**ENGN2020 – Midterm #1**

### Problem 1

#### (a) Answer:

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

Where, is the second derivative of displacement against time.

According to the question, , , and the equation above can be rewritten as :

This is a typical eigenvalue problem: **,** where , and are the natural frequencies of this system.

In this problem:

Therefore,

Solving this eigenvalue problem:

Eigenvalues are ﻿ , and

The corresponding eigenvalues are:

(m), , (m), (m), and (m)

**(b) Answer:**

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

The given displacement (m), after normalizing, .

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

**(c) Answer:**

, and

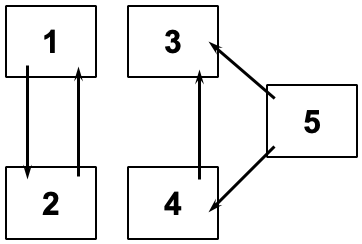
Therefore, the force on each mass *F* is

(N)

### Problem 2

#### 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:

,

When , then Matrix *M* is:

, and , after normalizing,

When , then Matrix *M* is:

, and , after normalizing,

When , then Matrix *M* is:

, and , after normalizing,

#### Answer:

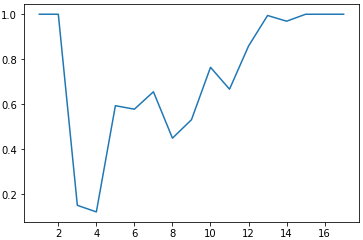
There are two distinct scores, namely 0.235606 and 1.

#### Answer:

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

|  |  |  |
| --- | --- | --- |
| Rank | Team Name | Importance Score |
| 1 | Virginia Tech | 1.0 |
| 2 | Texas | 0.97034525 |
| 3 | Miami (Florida) | 0.85699824 |
| 4 | Florida State | 0.78943401 |
| 5 | LSU | 0.74674676 |
| 6 | Penn State | 0.73363148 |
| 7 | Georgia Tech | 0.70105379 |
| 8 | Boston College | 0.69115393 |
| 9 | Clemson | 0.66096419 |
| 10 | Georgia | 0.64518823 |

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



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

### Problem 3

#### Answer:

For **node 1,**

Vertically:

Horizontally:

For **node 2,**

Horizontally:

For **node 3,**

Vertically:

#### Answer:

Combined with the two given equations, and , this problem can be converted to a linear system (), where:

， and

#### Answer:

By given , the matrix *A* and vector *b* becomes:

and

By solving the linear system numerically, we can get:

=

#### Answer:

Combining these three equations:

We can get this equation: , since , the equation can be rewritten as:

### Appendix

#### 1. Code of Problem 1

(1) 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) part b

(3) 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) 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) 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= getScoreBｙＩｎｄｅｘ(results,team\_names,i,index)

weeks[i,0] = i+1

scores[i,0] = temp['score']

#plot the importance vesus week

ax = plt.gca()

plt.xlim((-1, 18))

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

#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

#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)