MATLAB PROJECT 1

Please include this page in your Group file, as a front page. Type in the group number and the names of all members WHO PARTICIPATED in this project.

GROUP # \_\_\_\_\_4\_\_\_\_\_\_

FIRST & LAST NAMES (UFID numbers are NOT required):

\_1. Nathaniel Barragan\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_2. Gannon Woods\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

­­­

\_3. Salem Bahadi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_4. Pedro Munoz\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_5. Michael Arnold\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_6.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**By signing your names above, each of you had confirmed that you did the work and agree with the work submitted**.

**% Exercise 1**

randi([0 1],5,4)

ans =

1 0 1 0

0 1 0 1

1 1 0 0

0 1 0 1

1 1 1 0

randi([0 1],5,4)

ans =

0 1 1 0

0 1 1 1

1 1 0 1

0 1 1 1

0 0 0 0

randi([0 1],5,4)

ans =

1 1 0 0

0 0 1 0

0 1 1 1

0 0 1 0

0 1 0 1

A=randi(9,4,1)

A =

9

9

2

9

B = [A, A.^0, A.^1, A.^2, A.^3, A.^4, A.^5]

B =

Columns 1 through 5

9 1 9 81 729

9 1 9 81 729

2 1 2 4 8

9 1 9 81 729

Columns 6 through 7

6561 59049

6561 59049

16 32

6561 59049

C=flipud(eye(5).\*randi(9,5,5))

C =

0 0 0 0 7

0 0 0 5 0

0 0 9 0 0

0 4 0 0 0

6 0 0 0 0

G=[1 0 0 0 0 1;0 0 0 0 1 0;0 0 0 1 0 0;0 0 1 0 0 0;0 1 0 0 0 0;1 0 0 0 0 0].\*randi([10,100],6,6)

G =

66 0 0 0 0 65

0 0 0 0 48 0

0 0 0 97 0 0

0 0 23 0 0 0

0 69 0 0 0 0

64 0 0 0 0 0

**% Exercise 2**

type multi

function [C,CRows,CColumns] = multi(A,B)

% This function multiplies two matrices, provided their

% dimensions meet the conditions

[ARows,AColumns]=size(A);

[BRows,BColumns]=size(B);

if AColumns==BRows

C=A\*B

else

disp('The dimensions of A and B disagree')

C=[]

end

if AColumns==BRows

R1=[];

for i=1:ARows

R1=[R1;A(i,:)\*B];

end

CRows=R1

else

CRows=[]

end

if AColumns==BRows

R2=[];

for i=1:BColumns

R2=[R2 A\*B(:,i)];

end

CColumns=R2

else

CColumns=[]

end

A=randi(10,2,3), B=magic(2)

A =

3 10 2

6 10 10

B =

1 3

4 2

multi(A,B);

The dimensions of A and B disagree

C =

[]

CRows =

[]

CColumns =

[]

A=magic(5), B=ones(4,6)

A =

17 24 1 8 15

23 5 7 14 16

4 6 13 20 22

10 12 19 21 3

11 18 25 2 9

B =

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

multi(A,B);

The dimensions of A and B disagree

C =

[]

CRows =

[]

CColumns =

[]

A=magic(4), B=ones(4,3)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

B =

1 1 1

1 1 1

1 1 1

1 1 1

multi(A,B);

C =

34 34 34

34 34 34

34 34 34

34 34 34

CRows =

34 34 34

34 34 34

34 34 34

34 34 34

CColumns =

34 34 34

34 34 34

34 34 34

34 34 34

A=ones(4), B=diag([2,3,4,5])

A =

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

B =

2 0 0 0

0 3 0 0

0 0 4 0

0 0 0 5

multi(A,B)

C =

2 3 4 5

2 3 4 5

2 3 4 5

2 3 4 5

CRows =

2 3 4 5

2 3 4 5

2 3 4 5

2 3 4 5

CColumns =

2 3 4 5

2 3 4 5

2 3 4 5

2 3 4 5

% Outputs and expected errors match.

**% Exercise 3**

type givensrot

function G = givensrot(n,i,j,theta)

% This function create a rotation matrix.

if 1<=i && i<j && j<=n

G=eye(n);

G(i,i)=cos(theta);

G(i,j)=-sin(theta);

G(j,i)=sin(theta);

G(j,j)=cos(theta);

else

G=[]

end

end

G=givensrot(4,3,2,pi/2)

G =

[]

G=givensrot(5,2,4,pi/4)

G =

1.0000 0 0 0 0

0 0.7071 0 -0.7071 0

0 0 1.0000 0 0

0 0.7071 0 0.7071 0

0 0 0 0 1.0000

G=givensrot(3,1,2,pi)

G =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

G\*[1;0;0]

ans =

-1.0000

0.0000

0

G\*[0;1;0]

ans =

-0.0000

-1.0000

0

G\*[0;0;1]

ans =

0

0

1

G\*[1;1;1]

ans =

-1.0000

-1.0000

1.0000

% Yes, these outputs represent the type of rotation described.

**% Exercise 4**

type toeplitz

function A = toeplitz(m,n,a)

% This function creates a mxn matrix for which the

% each descending diagonal is constant.

for i=1:m

for j=1:n

A(i,j)=a(n+i-j);

end

end

end

a=transpose([1:6])

a =

1

2

3

4

5

6

A=toeplitz(4,3,a)

A =

3 2 1

4 3 2

5 4 3

6 5 4

a=randi(10,6,1)

a =

5

3

9

2

3

2

A=toeplitz(3,4,a)

A =

2 9 3 5

3 2 9 3

2 3 2 9

a=[zeros(3,1);[1:4]']

a =

0

0

0

1

2

3

4

A=toeplitz(4,4,a)

A =

1 0 0 0

2 1 0 0

3 2 1 0

4 3 2 1

b=randi([10,100],5,1);

a=[zeros(4,1);b]

a =

0

0

0

0

30

49

38

94

49

A=toeplitz(5,5,a)

A =

30 0 0 0 0

49 30 0 0 0

38 49 30 0 0

94 38 49 30 0

49 94 38 49 30

a=transpose([1:6]);

A=toeplitz(4,3,a)

A =

3 2 1

4 3 2

5 4 3

6 5 4

a=randi(10,6,1);

A=toeplitz(3,4,a)

A =

3 2 3 8

4 3 2 3

5 4 3 2

a=[zeros(3,1);[1:4]'];

A=toeplitz(4,4,a)

A =

1 0 0 0

2 1 0 0

3 2 1 0

4 3 2 1

b=randi([10 100],1,1);

a=[zeros(4,1);b;zeros(4,1)]

a =

0

0

0

0

17

0

0

0

0

A=toeplitz(4,4,a)

A =

17 0 0 0

0 17 0 0

0 0 17 0

0 0 0 17

**% Exercise 5**

function P=stochastic(A)

rowsize=sum(A,2);

colsize=sum(A,1);

flag=0;

for j=1:size(rowsize,2)

if(rowsize(j)==0)

for k=1:size(colsize,2)

if(colsize(k)==1)

fprintf('A is not stochastic and cannot be scaled to stochastic')

out=[];

flag=1;

end

end

end

end

if(flag==0)

rightside=1;

for i=1:size(rowsize,2)

if(rowsize(i)~=1)

rightside=0;

break;

end

end

leftside=1;

for i=1:size(colsize,2)

if(colsize(i)~=1)

leftside=0;

break;

end

end

if(rightside==1 && leftside==1)

fprintf('A is a doubly stochastic matrix \n')

P=A

elseif(rightside==1 && leftside==0)

fprintf('A is only a right stochastic matrix \n')

P=A

elseif(leftside==1 && rightside==0)

fprintf('A is only a left stochastic matrix')

P=A

else

righside=1;

for j=1:size(rowsize,2)

if (rowsize(j)==0)

rightside=0;

end

end

leftside=1;

for k=1:size(colsize,2)

if (colsize(k)==0)

leftside=0;

end

end

if (rightside==1)

fprintf('neither left nor right stochastic but can be scaled to

stochastic')

for i=1:size(rowsize,2)

A(i,:)=A(i,:)/rowsize(1,i);

end

P=A

else

fprintf('neither left nor right stochastic but can be scaled to

stochastic')

for i=1:size(colsize,2)

A(:,i)=A(:,i)/colsize(1,i);

end

P=A

end

end

end

end

A=[0.5, 0 , 0.5; 0, 0, 1; 0.5, 0, 0.5]

A =

0.5000 0 0.5000

0 0 1.0000

0.5000 0 0.5000

P=stochastic(A)

A is only a right stochastic matrix

P =

0.5000 0 0.5000

0 0 1.0000

0.5000 0 0.5000

P =

0.5000 0 0.5000

0 0 1.0000

0.5000 0 0.5000

A=A'

A =

0.5000 0 0.5000

0 0 0

0.5000 1.0000 0.5000

P=stochastic(A)

A is a doubly stochastic matrix

P =

0.5000 0 0.5000

0 0 0

0.5000 1.0000 0.5000

P =

0.5000 0 0.5000

0 0 0

0.5000 1.0000 0.5000

A=[0.5, 0, 0.5; 0, 0, 1; 0, 0, 0.5]

A =

0.5000 0 0.5000

0 0 1.0000

0 0 0.5000

P=stochastic(A)

A is only a right stochastic matrix

P =

0.5000 0 0.5000

0 0 1.0000

0 0 0.5000

P =

0.5000 0 0.5000

0 0 1.0000

0 0 0.5000

A=A'

A =

0.5000 0 0

0 0 0

0.5000 1.0000 0.5000

P=stochastic(A)

neither left nor right stochastic but can be scaled to stochastic

P =

0.5000 0 0

0 0 0

0.5000 1.0000 1.0000

P =

0.5000 0 0

0 0 0

0.5000 1.0000 1.0000

A=magic(3)

A =

8 1 6

3 5 7

4 9 2

P=stochastic(A)

neither left nor right stochastic but can be scaled to stochastic

P =

0.5333 0.0667 0.4000

0.2000 0.3333 0.4667

0.2667 0.6000 0.1333

P =

0.5333 0.0667 0.4000

0.2000 0.3333 0.4667

0.2667 0.6000 0.1333

A=diag([1,2,3])

A =

1 0 0

0 2 0

0 0 3

P=stochastic(A)

A is only a right stochastic matrix

P =

1 0 0

0 2 0

0 0 3

P =

1 0 0

0 2 0

0 0 3

A=[0, 0, 0;0, 0.5, 0.5; 0, 0.5, 0.5]

A =

0 0 0

0 0.5000 0.5000

0 0.5000 0.5000

P=stochastic(A)

A is not stochastic and cannot be scaled to stochastic

**% Exercise 6**

x=linspace(0,4,8);

y=atan(x)+x-1;

plot(x,y);

syms x

f=atan(x)+x-1

f =

x + atan(x) - 1

g=diff(f)

g =

1/(x^2 + 1) + 1

% newtons function:

function root=newtons(N,X)

format long

f=atan(X)+X-1;

g=diff(f);

for i=1:1:N

x1=X - (atan(X+X-1))/(1/(X^2+1)+1);

X=x1;

disp(X);

end

root=X;

fprintf('Approximate Root is %.8f\n',root);

end

root=newtons(5,0.525)

0.496995307424201

0.500330253875027

0.499963262082021

0.500004081457619

0.499999546498128

Approximate Root is 0.49999955

root =

0.499999546498128

% These values are all roughly around 0.5, which is the actual root syms x

f=x^3-x-1

f =

x^3 - x - 1

g=diff(f)

g =

3\*x^2 – 1

% newtons\_1 function

function root=newtons\_1(N,X)

format short

f=X^3-X-1;

g=diff(f);

for i=1:1:N

x1=X - (X^3-X-1)/(3\*X^2-1);

X=x1;

%disp(X);

end

root=X;

fprintf('Approximate Root is %.8f\n',root);

end

newtons\_1(5,1.5)

1.3478

1.3252

1.3247

1.3247

1.3247

Approximate Root is 1.32471796

ans =

1.3247

% We think that the 1.3247 value is closer to the actual because it appeared more times in the iterations.

newtons\_1(5,1)

1.5000

1.3478

1.3252

1.3247

1.3247

Approximate Root is 1.32471796

ans =

1.3247

% We think that the 1.3247 value is closer to the actual because it appeared more times in the iterations.

newtons\_1(5,0.6)

17.9000

11.9468

7.9855

5.3569

3.6250

Approximate Root is 3.62499603

ans =

3.6250

% We think that the root is closer to 3.6250 although there is more variance because the values in the iterations are further apart.

newtons\_1(5,0.57)

-54.1655

-36.1143

-24.0821

-16.0634

-10.7215

Approximate Root is -10.72148342

ans =

-10.7215

% We think that the -10.7215 value is closer to the actual because it appeared more times in the iterations.

newtons\_1(10,0.6)

17.9000

11.9468

7.9855

5.3569

3.6250

2.5056

1.8201

1.4610

1.3393

1.3249

Approximate Root is 1.32491287

ans =

1.3249

newtons\_1(10,0.57)

-54.1655

-36.1143

-24.0821

-16.0634

-10.7215

-7.1655

-4.8017

-3.2334

-2.1937

-1.4969

Approximate Root is -1.49686657

ans =

-1.4969

newtons\_1(100,0.6)

Approximate Root is 1.32471796

ans =

1.3247

newtons\_1(100,0.57)

Approximate Root is 1.32471796

ans =

1.3247

% Because more iterations were involved, both values eventually converged to

% 1.3247. Since it took 100 iterations to converge to that value, it

% converges slowly.