VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF APPLIED SCIENCE

THEORY OF MECHANICS (AS1003)

Report project

Determine stresses in the members of the planar truss problem

Mentor:	Assoc. Prof.PhD Thien Truong Tich
Members:	Minh Pham Duc
	Minh Tran Gia
	Tam Nguyen Phuc Minh

HO CHI MINH CITY, NOVEMBER 2024

Table of contents

1 Member list & Workload

Num	Full name	ID	Work	Rate
1	Phạm Đức Minh	2352759	Leader, code	100%
2	Trần Gia Minh	2312118	Theoretical basis, Translate	100%
3	Nguyễn Phúc Minh Tâm	2313033	Write LaTeX report, Algorithm, PowerPoint	100%

2 Introduction

First at all, our group would like to extend our heartfelt gratitude to teacher Trương Tích Thiện for his extremely insightful and dedicated lectures, as well as to teacher Lồ Sìu Vẫy for his engaging exercise sessions, and to the faculty members who have supported us throughout the course **Theory of Mechanics**. Through the knowledge gained from this course, we have developed a deeper understanding of *Classical Mechanic*s in general and the field of *Engineering Mechanics* in particular, which we have applied to completing this project.

Our group wanted to give our utmost effort not only to achieve the highest possible score for the project but also to compensate for lower scores in other areas, such as the midterm, where our performance was less than ideal. Thus, we decided to approach this assignment with the highest level of commitment. This included preparing a report, coding, and delivering a presentation all in English and composing the report using LaTeX.

The report includes Theoretical basis in Section 3, Algorithms, instructions for running and using the code in Section 4.1, and the code in Section 4.2. The main content of the report focuses on using Python code to Determine stresses in the members of the planar truss problem using Method of Joint.

As for why the code was written in Python, it is because we were taught this programming language in the Programming for Engineering course by teacher Nhã. We would also like to take this opportunity to express our gratitude to teacher Nhã for the valuable knowledge he imparted.

3 Theoretical basis

3.1 Method of Joints

3.1.1 Definition

The Method of Joints is a process used to solve for the unknown forces acting on the members of a truss system. This method focuses on the joints, which are the connections between the members.

In this context, a member is a straight structural element connecting the joints in the truss system. A truss is a structure consisting of members connected at the joints.

Specifically, the joints are considered as particles subjected to forces due to the connections between the truss members and the applied loads. Since the joints are always in a state of equilibrium, it follows two equations:

$$\sum F = 0 \quad (1)$$
$$\sum M = 0 \quad (2)$$

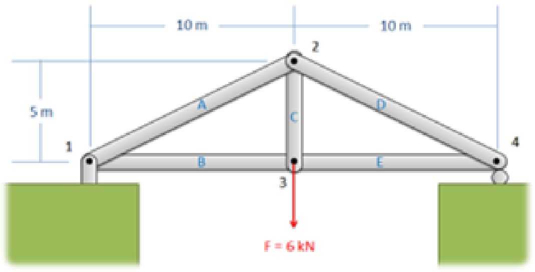
For a planar truss, we will apply equation (1) to the two coordinate axes Ox and Oy, and combine it with equation (2):

$$\sum F_x = 0$$
$$\sum F_y = 0$$

From there, we can determine the two unknown forces required in the problem. Therefore, we can find one or two unknown forces, and the condition for solving is that at least one known force must be acting at that point.

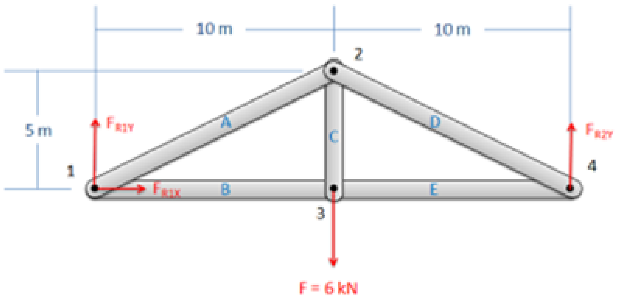
3.1.2 Procedure for applying Method of Joint

Step 1: A truss system consists of many members; therefore, we assign numbers or letters to avoid errors during the process of determining the required forces..



Hình 1: The first step is labeling each joints and each members

Step 2: We will release the system's constraints (also known as supports) and draw the corresponding forces (also known as reactions) on the supports through the diagram. (Free body diagram).

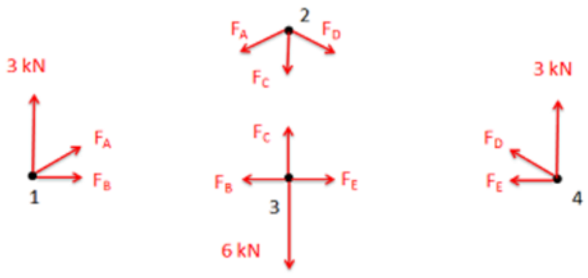


Hình 2: Consider the entire truss as a rigid body and find reaction forces

Step 3: We will apply the equilibrium equations to the system in order to solve for the unknown forces.

$$\sum F_x = 0$$
$$\sum F_y = 0$$
$$\sum M = 0$$

This is an example of how to draw the forces on the diagram (FBD):



3.2 Inverse of the Matrix

3.2.1 Definition

Inverse of the Matrix is n-by-n square matrix which has an inverse. A is called invertible if there exists an n-by-n square matrix B such that:

$$AB = BA = I$$

For example: give two Matrixs

$$A = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}$$
$$AB = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$
$$BA = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I$$

So, B is invertible matrix of A.

Invertible matrix of A is:

$$A^{-1} = B = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}$$

An invertible matrix is a non-singular matrix. Conversely, a non-invertible matrix is a singular matrix.

3.2.2 Properties

The inverse matrix has several common properties. Given matrices A and B are invertible, and $k \neq 0$ is a scalar. Then, AB, kA and A^{-1} are also invertible matrices.

$$\begin{aligned}(AB)^{-1} &= B^{-1}.A^{-1}\\ (kA)^{-1} &= \frac{1}{k}.A^{-1}, k \neq 0\\ (A^{-1})^{-1} &= A\\ (A^{T})^{-1} &= (A^{-1})^{T}\end{aligned}$$

3.3 Using Code

Numpy library

Function	The use of function
np.zeros()	Create an array filled with 0's.
np.linalg.pinv()	Compute the pseudo-inverse of a matrix.
np.dot()	Dot product of two arrays.
np.set_printoptions()	Set printing options.

Pandas library

Function	The use of function
pd.read_excel("test.xlsx", "Joint")	read data at sheet "Joint" from excel file "test.xlsx".
a.iterrows()	Iterate over a DataFrame rows as (index, Series) pairs.
row['Joint']	In value sheet, row['Joint'] read value in sheet 'Joint'.

Math library

Function	The use of function
math.atan2(y, x)	Return atan(y / x), in radians. The result is between -pi and pi. The vector in the plane from the origin to point (x, y) makes this angle with the positive X axis. The point of atan2() is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle.
math.cos(a)	Return the cosine of a radians.
math.sin(a)	Return the sine of aa radians.

Matplotlib library

Function	The use of function
plt.subplots()	Create a figure containing a single Axes.
ax.plot()	Plot some data on the (implicit) Axes.
ax.text()	Write text to the local space in plot
ax.quiver()	Draw vector on the axes
ax.scatter([],[],color='#fe6d7a',label='Tension')	Write note 'Tension' for axes have #hex corlor code '#fe6d7a'.
ax.legend()	Add a legend.
ax.set_xlabel()	Add an x-label to the Axes.
ax.set_ylabel()	Add an y-label to the Axes.
ax.set_title()	Add an title to the Axes.
ax.grid()	Turn On / Off grid.
ax.axis("equal")	Set the retro of axis equal.
plt.show()	Show the plot.

Others

Function	The use of function
i.count(a)	Return the number of times the value "a" appears in the i list.
zip(a, b)	Join two tuples a & b together.
len()	Return the number of items in a list, tuple, array,...
list()	Create a list containing names.

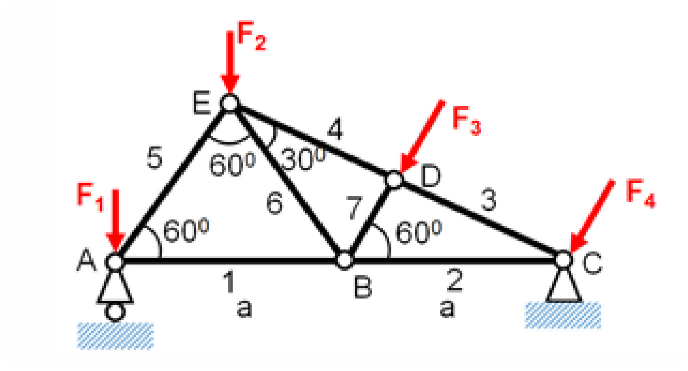
Dictionary

Function	The use of function
a.values()	Return a list to values a.
a.enumerate()	Convert a tuple into an enumerate object a.
a.items()	Return the dictionary's key-value pairs a.

4 Using Python code to solve problems

4.1 Algorithm

With the given problem: Determine the stresses in the members of the planar truss described by the following diagram:



Given: External forces applied to the joints have magnitudes $F_1 = F_2 = F_3 = F_4 = 100N$, and the length of members $AB = BC = a$

Solution

Step 1: Enter the coordinates of each joint into the "Joint" sheet. The following table is obtained:

Joint	X	Y
1	0	0
2	1	0
3	2	0
4	1.25	0.433013
5	0.5	0.866025

Step 2: Enter the F_x, F_y of the external forces acting on the corresponding joints into the "Force" sheet. Additionally, input the supports for each joint into the same sheet. If a joint does not move along the $x - direction$, input: $R_x = 1$; otherwise, $R_x = 0$. Similarly, if a joint does not move along the $y - direction$, input $R_y = 1$; otherwise, $R_y = 0$. The resulting table is as follows:

Joint	R_x	R_y	F_x	F_y
1	0	1	0	-100
2	0	0	0	-100
3	1	1	-50	-86.6025
4	0	0	-50	-86.6025
5	0	0	0	-100

Step 3: Since a member is defined by two joints, input the members of this truss system by entering one joint in the "Start" column and the other joint in the "End" column. Repeat this process for the remaining members. The resulting table is as follows:

Start	End
1	2
5	4
4	3
3	2
1	5
2	5
2	4

Step 4: Solve the system of equations:

$$A.S = -F$$

Where:

A: the matrix containing the coefficients of S

S: the vector of forces (stresses) in the members to be determined

F: the matrix of external forces

Find matrix A

Matrix A: $2b \times (k + r)$

Where:

b: the number of joints

k: the number of members

r: the number of supports

Start by considering member $(1,2)$ or member AB, which is the first member analyzed, so $i=0$. For joint $A(0,0), B(1,0)$ calculate the angle α, which is the angle between vector $\vec{AB}$ and the positive $x-axis$. The α is expressed in radians. The calculation is as follows:

$$\vec{AB} = (1,0)$$
$$\cos\alpha = \frac{\vec{AB}.\vec{i}}{|\vec{AB}|.|\vec{i}|} = \frac{1}{1.1} = 1$$
$$=> \alpha = 0$$

To determine the A_{mn}

Where:

$$n = 0, 1, ..., 2b-1$$
$$m = 0, 1, .., k+r-1$$

We do the following:

Consider Joint 1:

With $n = 2\times(nut-1) = 2\times(1-1) = 0$, $m = i = 0$, we get: $A_{00} = \cos\alpha = \cos 0 = 1$

With $n = 2\times(nut-1)+1 = 2\times(1-1)+1 = 1$, $m = i = 0$, we get: $A_{10} = \sin\alpha = \sin 0 = 0$

Consider Joint 2:

With $n = 2\times(nut-1) = 2\times(2-1) = 2$, $m = i = 0$, we get: $A_{20} = -\cos\alpha = -\cos 0 = -1$

With $n = 2\times(nut-1)+1 = 2\times(2-1)+1 = 3$, $m = i = 0$, we get: $A_{30} = -\sin\alpha = -\sin 0 = 0$

Similarly, for the remaining bars, we obtain the matrix A as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.87 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0.87 & 0.87 & 0 & 0 & 0 \\ 0 & 0 & -0.87 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.87 & 0.87 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0.5 & -0.5 & 0 & 0 & 0 & -0.87 & 0 & 0 & 0 \\ 0 & 0.87 & 0 & 0 & -0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & -0.87 & -0.87 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Next, we need to add the constraint connections to A, we proceed as follows:

At Joint 1: $R_x = 0$, $R_y = 1$, we get:

With $R_y = 1$:

$n = 2 \times (nut - 1) + 1 = 2 \times (1 - 1) + 1 = 1$, $m = k = 7$, we get: $A_{17} = 1$

k now increases to 8

At Joint 3: $R_x = 1$, $R_y = 1$, we get:

With $R_x = 1$:

$n = 2 \times (nut - 1) = 2 \times (3 - 1) = 4$, $m = k = 8$, we get: $A_{48} = 1$

k now increases to 9

With $R_y = 1$:

$n = 2 \times (nut - 1) + 1 = 2 \times (3 - 1) + 1 = 5$, $m = k = 9$, we get: $A_{59} = 1$

k now increases to 10

Similarly, for the other joints with supports, we obtain the matrix A as follows:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.87 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & -0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0.87 & 0.87 & 0 & 0 & 0 \\ 0 & 0 & -0.87 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -0.87 & 0.87 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0.5 & -0.5 & 0 & 0 & 0 & -0.87 & 0 & 0 & 0 \\ 0 & 0.87 & 0 & 0 & -0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & -0.87 & -0.87 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Find matrix F

Matrix F: $2b \times 1$

To determine F_{n0} with $n = 0, 1, ..., 2b - 1$, we proceed as follows:

At Joint 1: $F_x = 0$, $F_y = -100$:

With $n = 2 \times (nut - 1) = 2 \times (1 - 1) = 0$, we get: $F_{00} = F_x = 0$

With $n = 2 \times (nut - 1) + 1 = 2 \times (1 - 1) + 1 = 1$, we get: $F_{10} = F_y = -100$

Similarly, for the other joints, we obtain matrix F as follows:

$$F = \begin{bmatrix} 0 \\ -100 \\ 0 \\ 0 \\ -50 \\ -86.6 \\ -50 \\ -86.6 \\ 0 \\ -100 \end{bmatrix}$$

Finally, solve S, we get: $S = A^{-1}.(-F)$:

$$S = \begin{bmatrix} 68.3 \\ -136.6 \\ -136.6 \\ 168.3 \\ -136.6 \\ 100 \\ -100 \\ 218.3 \\ 100 \\ 154. \end{bmatrix}$$

From matrix S we have solutions:

External Forces:

Bar AB = 68.30 (Tension)

Bar ED = 136.60 (Compression)

Bar CD = 136.60 (Compression)

Bar BC = 168.30 (Tension)

Bar AE = 136.60 (Compression)

Bar BE = 100 (Tension)

Bar BD = 100 (Compression)

Reactions:

Point 1: $R_y = 218.3$

Point 3: $R_x = 100$, $R_y = 159.4$

4.2 Code

```python
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt

'''
first: add data
second: calculating
last: draw truss
'''

###---data processing---###
##---input---##
members = pd.read_excel("inputdata.xlsx", "Member")
joints = pd.read_excel("inputdata.xlsx", "Joint")
forces = pd.read_excel("inputdata.xlsx", "Force")

np.set_printoptions(precision=2, suppress=True) #round to 00 after ','

##-----analysis-----##
#-----joint-----#
joint = {}
for _, row in joints.iterrows():
    joint_coordition = int(row['Joint'])
    x = float(row['X'])
    y = float(row['Y'])
    joint[joint_coordition] = (x, y) #joint coordition

#-----force-----#
eforce = {} #external force
for _, row in forces.iterrows():
    if row['FX'] != 0 or row['FY'] != 0:
        joint_coordition = int(row['Joint'])
        fx = float(row['FX'])
        fy = float(row['FY'])
        eforce[joint_coordition] = (fx, fy)

#-----reaction-----#
reaction = {} #reaction
for _, row in forces.iterrows():
   if row['RX'] != 0 or row['RY'] != 0:
        joint_coordition = int(row['Joint'])
        rx = int(row['RX'])
        ry = int(row['RY'])
        reaction[joint_coordition] = (rx, ry)

count_reaction = 0  #count how many reaction in truss
for i in reaction.values():
    count_reaction += i.count(1)

#-----members-----#
member = list(zip(members['Start'], members['End'])) #member with the start and
    end

count_member = len(member) #counting member

'''
print('input: ')
print('eforce =', eforce)
print('reaction =', reaction)
print('joint =', joint)
print('member =', member)
'''

###-----calculating------###

#-----calculating angle at 2 joint-----#
def calculate_angle(p1, p2):
    dx = p2[0] - p1[0]
    dy = p2[1] - p1[1]
    angle = math.atan2(dy, dx)
    return angle

'''
AS = B
A is all information about any joint
```

```
F is external forces of any joint

--> S = (A^-1) F
using pinv to calculate matrix A inverse
'''

'''
calculating:
step 1: create matrix A and matrix F
step 2: add forces, reaction infor to matrix A
step 3: add external force infor to matrix F
step 4: solve S
'''

##-----create matrix-----##
m_a = np.zeros((2 * len(joint), count_member + count_reaction), dtype=float)
m_f = np.zeros((2 * len(joint), 1), dtype=float)

##-----add in4 to A-----###
#-----add force-----#
for i, (start, end) in enumerate(member):
    angle = calculate_angle(joint[start], joint[end])
    cos_theta = math.cos(angle)
    sin_theta = math.sin(angle)

    #start joint
    m_a[2 * (start - 1), i] += cos_theta
    m_a[2 * (start - 1) + 1, i] += sin_theta

    #end joint
    m_a[2 * (end - 1), i] -= cos_theta
    m_a[2 * (end - 1) + 1, i] -= sin_theta

#-----add reaction-----#
reaction_add = len(member)
for point, (r_x, r_y) in reaction.items():
    if r_x == 1:
        m_a[2 * (point - 1), reaction_add] = 1
        reaction_add += 1
    if r_y == 1:
        m_a[2 * (point - 1) + 1, reaction_add] = 1
```

```python
        reaction_add += 1

##-----add in4 to F-----##
for point, (f_x, f_y) in eforce.items():
    m_f[2 * (point - 1)][0] += f_x
    m_f[2 * (point - 1) + 1][0] += f_y

##-----solve S-----##
minv_a = np.linalg.pinv(m_a)
m_s = np.dot(minv_a, -m_f)


##-----answers-----##

print('Forces:')
for i, (start, end) in enumerate(member):
    print(f'Bar {start}-{end}: {m_s[i][0]:.2f} ({"Tension" if m_s[i][0] > 0 else "
    Compression" if m_s[i][0] < 0 else "None"})')

print('\nExternal Forces:')
for point, force in eforce.items():
    print(f'Point {point}: FX = {force[0]:.2f}, FY = {force[1]:.2f}')

print('\nReactions:')
reaction_values = m_s[count_member:]
index = 0

for point, (rx, ry) in reaction.items():
    if rx == 1:
        print(f'Point {point}: RX = {reaction_values[index][0]:.2f}')
        index += 1
    if ry == 1:
        print(f'Point {point}: RY = {reaction_values[index][0]:.2f}')
        index += 1

###-----draw-----###
fig, ax = plt.subplots()

#-----draw button-----#
for j, (x, y) in joint.items():
    ax.plot(x, y, 'o', color='black')
    ax.text(x, y, f'{j}', color='black', fontsize=16, ha='right')

#-----color-----#
def get_color(force):
    if force > 0:
        return '#fe6d7a'   #Tension
    elif force < 0:
        return '#2caddb'   #Compression
    else:
        return '#6b6b6b'   #None

#-----draw trusses-----#
for i, (start, end) in enumerate(member):

    x_coor = [joint[start][0], joint[end][0]]
    y_coor = [joint[start][1], joint[end][1]]

    ax.plot(x_coor, y_coor, color=get_color(m_s[i][0]), linewidth=2)

    #write answers
    mid_x = (joint[start][0] + joint[end][0]) / 2
    mid_y = (joint[start][1] + joint[end][1]) / 2
    ax.text(mid_x, mid_y, f'{m_s[i][0]:.2f}', fontsize=10)

#-----draw external force vectors-----#
for point, (fx, fy) in eforce.items():
    x, y = joint[point]
    # Draw FX
    if fx != 0:
        fxa = -1 if fx < 0 else 1
        ax.quiver(x, y, fxa, 0, angles='xy', scale_units='xy', scale=1, color='#4
    A26AB', width=0.005)
        ax.text(x + 0.1 * fxa, y, f'FX = {fx:.2f}', fontsize=10, ha='left')

    # Draw FY
    if fy != 0:
        fya = -1 if fy < 0 else 1
        ax.quiver(x, y, 0, fya, angles='xy', scale_units='xy', scale=1, color='#4
    A26AB', width=0.005)
        ax.text(x, y + 0.1 * fya, f'FY = {fy:.2f}', fontsize=10, ha='left')

```

```python
#-----draw reaction force vectors-----#
index = count_member

for point, (rx, ry) in reaction.items():
    x, y = joint[point]

    #check for direction of RX
    if rx == 1:

        rxa = -1 if m_s[index][0] < 0 else 1
        ax.quiver(x, y, rxa, 0, angles='xy', scale_units='xy', scale=1, color='#399407', linewidth=1.5)
        ax.text(x + 0.1 * rxa, y, f'{m_s[index][0]:.2f}', fontsize=10, ha='center', va='center')
        index += 1

    #check for direction of Ry
    if ry == 1:

        rya = -1 if m_s[index][0] < 0 else 1
        ax.quiver(x, y, 0, rya, angles='xy', scale_units='xy', scale=1, color='#399407', linewidth=1.5)
        ax.text(x, y + 0.1 * rya, f'{m_s[index][0]:.2f}', fontsize=10, ha='center', va='center')
        index += 1



#-----legend description-----#
ax.scatter([], [], color='#fe6d7a', label='Tension')
ax.scatter([], [], color='#2caddb', label='Compression')
ax.scatter([], [], color='#6b6b6b', label='None')
ax.scatter([], [], color='#4A26AB', label='External force')
ax.scatter([], [], color='#399407', label='Reaction')

ax.legend()

'''
hex corlor(description nearly):
#fe6d7a red
#2caddb blue
```

```
#6b6b6b gray
#4A26AB purple
#399407 green
'''

#-----setting-----#
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_title("Trusses using Method of joint")
ax.grid(True)
ax.axis("equal")

plt.show()
```

5 Conclusion

The code successfully meets the requirements of the problem, including accurately calculating the forces in each member and the reactions at the supports.

Improvements to the code:

Data input is facilitated via Excel files, making it easier and faster to input data and improving convenience.

The output includes a graphical representation of the truss system, with color present in each members, making data visualization and result interpretation more easy user.

References

[1] Python Software Foundation. (2001). Python. Retrieved from math — Mathematical functions: https://www.python.org/

[2] Beer, F., E. Johnston, DeWolf, J., & Mazurek, D. (2010). Statics and Mechanics of Materials (Vol. 6). New York, U.S.A: McGraw-Hill Education.

[3] Cherney, D., Denton, T., Thomas, R., & Waldron, A. (2013). Linear Algebra. California: Creative Commons.

[4] NumPy Developers. (2008). NumPy documentation. Retrieved from NumPy: the absolute basics for beginners: https://numpy.org/doc/stable/index.html

[5] The Matplotlib development team. (2012). Matplotlib 3.9.2 documentation. Retrieved from Quick start guide: https://matplotlib.org/stable/