 0 | 0 | 0.707 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -20 | -49.5 |
| BI | 0 | 1 | 0.707 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 15.0 |
| AB | 0 | 0 | 0 | -1 | 0 | -0.707 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 35.0 |
| BH | 0 | 0 | 0 | 0 | 1 | -0.707 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -21.2 |
| HI | 0 | 0 | 0 | 0 | 0 | 0.707 | 0 | 1 | 0 | 0 | -0.707 | 0 | 0 | 0 | 0 | 0 | 0 | -20 | -35.0 |
| CH | 0 | 0 | 0 | 0 | 0 | 0.707 | 1 | 0 | 0 | -1 | -0.707 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.0 |
| BC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.707 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 50.0 |
| HG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.707 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | -45.0 |
| HD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -0.707 | 0 | -1 | 45 | -7.1 |
| CD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0.707 | 0 | 0 | 0 | 50.0 |
| DE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0.707 | 0 | 0 | -40 | 45.0 |
| DG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -0.707 | -1 | 0 | 0 | 5.0 |
| EG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | -63.7 |
| GF | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0.0 |
| EF | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -30 | 0.0 |

Table : Nodes with All Calculations

# Conclusion

The force vectors at the nodes were calculated out using linear algebra matrix math. This method allows for quick calculations across the entire problem.