Experiment 1

Recollection of MATLAB Basics

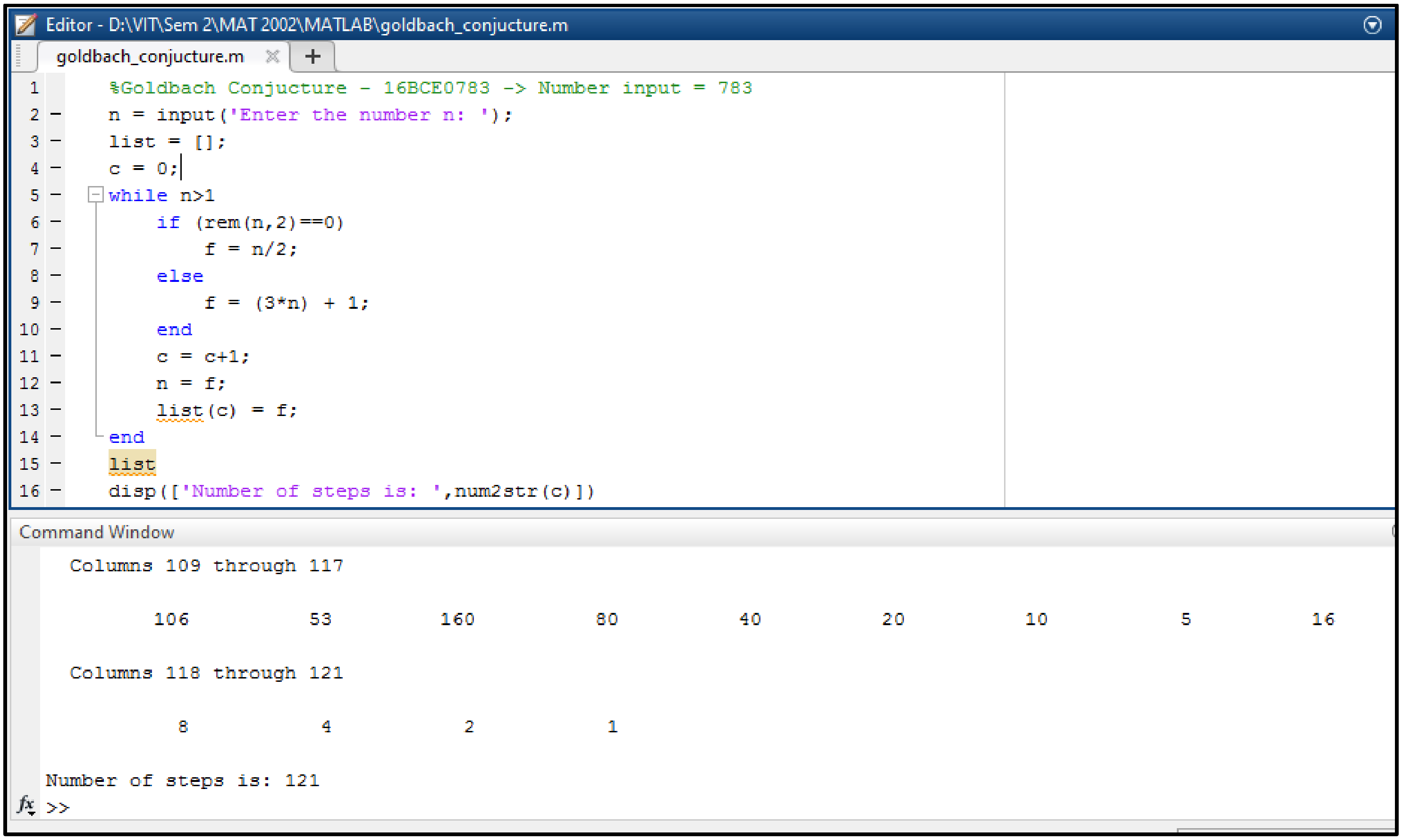
Recollected the basics on Indexing/Slicing/Concatenating of Matrices and Vectors. Also different functions of/on Matrices. Recollected the plotting commands such as plot/plot3/ezplot/surf etc.

Question – 1 Goldbach Conjecture

Code:

```
%Goldbach Conjucture - 16BCE0783 -> Number input = 783
n = input('Enter the number n: ');
list = [];
c = 0;
while n>1
    if (rem(n,2)==0)
        f = n/2;
    else
        f = (3*n) + 1;
    end
    c = c+1;
    n = f;
    list(c) = f;
end
list
disp(['Number of steps is: ',num2str(c)])
```

Output: (Entered n = 783)



Question 2: AB = C

Code:

```
%Question 1 - 16BCE0783
l = [];
for i=1:8;l(i) = 0.25*(2^(i-1));end
A = cat(1,1:2:15,2.5:-0.5:-1,l)
C = A(:,[1,4,7])
B = A\C
```

Output:

```
D: ▸ VIT ▸ Sem 2 ▸ MAT 2002 ▸ MATLAB ▸ FFCS
Command Window
>> exp1q1

A =

    1.0000    3.0000    5.0000    7.0000    9.0000   11.0000   13.0000   15.0000
    2.5000    2.0000    1.5000    1.0000    0.5000         0   -0.5000   -1.0000
    0.2500    0.5000    1.0000    2.0000    4.0000    8.0000   16.0000   32.0000


C =

    1.0000    7.0000   13.0000
    2.5000    1.0000   -0.5000
    0.2500    2.0000   16.0000


B =

    1.0000    0.3445   -0.0766
         0         0         0
         0         0         0
         0         0         0
         0         0         0
    0.0000    0.7943    0.7679
         0         0         0
   -0.0000   -0.1388    0.3086

fx >>
```

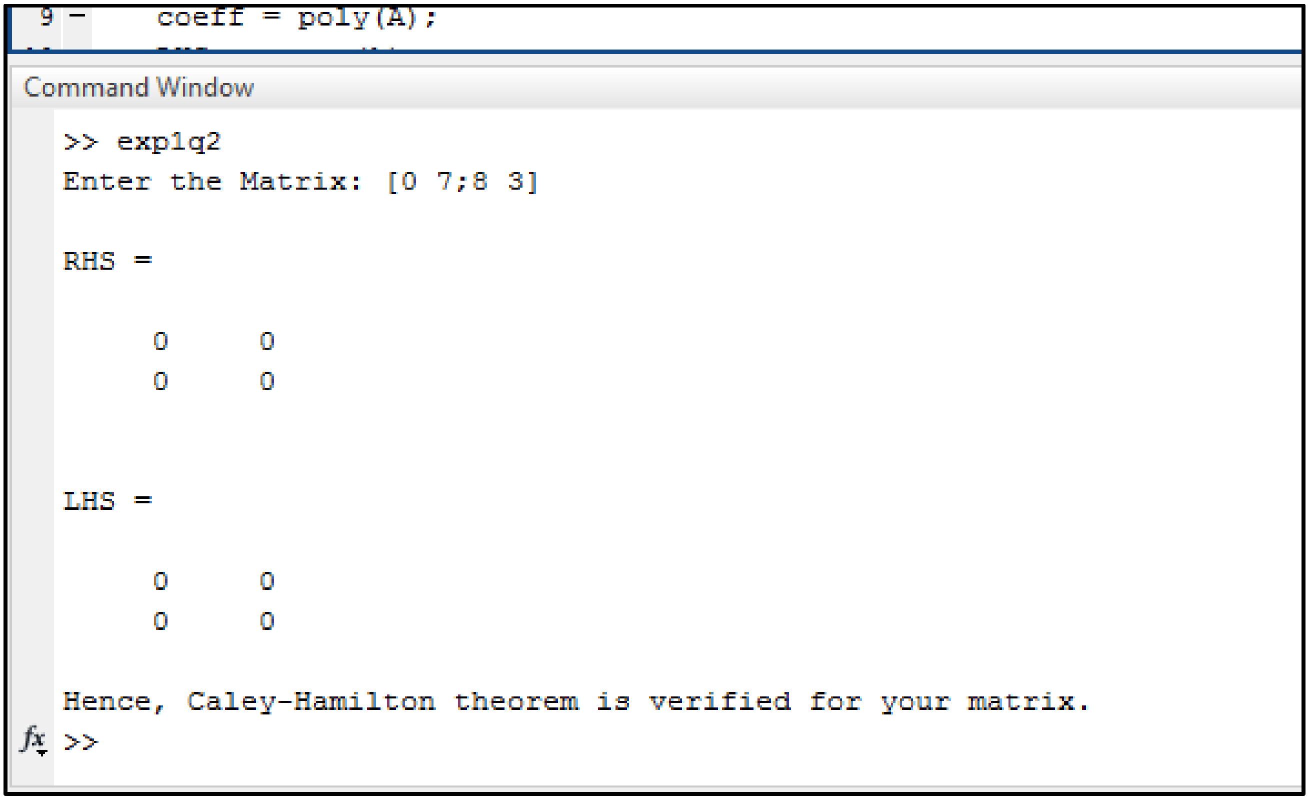
Question 3: Verifying Calyley Hamilton Theorem

Code:

```
%Cayley Hamilton Theorem - 16BCE0783
A = input('Enter the Matrix: ');
k = size(A);
RHS = zeros(k)
if k(1) ~= k(2)
    disp('Not a square matrix!!')
    return;
end
coeff = poly(A);
LHS = zeros(k);
for i=1:numel(coeff)
    LHS = LHS + round(coeff(i))*A^(k(1)+1-i);
end
LHS
if (LHS == RHS)
    disp('Hence, Caley-Hamilton theorem is verified for your matrix.')
else
```

```
    disp('Caley-Hamilton theorem is not verified for your matrix.')
end
```

Input & Output:



```
9 -     coeff = poly(A);
Command Window
>> exp1q2
Enter the Matrix: [0 7;8 3]

RHS =

     0     0
     0     0


LHS =

     0     0
     0     0

Hence, Caley-Hamilton theorem is verified for your matrix.
>>
```

Question 4: Newton-Raphson Approximation Method

Code:

```
%Newton - Raphson Approximation - 16BCE0783
syms x
f = input('Input an algebraic or trancedental function of x: ');
a0 = input('Input the value of initial Approximation for your function: ');
l = [a0];
df = diff(f,x);
for i=2:11
    l(i) = l(i-1) - (subs(f,x,l(i-1))/subs(df,x,l(i-1)));
end
disp('The list of roots from x1 to x10 are:')
disp(l(2:11))
fprintf('The fifth root of the given number is %f\n',l(11))
```

Input & Output:

```
12 -    fprintf('The fifth root of the given number is %f\n',l(11))
```

Command Window

```
>> exp1q3
Input an algebraic or trancedental function of x: x^5 - 83
Input the value of initial Approximation for your function: 3
The list of roots from x1 to x10 are:
    2.6049    2.4445    2.4205    2.4200    2.4200    2.4200    2.4200    2.4200    2.4200    2.4200

The fifth root of the given number is 2.420001
>>
```

Experiment 2

Google's Mechanism for ranking WebPages (Random Surfer)

Aim:

To Understand the Random Surfer Algorithm which was used initially in Google's Search Engine and was developed by Lawrence Page (and Sergey Brin) using MATLAB.

Mathematical Background:

According to Google:

"PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites." (Term is Citation)

For a web of pages A, B, C, D,… the PageRank of A is given by:

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)} + \cdots\right).$$

PR(A) denotes PageRank of A and L() denotes the total number of outgoing links. In our experiment d = 1. (Generally dampning factor, d = 0.85). Links from a page to itself (known as self-citation) are ignored and multiple outgoing links to a single page are counted as 1 only.

Solving for the PR() equations(linear system) involves the use of Matrices and further the use of Eigen Values and Eigen Vectors. The system of these linear equations conclude to AX = X. Hence, we can say that X is a eigen vector corresponding to eigen value 1. This X eigen vector will give us a non-trivial Solutions.

Code:

```
%Pagerank Algorithm - 16BCE0783
A = input('Input the (Transition)matrix: ');
k = size(A);
if (k(1)~=k(2))
    fprintf('\nNot a square matrix\n\n')
    return;
end
[X,Y] = eig(A);
cor_vec = 0;
for i=1:k
    if round(Y(i,i),10) == 1
        cor_vec = i;   %cor_vec is diagonal number of eigen vector corresponding to eigen 
value 1
```

```
        break;
    end
end
if cor_vec==0
    fprintf('\nNone of the eigenvalues is 1\n\n')
    return;
end
rank = X(:,cor_vec)/sum(X(:,cor_vec));  %normalization - Dividing by sum of the elements
of eigen vector corresponding to eigen value 1s
for i=1:k
    [val,pos] = max(rank);
    fprintf('\nRank %d is page %c with a Probability of %s\n',i,64+pos,num2str(val));
    rank(pos) = [-1];
end
%can use sort also [A,B] = sort()
```

Question 1: Checking for a stochastic Matrix

Code:

```
%Checking Stochastic Matrix - 16BCE0783
A = input('Enter the Matrix: ');
k = size(A);
if k(1) ~= k(2)
    disp('Not a Square Matrix!')
    return;
end
for j = 1:k(1)^2
    if A(j)<0
        disp('Given Matrix is not a stochastic Matrix since it contains a non-positive
number')
        return;
    end
end
option = input('Enter 1 to check for column stochastic or 2 for row stochastic: ');
flag = 1;
if option == 1
    for i=1:k(1)
        if sum(A(:,i))~=1
            flag = 0;
            break;
        end
    end
else
    for i=1:k(1)
        if sum(A(i,:))~=1
            flag = 0;
            break;
        end
    end
end
if flag == 1
    disp('Given Matrix is a stochastic Matrix')
else
    disp('Given Matrix is not a stochastic Matrix')
end
```

Input and Output:

```
12 -        end
Command Window
>> exp2q1
Enter the Matrix: [1/3 1/2 0;1/3 0 1;1/3 1/2 0]
Enter 1 to check for column stochastic or 2 for row stochastic: 1
Given Matrix is a stochastic Matrix
>> exp2q1
Enter the Matrix: [1/3 1/2 -1;1/3 0 1;1/3 1/2 1]
Given Matrix is not a stochastic Matrix since it contains a non-positive number
fx >>
```

Question 2: Getting Pageranks for the given web of pages

Code: As written in the main Code Section above

Input and Outputs:

```
Command Window
>> pagerank_algo
Input the (Transition)matrix: [0 1/3 1/3 0 1/3 0;0 0 0 1/3 1/3 1;1/3 0 0 1/3 1/3 0;1/3 1/3 1/3 0 0 0;1/3 1/3 1/3 0 0 0;0 0 0 1/3 0 0]

Rank 1 is page D with a Probability of 0.1875

Rank 2 is page E with a Probability of 0.1875          -----> For Qu. 2 (b)

Rank 3 is page C with a Probability of 0.1875

Rank 4 is page B with a Probability of 0.1875

Rank 5 is page A with a Probability of 0.1875

Rank 6 is page F with a Probability of 0.0625
>> pagerank_algo
Input the (Transition)matrix: [0 1/3 1 1/3 0;1/2 0 0 0 0;0 1/3 0 1/3 0;1/2 0 0 0 0;0 1/3 0 1/3 0]
                                                                Error
None of the eigenvalues is 1

>> pagerank_algo
Input the (Transition)matrix: [0 1/3 1 1/3 0;1/2 0 0 0 0;0 1/3 0 1/3 1;1/2 0 0 0 0;0 1/3 0 1/3 0]

Rank 1 is page A with a Probability of 0.33333

Rank 2 is page C with a Probability of 0.22222

Rank 3 is page D with a Probability of 0.16667          -----> For Qu. 2 (a)

Rank 4 is page B with a Probability of 0.16667

Rank 5 is page E with a Probability of 0.11111
fx >>
```

2 (a)

$$x_1 = 0x_1 + \tfrac{1}{3}x_2 + 1x_3 + \tfrac{1}{3}x_4 + 0x_5$$
$$x_2 = \tfrac{1}{2}x_1 + 0x_2 + 0x_3 + 0x_4 + 0x_5$$
$$x_3 = 0x_1 + \tfrac{1}{3}x_2 + 0x_3 + \tfrac{1}{3}x_4 + 1x_5.$$
$$x_4 = \tfrac{1}{2}x_1 + 0x_2 + 0x_3 + 0x_4 + 0x_5.$$
$$x_5 = 0x_1 + \tfrac{1}{3}x_2 + 0x_3 + \tfrac{1}{3}x_4 + 0x_5.$$

$$\begin{bmatrix} 0 & 1 & \tfrac{1}{3} & 1 & 0 \\ \tfrac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \tfrac{1}{3} & 0 & \tfrac{1}{3} & 1 \\ \tfrac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \tfrac{1}{3} & 0 & \tfrac{1}{3} & 0 \end{bmatrix}$$

2 (b)

$$x_1 = 0x_1 + \tfrac{1}{3}x_2 + \tfrac{1}{3}x_3 + 0x_4 + \tfrac{1}{3}x_5 + 0x_6.$$
$$x_2 = 0x_1 + 0x_2 + 0x_3 + \tfrac{1}{3}x_4 + \tfrac{1}{3}x_5 + 1x_6.$$
$$x_3 = \tfrac{1}{3}x_1 + 0x_2 + 0x_3 + \tfrac{1}{3}x_4 + \tfrac{1}{3}x_5 + 0x_6$$
$$x_4 = \tfrac{1}{3}x_1 + \tfrac{1}{3}x_2 + \tfrac{1}{3}x_3 + 0x_4 + 0x_5 + 0x_6.$$
$$x_5 = \tfrac{1}{3}x_1 + \tfrac{1}{3}x_2 + \tfrac{1}{3}x_3 + 0x_4 + 0x_5 + 0x_6.$$
$$x_6 = 0x_1 + 0x_2 + 0x_3 + \tfrac{1}{3}x_4 + 0x_5 + 0x_6.$$

$$\begin{bmatrix} 0 & \tfrac{1}{3} & \tfrac{1}{3} & 0 & \tfrac{1}{3} & 0 \\ 0 & 0 & 0 & \tfrac{1}{3} & \tfrac{1}{3} & 1 \\ \tfrac{1}{3} & 0 & 0 & \tfrac{1}{3} & \tfrac{1}{3} & 0 \\ \tfrac{1}{3} & \tfrac{1}{3} & \tfrac{1}{3} & 0 & 0 & 0 \\ \tfrac{1}{3} & \tfrac{1}{3} & \tfrac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \tfrac{1}{3} & 0 & 0 \end{bmatrix}$$

Experiment 3

Stress distribution in a Tower bridge

Aim:

Finding principal stresses for a two-dimensional simply supported beam by finding the eigenvalues of the stress matrix. Also, visualizing the Eigen values of stress matrix for a simply supported beam.

Mathematical Background:

The **principal stresses** for a two dimensional simply supported beam **are the eigenvalues** of the stress matrix (say S). The stress matrices (2-D and 3-D) are given by:

$$\begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} \quad \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix}$$

The sigma components are normal stresses and tau components are Shear Stresses. If we change the orientation of the plane, the normal stress component will vary. There exists a **special orientation** where the **normal stresses are maximum**, and these planes are called principal planes and the normal stresses acting on them are called the **principal stresses**. Principal Stress is nothing but maximum normal stress which will happen when Shear Stress are zero, i.e., **the elements other than the diagonal elements in the stress matrix are zero.** Here comes the use of **Diagonalization**. After Diagonalization, the diagonalized matrix will look like:

$$\begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

Here, sigma(1, 2 and 3) are the eigen values of the stress matrix which are actually values of principal stresses.

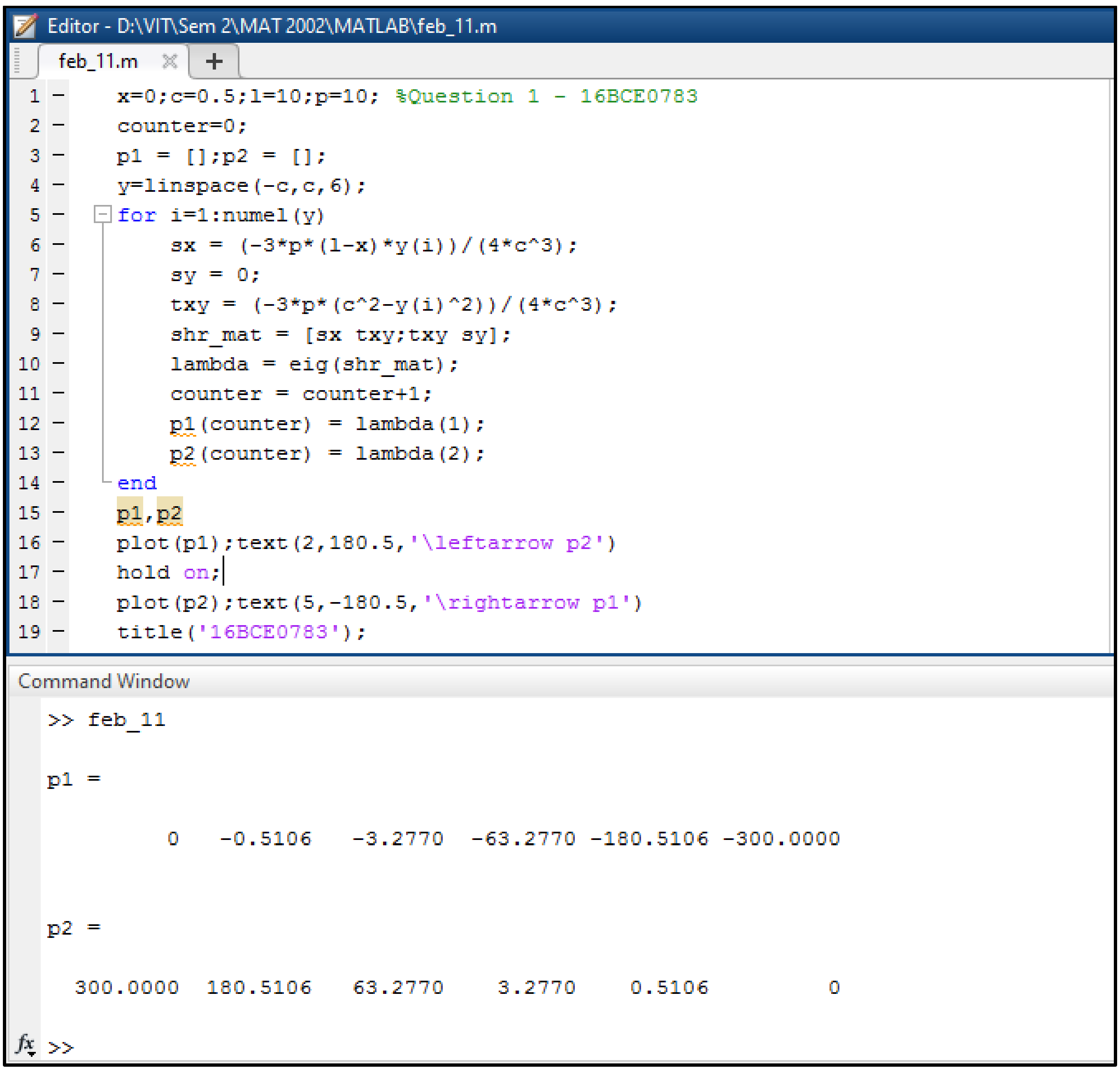
Question 1: Stress distribution in a simply supported beam (normal 2D Plot) (only y varies)

Code:

```
x=0;c=0.5;l=10;p=10; %Question 1 - 16BCE0783
counter=0;
p1 = [];p2 = [];
```

```
y=linspace(-c,c,6);
for i=1:numel(y)
    sx = (-3*p*(l-x)*y(i))/(4*c^3);
    sy = 0;
    txy = (-3*p*(c^2-y(i)^2))/(4*c^3);
    shr_mat = [sx txy;txy sy];
    lambda = eig(shr_mat);
    counter = counter+1;
    p1(counter) = lambda(1);
    p2(counter) = lambda(2);
end
p1,p2
plot(p1);text(2,180.5,'\leftarrow p2')
hold on;
plot(p2);text(5,-180.5,'\rightarrow p1')
title('16BCE0783');
```

Output:



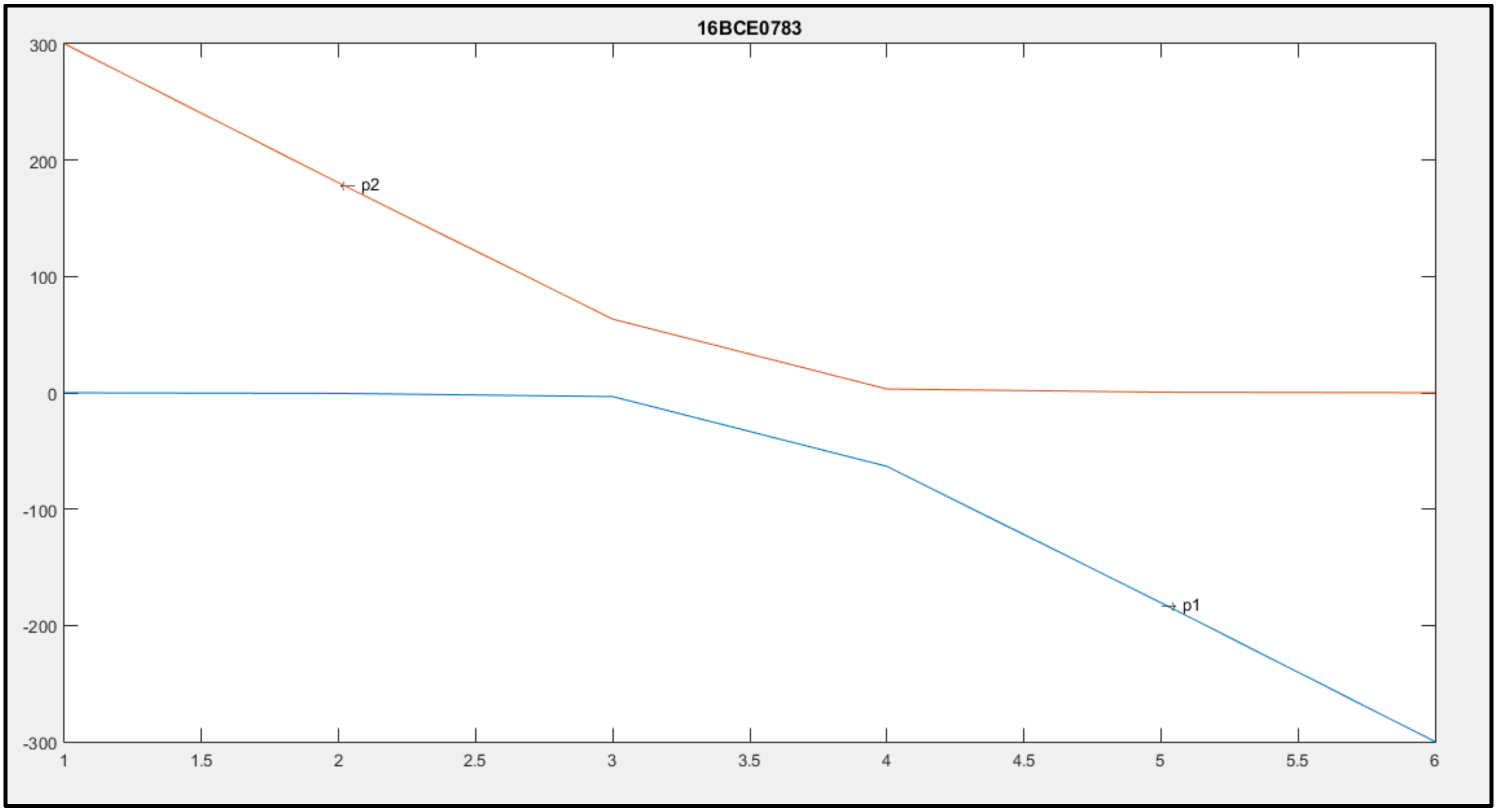
Editor - D:\VIT\Sem 2\MAT 2002\MATLAB\feb_11.m

feb_11.m

```
x=0;c=0.5;l=10;p=10; %Question 1 - 16BCE0783
counter=0;
p1 = [];p2 = [];
y=linspace(-c,c,6);
for i=1:numel(y)
    sx = (-3*p*(l-x)*y(i))/(4*c^3);
    sy = 0;
    txy = (-3*p*(c^2-y(i)^2))/(4*c^3);
    shr_mat = [sx txy;txy sy];
    lambda = eig(shr_mat);
    counter = counter+1;
    p1(counter) = lambda(1);
    p2(counter) = lambda(2);
end
p1,p2
plot(p1);text(2,180.5,'\leftarrow p2')
hold on;
plot(p2);text(5,-180.5,'\rightarrow p1')
title('16BCE0783');
```

Command Window

```
>> feb_11

p1 =

         0   -0.5106   -3.2770  -63.2770 -180.5106 -300.0000


p2 =

  300.0000  180.5106   63.2770    3.2770    0.5106         0

>>
```

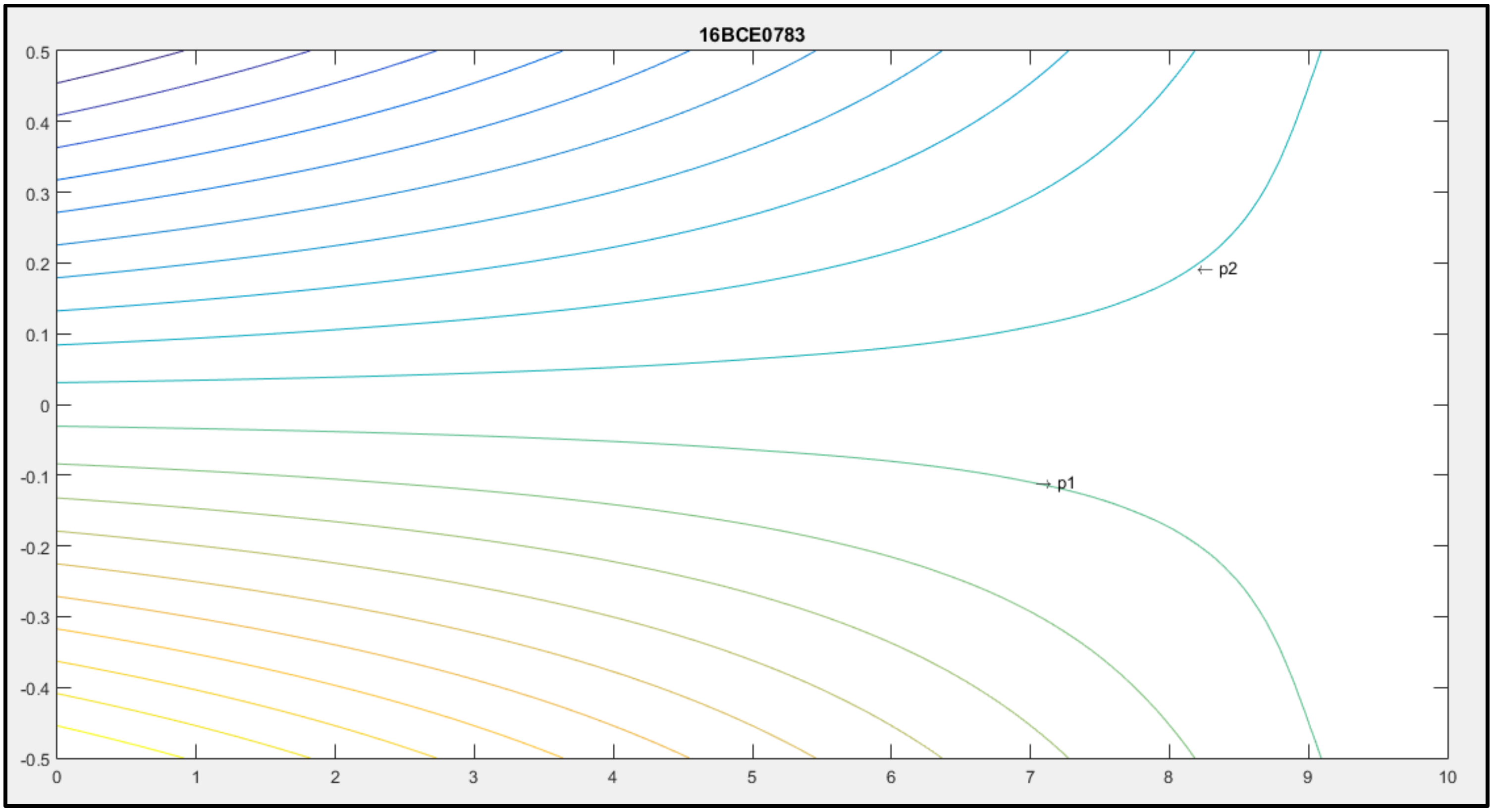


Question 2: Stress distribution in a simply supported beam (contour Plot) (both x and y vary)

Code:

```
c=0.5;l=10;p=10;
p1 = [];p2 = [];
x=linspace(0,l,200);
y=linspace(-c,c,24);
[X,Y] = meshgrid(x,y);
for i=1:numel(x)
    for j=1:numel(y)
        sx = (-3*p*(l-X(j,i)).*Y(j,i))/(4*c^3);
        sy = 0;
        txy = (-3*p*(c^2-Y(j,i).^2))/(4*c^3);
        shr_mat = [sx txy;txy sy];
        lambda = eig(shr_mat);
        p1(j,i) = lambda(1);
        p2(j,i) = lambda(2);
    end
end
contour(X,Y,p2,10);text(8.191,0.1957,'\leftarrow p2')
hold on;
contour(X,Y,p1,10);text(7.035,-0.1087,'\rightarrow p1')
title('16BCE0783')
```

Output:


16BCE0783
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
1
2
3
4
5
6
7
8
9
10
p2
p1