Kory R. Schultz

22 October 2025

EGN3214

Assignment 6 - Matrix Problem

**INTRODUCTION**

The purpose of this exercise is to create a Python code that is capable of calculating maximum tension and compression of a specified truss. This code should apply this functionality to the given truss under Assignment 6’s description. It should perform these calculations by importing a CSV file with force modifiers and external forces and performing a cross product of matrices.

The Colab program can be found at <https://colab.research.google.com/drive/1R1X-9mg_O7hH11L4v5LoIr4vARAiEFIH?usp=sharing>

This Colab project also imports the CSV file via the use of a Github link (no additional files should need to be imported).

**PROBLEM STATEMENT**

Using the following depiction of a truss:

|  |
| --- |

Find the 12 formulas and 3 reactionary force multipliers (i.e. sin(45)) for the truss. Using these values, along with the two external forces, create a matrix within Excel in the form of a csv file that can be imported into a Python file. Upon importing this file, process this data in two arrays using the module Numpy, and calculate the cross product of these arrays in order to determine the X and Y forces for each node and reaction forces. Then determine which values return the maximum compression and maximum tension of the truss.

**METHODOLOGY**

Using Colab, I began by importing the URL of the CSV file from Github as well as the modules Numpy and Pandas. Then I read the CSV file using Numpy and process the CSV into two separate arrays: a 2-D array containing all of the force multipliers for the X and Y forces, and a 1-D array containing only the external forces.

| df = pd.read\_csv(url)  arrSize = df.columns.size - 2  forceMult = np.zeros((arrSize, arrSize))  extForces = df[df.columns[-1:]]  for i in range(0, arrSize):  forceMult[i] = df.iloc[i][1:-1] |
| --- |

Afterwards, I calculate the forces based on the cross product of the two arrays (matrices) and store this data within a separate numpy array.

| invA = np.linalg.inv(forceMult)  invDot = invA @ extForces  invDotSolution = np.array(invDot) |
| --- |

I then create the set-up for what will be the final output of the data. I create an array with each header that will be used (i.e. “F(AB)”), and make variables for the maximum tension, maximum compression, and the indices for each.

| memberNames = ["F(AB)", "F(AC)", "F(AD)", "F(BD)", "F(CE)", "F(CD)", "F(DF)", "F(DE)", "F(EF)", "R(Ax)", "R(Bx)", "R(By)"]  maxTension = 0  maxTensionIteration = 0  maxCompression = 0  maxCompressionIteration = 0 |
| --- |

Finally, the values are then run through a loop to determine the maximum compression, maximum tension, and the indices for each. This loop also formats the numbers into strings that are adjusted for a user-friendly, readable output. While I’m sure there is an easier way to accomplish this, I was encountering errors attempting to format these values while still in integer and float forms as a part of the numpy arrays. This method of string formatting still gives the correct output and results, but it is not exactly practical should this code need to be reused with other data sets and would likely error or display incorrect information.

| for i, name in enumerate(memberNames):  val = str(invDotSolution[i])  val = val[1:-1]  if val.find("."):  val = val + "0"  val = val[:val.find(".") + 3]  if float(val) < maxTension:  maxTension = float(val)  maxTensionIteration = i  if float(val) > maxCompression:  maxCompression = float(val)  maxCompressionIteration = i  print(f"{name} = {val} kN") |
| --- |

**RESULTS**

| F(AB) = -90.0 kN  F(AC) = 90.0 kN  F(AD) = 127.28 kN  F(BD) = -220.0 kN  F(CE) = 90.0 kN  F(CD) = 0.0 kN  F(DF) = -40.0 kN  F(DE) = -127.28 kN  F(EF) = 0.0 kN  R(Ax) = -180.0 kN  R(Bx) = 220.0 kN  R(By) = -90.0 kN  Max Compression: 220.0 at R(Bx)  MaxTension: -220.0 at F(BD) |
| --- |

**CONCLUSION**

This code successfully determines the maximum compression and tension of the truss given within the assignment description. It also successfully uses matrix multiplication to determine the forces acting upon each node of the truss using an importing CSV file containing the force multipliers as a matrix. Additional coding would need to be done to improve the reusability of the code in the case that csv files with different formatting were presented and needed to undergo calculations as well. However, for the sake of this example, this code succeeds in determining the correct values required.

**APPENDICES**

| url = "https://raw.githubusercontent.com/schultzk-dsc/matrix/main/file.csv"  import numpy as np  import pandas as pd  df = pd.read\_csv(url)  arrSize = df.columns.size - 2  forceMult = np.zeros((arrSize, arrSize))  extForces = df[df.columns[-1:]]  for i in range(0, arrSize):  forceMult[i] = df.iloc[i][1:-1]  invA = np.linalg.inv(forceMult)  invDot = invA @ extForces  invDotSolution = np.array(invDot)  memberNames = ["F(AB)", "F(AC)", "F(AD)", "F(BD)", "F(CE)", "F(CD)", "F(DF)", "F(DE)", "F(EF)", "R(Ax)", "R(Bx)", "R(By)"]  maxTension = 0  maxTensionIteration = 0  maxCompression = 0  maxCompressionIteration = 0  val = "0"  for i, name in enumerate(memberNames):  val = str(invDotSolution[i])  val = val[1:-1]  if val.find("."):  val = val + "0"  val = val[:val.find(".") + 3]  if float(val) < maxTension:  maxTension = float(val)  maxTensionIteration = i  if float(val) > maxCompression:  maxCompression = float(val)  maxCompressionIteration = i  print(f"{name} = {val} kN")  print(f"Max Compression: {maxCompression} at {memberNames[maxCompressionIteration]}")  print(f"MaxTension: {maxTension} at {memberNames[maxTensionIteration]}") |
| --- |