# ENGN2020 – Midterm #1

## Problem 1

### (a) Answer:

For any mass *j*, by using Hook`s Law, the force can be written as:

$$\boldsymbol{F_j} = m_j * q_j'' = -\sum_{i=1}^{N} k_{i,j} * q_i$$

Where, $q_j''$ is the second derivative of displacement $q_j$ against time.

According to the question, $m = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 3 \\ 5 \\ 9 \end{bmatrix}$, $k = \begin{bmatrix} 8 & 3 & 7 & 1 & 0 \\ 3 & 7 & 5 & 0 & 1 \\ 7 & 5 & 9 & 3 & 1 \\ 1 & 0 & 3 & 5 & 3 \\ 0 & 1 & 1 & 3 & 10 \end{bmatrix}$, and the equation above can be rewritten as :

$$q_j'' = -\sum_{i=1}^{N} \frac{k_{i,j}}{m_j} * q_i$$

This is a typical eigenvalue problem: $\boldsymbol{A \times q} = \lambda \boldsymbol{q}$, where $\lambda = \omega^2$, and $\omega$ are the natural frequencies of this system.

In this problem:

$$A = -\sum_{j=1}^{5} \sum_{i=1}^{5} \frac{k_{i,j}}{m_j}$$

Therefore, $A = \begin{bmatrix} 1 & 0.375 & 0.875 & 0.125 & 0 \\ 1.5 & 3.5 & 2.5 & 0 & 0.5 \\ 2.333 & 1.667 & 3 & 1 & 0.333 \\ 0.2 & 0 & 0.6 & 1 & 1 \\ 0 & 0.111 & 0.111 & 0.333 & 1.1111 \end{bmatrix}$

Solving this eigenvalue problem:

Eigenvalues are $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{bmatrix} = \begin{bmatrix} 5.8783 \\ 0.0874 \\ 0.5156 \\ 1.8248 \\ 1.3050 \end{bmatrix}$ $(\mathrm{Hz}^2)$, and $\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \\ \omega_5 \end{bmatrix} = \begin{bmatrix} \pm\ 2.4245i \\ \pm\ 0.2956i \\ \pm\ 0.7180i \\ \pm\ 1.3508i \\ \pm\ 1.1424i \end{bmatrix}$ (Hz)

The corresponding eigenvalues are:

$q_1 = \begin{bmatrix} -0.1713 \\ -0.7632 \\ -0.6156 \\ -0.0874 \\ -0.0383 \end{bmatrix}$(m), $q_2 = \begin{bmatrix} -0.4846 \\ -0.3122 \\ 0.6984 \\ -0.4140 \\ 0.0929 \end{bmatrix}$(m), $q_3 = \begin{bmatrix} -0.4852 \\ 0.2768 \\ 0.05078 \\ 0.6949 \\ -0.4500 \end{bmatrix}$(m), $q_4 = \begin{bmatrix} -0.1431 \\ 0.7770 \\ -0.4057 \\ -0.4358 \\ -0.1457 \end{bmatrix}$(m), and $q_5 = \begin{bmatrix} -0.3171 \\ 0.4891 \\ -0.3678 \\ 0.3337 \\ 0.6431 \end{bmatrix}$ (m)

### (b) Answer:

According to (a), the natural movements matrix *N* is:

$$N = \begin{bmatrix} -0.1713 & -0.4846 & -0.4852 & -0.1431 & -0.3171 \\ -0.7632 & -0.3122 & 0.2768 & 0.7770 & 0.4891 \\ -0.6156 & 0.6984 & 0.05078 & -0.4057 & -0.3678 \\ -0.0874 & -0.4140 & 0.6949 & -0.4358 & 0.3337 \\ -0.0383 & 0.0929 & -0.4500 & -0.1457 & 0.6431 \end{bmatrix}$$

The given displacement $q = \begin{bmatrix} 0.1 \\ 0.05 \\ 0.01 \\ -0.3 \\ 0.3 \end{bmatrix}$(m), after normalizing, $q = \begin{bmatrix} 0.2278 \\ 0.1139 \\ 0.0228 \\ -0.6835 \\ 0.6835 \end{bmatrix}$.

Therefore, the position of the system, is term of the natural movements $Q$ is:

$$Q = N \times q = \begin{bmatrix} -0.2242 \\ -0.4000 \\ -0.0337 \\ 0.4748 \\ 0.5309 \end{bmatrix}$$

**(c) Answer:**

$k = \begin{bmatrix} 8 & 3 & 7 & 1 & 0 \\ 3 & 7 & 5 & 0 & 1 \\ 7 & 5 & 9 & 3 & 1 \\ 1 & 0 & 3 & 5 & 3 \\ 0 & 1 & 1 & 3 & 10 \end{bmatrix}$, and $q = \begin{bmatrix} 0.1 \\ 0.05 \\ 0.01 \\ -0.3 \\ 0.3 \end{bmatrix}$

Therefore, the force on each mass *F* is

$$F = k \times q = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{bmatrix} = \begin{bmatrix} 0.72 \\ 1 \\ 0.44 \\ -0.47 \\ 2.16 \end{bmatrix} \text{ (N)}$$

## Problem 2

(a) Answer:

The figure of the 5-page network is shown as below:

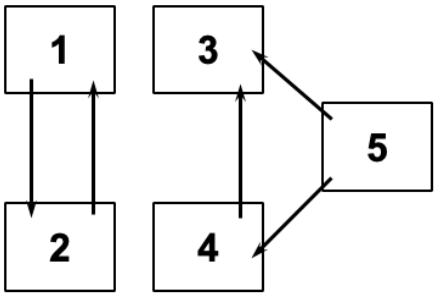


**Fig 1.** 5-page Network

According to the figure, Matrix *A* and Matrix *S* are:

$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $S = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix}$

When $m = 0.15$, then Matrix *M* is:

$M = \begin{bmatrix} 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.88 & 0.455 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.455 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix}$, and $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0.695 \\ 0.695 \\ 0.153 \\ 0.080 \\ 0.055 \end{bmatrix}$, after normalizing, $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.220 \\ 0.115 \\ 0.078 \end{bmatrix}$

When $m = 0.25$, then Matrix *M* is:

$M = \begin{bmatrix} 0.05 & 0.8 & 0.05 & 0.05 & 0.05 \\ 0.8 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.8 & 0.425 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.425 \\ 0.05 & 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$, and $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0.668 \\ 0.668 \\ 0.274 \\ 0.149 \\ 0.105 \end{bmatrix}$, after normalizing, $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.411 \\ 0.223 \\ 0.156 \end{bmatrix}$

When $m = 0.35$, then Matrix *M* is:

$$M = \begin{bmatrix} 0.07 & 0.72 & 0.07 & 0.07 & 0.07 \\ 0.88 & 0.07 & 0.07 & 0.07 & 0.07 \\ 0.07 & 0.07 & 0.07 & 0.72 & 0.395 \\ 0.07 & 0.07 & 0.07 & 0.07 & 0.395 \\ 0.07 & 0.07 & 0.07 & 0.07 & 0.07 \end{bmatrix}, \text{ and } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0.624 \\ 0.624 \\ 0.384 \\ 0.221 \\ 0.161 \end{bmatrix}, \text{ after normalizing, } x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0.616 \\ 0.354 \\ 0.258 \end{bmatrix}$$

(b) Answer:

There are two distinct scores, namely 0.235606 and 1.

(c) Answer:

The top 10 teams at the end of the season are shown in Table 1.

**Table 1.** Top 10 teams at the end of the season and their importance scores

| Rank | Team Name | Team Index | Importance Score |
|---|---|---|---|
| 1 | Texas | 202 | 1 |
| 2 | LSU | 98 | 0.834906 |
| 3 | Virginia Tech | 218 | 0.822686 |
| 4 | Michigan | 111 | 0.81459 |
| 5 | Florida State | 61 | 0.756901 |
| 6 | Georgia | 68 | 0.699709 |
| 7 | Penn State | 152 | 0.694528 |
| 8 | Miami (Florida) | 109 | 0.652556 |
| 9 | Georgia Tech | 67 | 0.544553 |
| 10 | Auburn | 12 | 0.5439 |

Texas is the at the top of ranking. The importance score of Texas College versus week of the season is shown in Fig 2.

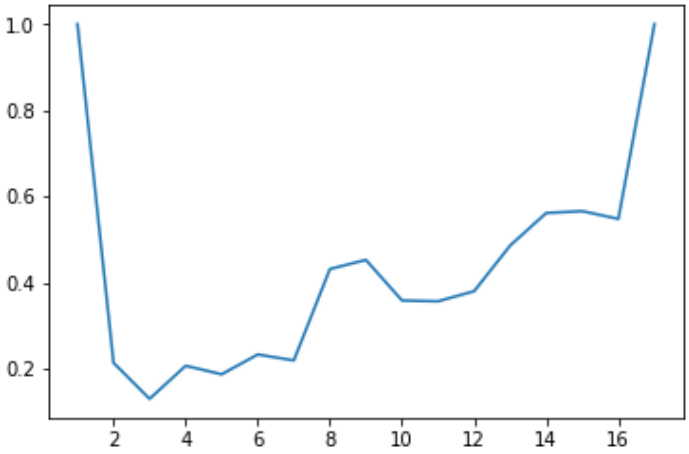


**Fig 2.** The importance score of Virginia College versus week of the season

The importance score of top ten teams versus week of the season is shown in Fig 3.

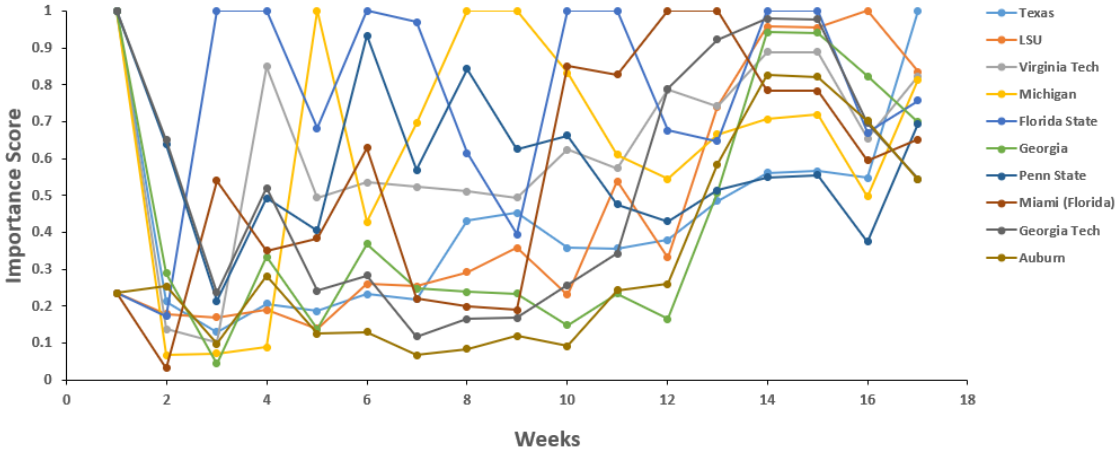


**Fig 3.** The importance score of top ten teams versus week of the season

## Problem 3

(a) Answer:

For **node 1,**

Vertically: $F_C * cos\gamma - F_A * cos\beta = 0$

Horizontally: $-F_{load} - F_C * sin\gamma - F_A * sin\beta = 0$

For **node 2,**

Horizontally: $H_2 + F_B + F_A * cos\beta = 0$

For **node 3,**

Vertically: $V_3 + F_c * sin\gamma = 0$

(b) Answer:

Combined with the two given equations, $V_2 + F_A * sin\beta = 0$ and $-F_B - F_C * cos\gamma = 0$, this problem can be converted to a linear system ($Ax = b$), where:

$$A = \begin{bmatrix} -cos\beta & 0 & cos\gamma & 0 & 0 & 0 \\ -sin\beta & 0 & -sin\gamma & 0 & 0 & 0 \\ cos\beta & 1 & 0 & 1 & 0 & 0 \\ sin\beta & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -cos\gamma & 0 & 0 & 0 \\ 0 & 0 & sin\gamma & 0 & 0 & 1 \end{bmatrix}, \quad x = \begin{bmatrix} F_A \\ F_B \\ F_C \\ H_2 \\ V_2 \\ V_3 \end{bmatrix} \text{ and } b = \begin{bmatrix} 0 \\ F_{load} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(c) Answer:

By given $F_{load} = 1000N, \alpha = 90°, \beta = 30° \text{ and } \gamma = 60°$, the matrix *A* and vector *b* becomes:

$$A = \begin{bmatrix} -0.866 & 0 & 0.5 & 0 & 0 & 0 \\ -0.5 & 0 & -0.866 & 0 & 0 & 0 \\ 0.866 & 1 & 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.866 & 0 & 0 & 1 \end{bmatrix} \text{ and } b = \begin{bmatrix} 0 \\ 1000 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

By solving the linear system $Ax = b$ numerically, we can get:

$$x = \begin{bmatrix} F_A \\ F_B \\ F_C \\ H_2 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -500 \\ 433.01 \\ -866.02 \\ 0 \\ 250 \\ 750 \end{bmatrix} \text{(N)}$$

(d) Answer:

The relation between $\alpha$, $\beta$, and $\gamma$ can be calculated based on the geometry of the truss. The truss is shown in Fig. 4.

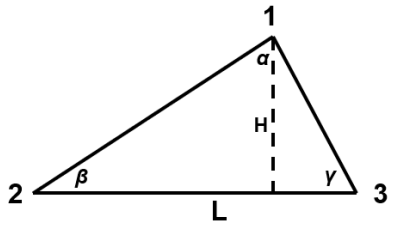


**Fig 4.** The geometric figure of the truss.

Let ***H*** be the height of Node 1, ***L*** be the distance between Node 2 and Node 3. Then we can get:

$$L = \frac{H}{tan\beta} + \frac{H}{tan\gamma}$$

Since in (c) $\beta = 30° \text{ and } \gamma = 60°$, then $H = \frac{\sqrt{3}}{4}L$, then the equation above can be rewritten as:

$$\frac{4}{\sqrt{3}} = \frac{1}{tan\beta} + \frac{1}{tan\gamma} = cot\beta + cot\gamma$$

Combining these three equations:

$$\begin{cases} F_C * cos\gamma - F_A * cos\beta = 0 \\ -F_{load} - F_C * sin\gamma - F_A * sin\beta = 0 \\ V_3 + F_c * sin\gamma = 0 \end{cases}$$

We can get this equation: $(1 + \frac{tan\beta}{tan\gamma})V_3 = F_{load}$ , since $F_{load} = 1000$, the equation can be rewritten as:

$$\left(1 + \frac{cot\gamma}{cot\beta}\right) V_3 = F_{load}$$

Combining the equation above we have:

$$\left(1 + \frac{cot\gamma}{(\frac{4}{\sqrt{3}} - cot\gamma)}\right) V_3 = 1000$$

Define a function in python taking in $\gamma$ as input parameter and return ($V_{3max} - V_3$), please find the code in Appendix 3(d) part.

Solving this non-linear function by ***scripy.optimize.newton***, the root is 1.1381068493529425 (rad), which is nearly 32.6°.

# Appendix

## 1. Code of Problem 1

### (1) code for part a

```
#the mass vector
mass = np.array([[8.,2.,3.,5.,9.]]).T
#the matrix K
K = np.array([[8.,3.,7.,1.,0.],
              [3.,7.,5.,0.,1.],
              [7.,5.,9.,3.,1.],
              [1.,0.,3.,5.,3.],
              [0.,1.,1.,3.,10.]])

#divide each row with the corresponding row of vector mass to get a new matrix of K
for i in range(5):
    K[i,:] = K[i,:]/mass[i,0]
#solve for eigen values and vectors
C = np.linalg.eig(K)
```

### (2) code for part b

```
#the given vector of displacement
q = np.array([[0.1,0.05,0.01,-0.3,0.3]]).T
#get norm 2
norm=np.linalg.norm(q, ord=2)
#normalize
q = q/norm
#multiply
Q = np.matmul(C[1],q)
```

### (3) code for part c

```
#the matrix K
K = np.array([[8.,3.,7.,1.,0.],
              [3.,7.,5.,0.,1.],
              [7.,5.,9.,3.,1.],
              [1.,0.,3.,5.,3.],
              [0.,1.,1.,3.,10.]])
#the given vector of displacement
q = np.array([[0.1,0.05,0.01,-0.3,0.3]]).T
#calculate the force of each mass
F = np.matmul(K,q)
```

## 2. Code of Problem 2

### (1) code for part a

```
#the matrix of A
A = np.array([[0.,1.,0.,0.,0.],
              [1.,0.,0.,0.,0.],
              [0.,0.,0.,1.,0.5],
              [0.,0.,0.,0.,0.5],
              [0.,0.,0.,0.,0.]])
#get size of matrix A
```

```
n = A.shape[0]
#build matrix S
S = np.ones((n,n))/n
#set value of m
m = 0.15
#build matrix S
M = (1-m)*A + m*S
#solve for eigenvalues and eigen vectors
C = np.linalg.eig(M)
#get the result
V = C[1][:,0]
#normalize by norm 1
V = V/np.max(V)
```

## (2) code for part b and c

```
#include libraries
import numpy as np
import json
import matplotlib.pyplot as plt
#read in results
results = json.load(open('results.json'))
team_names = json.load(open('teams.json'))
'''
  * @name: getTopTenScores
  * @description: get the top 10 results after given weeks
  * @param results: results of all matches
  * @param team_names: names of the college teams
  * @param week: the given week
  * @param index: the index of the team
  * @return: info, data structure{"teamName", teamNumber,importance score}
'''
def getTopTenScores(results,team_names,week):
        #get datas for all teams after given weeks
        info = getScore(results,team_names,week)
        #sort the result by importance score
        info = np.sort(info, order='score')
        info = np.flip(info)
        #get top ten results
        result = info[0:10]

        return result
'''
  * @name: getScoreByIndex
  * @description: get the scores of a certain team after given weeks
  * @param results: results of all matches
  * @param team_names: names of the college teams
  * @param week: the given week
  * @param index: the index of the team
```

```
 * @return: info, data structure{"teamName", teamNumber,importance score}
'''
def getScoreByIndex(results,team_names,week,index):
    #get datas for all teams after given weeks
    info = getScore(results,team_names,week)

    return info[index]
'''
 * @name: displayScore
 * @description: get all the scores of a certain team and display the result
 * @param results: results of all matches
 * @param team_names: names of the college teams
 * @param index: the index of the team
 * @return: array of scores of the given team
'''
def displayScore(results,team_names,index):
    #set initial values of scores and weeks
    scores = np.zeros((17,1))
    weeks = np.zeros((17,1))
    #loop each week to get importance score for the given team
    for i in range(17):
        temp= getScoreBy I n d e x(results,team_names,i,index)
        weeks[i,0] = i+1
        scores[i,0] = temp['score']
    #plot the importance vesus week
    ax = plt.gca()
    plt.plot(weeks,scores)

    return scores
'''
 * @name: getScore
 * @description: get the scores of all teams after given weeks
 * @param results: results of all matches
 * @param team_names: names of the college teams
 * @param week: the given week
 * @return: info, data structure{"teamName", teamNumber,importance score}
'''
def getScore(results,team_names,week):
    #get all results from week 1 to given week
    weekresults = [result for result in results if result['week'] <= week]

    #get total number of teams
    teamNum = len(team_names)

    #set initial values of A and S
    A = np.zeros((teamNum,teamNum))
    S = np.ones((teamNum,teamNum))/teamNum
```

```
    #set the values of A according to match result

    #loop all matches
    for result in weekresults:
        #get home team and away team
        homeNo = result['home_team']
        awayNo = result['away_team']
        #if away wins
        if result['home_score']<result['away_score']:
            A[awayNo,homeNo] = 1
        else:
            A[homeNo,awayNo] = 1
    #make each column sum equals to 1
    for col in range(teamNum):
        temp = np.sum(A[:,col])
        if temp != 0:
            A[:,col] = A[:,col]/temp
    #set the value of m
    m = 0.15
    #get matrix M
    M = (1-m)*A + m*S

    #set the initial guess of x
    x = np.ones((teamNum,1))

    #power method to calculate x
    for i in range(50):
        x = np.matmul(M,x)
        #normalize x
        x = x/np.max(x)

    #declare a new data structure to save info
    dtype = [('teamName', 'S20'), ('teamNo', float), ('score', float)]
    info = np.empty(teamNum,dtype)

    #set the values of the information, including team name, team no, importance score
    for i in range(teamNum):
        info[i] = (team_names[i],i,x[i,0])

    return info
```

## 3. Code of Problem 3

### (1) code for part c

```
#include libraries
import numpy as np
import matplotlib.pyplot as plt
import math
#set initial values
beta = math.pi/6
```

```
gamma = math.pi/3
Fload = 1000
#get vector b
b = np.zeros((6,1))
b[1,0] = Fload
#initial matrix of A
A = np.zeros((6,6))
#1sr equation
A[0,0] = -np.cos(beta)
A[0,2] = np.cos(gamma)
#2nd equation
A[1,0] = -np.sin(beta)
A[1,2] = -np.sin(gamma)
#3rd equation
A[2,0] = np.cos(beta)
A[2,1] = 1
A[2,3] = 1
#4th equation
A[3,0] = np.sin(beta)
A[3,4] = 1
#5th equation
A[4,1] = -1
A[4,2] = -np.cos(gamma)
#6th equation
A[5,2] = np.sin(gamma)
A[5,5] = 1
#reverse matrix A
invA = np.linalg.inv(A)
#solve the linear system
F = np.matmul(invA,b)
```

## (2) code for part d

```
#include libraries
import math
from scipy import optimize
'''
  * @name: getV3
  * @description: the function to calculate V3 based on given gamma
  * @param gamma: input gamma
  * @return: value of V3max – V3
'''
def getV3(gamma):
      #get cot of gamma
      cot = 1/np.tan(gamma)
      k = 1+cot/(4/math.sqrt(3)-cot)

      return 800-1000/k
root = optimize.newton(getV3, 1)
```