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 # 27

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

\_1.\_Shanil Patel\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_2.\_Sean Tromans\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

\_3. Harrison Paparatto\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_4.\_Adam Wood\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_5.\_Joseph Samman\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_6.\_Mason Taylor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

diary on

format compact

%Exercise1

randi([0 1],5,4)

ans =

     0     0     1     1

     1     0     1     0

     0     0     0     0

     0     0     0     1

     0     0     0     0

randi([0 1],5,4)

ans =

     0     1     1     1

     0     1     0     1

     0     0     0     0

     0     0     0     1

     1     0     1     0

randi([0 1],5,4)

ans =

     1     1     0     1

     0     1     0     1

     1     0     0     1

     0     1     0     1

     0     1     1     0

x = [5;3;8;4]

x =

     5

     3

     8

     4

[x.^0 x.^1 x.^2 x.^3 x.^4 x.^5]

ans =

           1           5          25         125         625        3125

           1           3           9          27          81         243

           1           8          64         512        4096       32768

           1           4          16          64         256        1024

D = diag([randi([0 9]) randi([0 9]) randi([0 9]) randi([0 9]) randi([0 9])])

D =

     9     0     0     0     0

     0     4     0     0     0

     0     0     4     0     0

     0     0     0     3     0

     0     0     0     0     9

flipud(D)

ans =

     0     0     0     0     9

     0     0     0     3     0

     0     0     4     0     0

     0     4     0     0     0

     9     0     0     0     0

y = diag([randi([10 100],6,1)])

y =

    50     0     0     0     0     0

     0    37     0     0     0     0

     0     0    56     0     0     0

     0     0     0    56     0     0

     0     0     0     0    84     0

     0     0     0     0     0    82

y(:,1) = [randi([10 100],6,1)]

y =

    89     0     0     0     0     0

    60    37     0     0     0     0

    66     0    56     0     0     0

    63     0     0    56     0     0

    28     0     0     0    84     0

    37     0     0     0     0    82

y(1,:) = [randi([10 100],6,1)]

y =

    52    30    86    27    30    25

    60    37     0     0     0     0

    66     0    56     0     0     0

    63     0     0    56     0     0

    28     0     0     0    84     0

    37     0     0     0     0    82

diary off

diary on

format compact

%Exercise 2

type multi

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

%This is a function designed to multiply 2 matrices

%A & B if possible. It performs an identical function to

%The MATLAB command A\*B.

[m,n]=size(A);

[s,t]=size(B);

CColumns=0;

CRows=0;

if n==s

for i=1:m

for j=1:t

C(i,j)=A(i,:)\*B(:,j);

CRows(i,j)=0;

CColumns(i,j)=0;

for k=1:n

CRows(i,j)=CRows(i,j)+A(i,k)\*B(k,j);

CColumns(i,j)=CColumns(i,j)+A(i,k)\*B(k,j);

end

end

end

else

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

C=[];

CRows=[];

CColumns=[];

end

end

%(a)

A=randi(10,2,3)

A =

10 9 5

5 2 10

B=magic(2)

B =

1 3

4 2

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

The dimensions of A and B disagree.

C =

[]

CRows =

[]

CColumns =

[]

A\*B

{\_Error using <a href="matlab:matlab.internal.language.introspective.errorDocCallback('mtimes')" style="font-weight:bold"> \* </a>

Inner matrix dimensions must agree.}\_

%(b)

A=magic(5)

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=ones(4,6)

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

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

The dimensions of A and B disagree.

C =

[]

CRows =

[]

CColumns =

[]

A\*B

{\_Error using <a href="matlab:matlab.internal.language.introspective.errorDocCallback('mtimes')" style="font-weight:bold"> \* </a>

Inner matrix dimensions must agree.}\_

%(c)

A=magic(4)

A =

16 2 3 13

5 11 10 8

9 7 6 12

4 14 15 1

B=ones(4,3)

B =

1 1 1

1 1 1

1 1 1

1 1 1

[C,CRows,CColumns]=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\*B

ans =

34 34 34

34 34 34

34 34 34

34 34 34

%(d)

A=ones(4)

A =

1 1 1 1

1 1 1 1

1 1 1 1

1 1 1 1

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

B =

2 0 0 0

0 3 0 0

0 0 4 0

0 0 0 5

[C,CRows,CColumns]=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

A\*B

ans =

2 3 4 5

2 3 4 5

2 3 4 5

2 3 4 5

%The outputs and expected error messages for the function [C,CRows,CColumns]=multi(A,B) and the built in MATLAB function A\*B are exactly the same.

diary off

diary on

format compact

% Exercise3

type givensrot

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

A=[cos(theta) -sin(theta) ; sin(theta) cos(theta)]

I=eye(n)

if mod(n,1)==0 && mod(i,1)==0 && mod(j,1)==0 && n>=2 && i>=1 && j>i && n>j

I([i j],[i j])=A

else

disp('Error. 1<=i<j<n and i,j,n must be integers')

I=[];

end

G=I

end

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

A =

0.0000 -1.0000

1.0000 0.0000

I =

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

Error. 1<=i<j<n and i,j,n must be integers

G =

[]

ans =

[]

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

A =

0.7071 -0.7071

0.7071 0.7071

I =

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

I =

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 =

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

ans =

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

givensrot(3,1,2,pi)

A =

-1.0000 -0.0000

0.0000 -1.0000

I =

1 0 0

0 1 0

0 0 1

I =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

G =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

ans =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

givensrot(3,1,2,pi)

A =

-1.0000 -0.0000

0.0000 -1.0000

I =

1 0 0

0 1 0

0 0 1

I =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

G =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

ans =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

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

A =

-1.0000 -0.0000

0.0000 -1.0000

I =

1 0 0

0 1 0

0 0 1

I =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

G =

-1.0000 -0.0000 0

0.0000 -1.0000 0

0 0 1.0000

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

% The Givens rotation matrix in part(3) rotates a vector through pi radians in the x\_1 x\_2 plane

% The outputs are representative of this rotation as e\_1 and e\_2 are both rotated by pi radians and e\_3 is unchanged since it has no components in the x\_1 x\_2 plane

diary off

diary on

format compact

%Exercise4

type Toeplitz

function A=Toeplitz(m,n,a)

A = zeros(m,n);

for i=1:m

for j=1:n

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

end

end

end

%1(a)

m=4;,n=3;,a=transpose([1:6])

a =

1

2

3

4

5

6

A=Toeplitz(m,n,a)

A =

3 2 1

4 3 2

5 4 3

6 5 4

%1(b)

m=3;,n=4;,a=randi(10,6,1)

a =

9

10

2

10

7

1

A=Toeplitz(m,n,a)

A =

10 2 10 9

7 10 2 10

1 7 10 2

%1(c)

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

a =

0

0

0

1

2

3

4

A=Toeplitz(m,n,a)

A =

1 0 0 0

2 1 0 0

3 2 1 0

4 3 2 1

%2

m=5;,n=5;,a=randi([10 100],9,1)

a =

97

54

82

22

48

93

82

97

69

A=Toeplitz(m,n,a)

A =

48 22 82 54 97

93 48 22 82 54

82 93 48 22 82

97 82 93 48 22

69 97 82 93 48

%3

m=4;n=4;a=[0;0;0;1;0;0;0]

a =

0

0

0

1

0

0

0

A=Toeplitz(m,n,a)

A =

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

diary off

diary on

format compact

% Exercise5

type stochastic

function P = stochastic(A)

columnTest = any(A,1);

rowTest = any(A,2);

S1=sum(A,1);

S2=sum(A,2);

if any(columnTest == 0)

zeroColumn = true;

else

zeroColumn = false;

end

if any(rowTest == 0)

zeroRow = true;

else

zeroRow = false;

end

if (zeroRow == true && zeroColumn == true)

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

P =[];

else

if all(S1 == 1)

leftStoch = true;

else

leftStoch = false;

end

if all(S2 == 1)

rightStoch = true;

else

rightStoch = false;

end

if (rightStoch == true && leftStoch == true)

doubleStoch = true;

else

doubleStoch = false;

end

if (doubleStoch == true)

disp('The matrix is doubly stochastic')

P=A;

elseif (doubleStoch == false && rightStoch == true)

disp('The matrix is only right stochastic')

P=A;

elseif (doubleStoch == false && leftStoch == true)

disp('The matrix is only left stochastic')

P=A;

else

disp('Niether left nor right stochastic but can be scaled to scochastic')

S1

S2

if all(S1 ~= 0)

P = bsxfun(@rdivide, A, S1);

elseif all(S2 ~= 0)

P = bsxfun(@rdivide, A, S2);

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

stochastic(A)

The matrix is only right stochastic

ans =

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

stochastic(A)

The matrix is only left stochastic

ans =

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

stochastic(A)

Niether left nor right stochastic but can be scaled to scochastic

S1 =

0.5000 0 2.0000

S2 =

1.0000

1.0000

0.5000

ans =

0.5000 0 0.5000

0 0 1.0000

0 0 1.0000

A=A'

A =

0.5000 0 0

0 0 0

0.5000 1.0000 0.5000

stochastic(A)

Niether left nor right stochastic but can be scaled to scochastic

S1 =

1.0000 1.0000 0.5000

S2 =

0.5000

0

2.0000

ans =

0.5000 0 0

0 0 0

0.5000 1.0000 1.0000

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

A =

0.5000 0 0.5000

0 0.5000 0.5000

0.5000 0.5000 0

stochastic(A)

The matrix is doubly stochastic

ans =

0.5000 0 0.5000

0 0.5000 0.5000

0.5000 0.5000 0

A=magic(3)

A =

8 1 6

3 5 7

4 9 2

stochastic(A)

Niether left nor right stochastic but can be scaled to scochastic

S1 =

15 15 15

S2 =

15

15

15

ans =

0.5333 0.0667 0.4000

0.2000 0.3333 0.4667

0.2667 0.6000 0.1333

A=magic(3)

A =

8 1 6

3 5 7

4 9 2

stochastic(A)

Niether left nor right stochastic but can be scaled to scochastic

S1 =

15 15 15

S2 =

15

15

15

ans =

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

stochastic(A)

Niether left nor right stochastic but can be scaled to scochastic

S1 =

1 2 3

S2 =

1

2

3

ans =

1 0 0

0 1 0

0 0 1

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

stochastic(A)

A is not stochastic and cannot be scaled to stochastic

ans =

[]

diary off

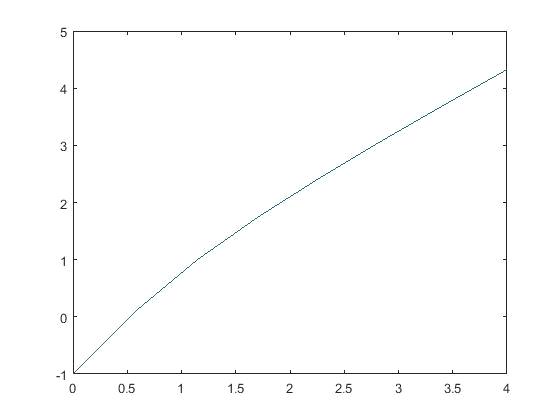
diary on

format compact

%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

type newtons

function root=newtons(N,x)

format long

for i=1:N

x=x-((atan(x)+(x-1))./(1/(x^2+1)+1))

end

root=x;

end

N=5

N =

5

x=0.6

x =

0.600000000000000

root=newtons(N,x)

x =

0.519080287979663

x =

0.520268738095731

x =

0.520268992719579

x =

0.520268992719590

x =

0.520268992719590

root =

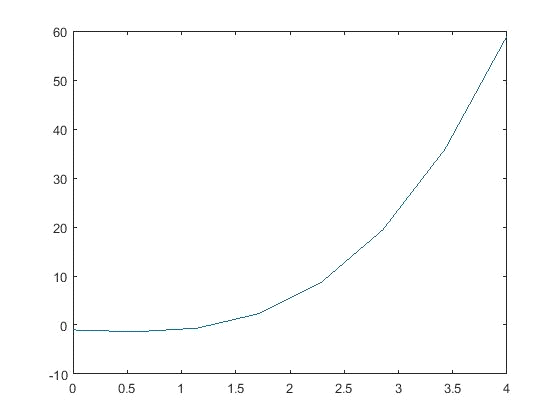
0.520268992719590

%The first iteration is close to the root. With each iteration accuracy is improved until the 4th iteration when it seems to have found the root to 8 decimal places. The approximate root is .52026899

x=linspace(0,4,8)

y=x.^3-x-1;

plot(x,y)



syms x

f=x^3-x-1

f =

x^3 - x - 1

g=diff(f)

g =

3\*x^2 - 1

format compact

type newtons\_1

function root=newtons\_1(N,x)

format long

for i=1:N

x=x-(x^3-x-1)/(3\*x^2-1)

end

root=x;

end

%1

N=5

N =

5

x=1.5

x =

1.5000

root=newtons\_1(N,x)

x =

1.347826086956522

x =

1.325200398950907

x =

1.324718173999054

x =

1.324717957244790

x =

1.324717957244746

root =

1.324717957244746

%2

x=1

x =

1

root=newtons\_1(N,x)

x =

1.500000000000000

x =

1.347826086956522

x =

1.325200398950907

x =

1.324718173999054

x =

1.324717957244790

root =

1.324717957244790

%3

x=0.6

x =

0.600000000000000

root=newtons\_1(N,x)

x =

17.899999999999984

x =

11.946802328608761

x =

7.985520351936208

x =

5.356909314795458

x =

3.624996032946096

root =

3.624996032946096

%4

x=0.57

x =

0.570000000000000

root=newtons\_1(N,x)

x =

-54.165454545454324

x =

-36.114292524925531

x =

-24.082094252098212

x =

-16.063387407817846

x =

-10.721483416797254

root =

-10.721483416797254

%The 1st and 2nd approximations give the root as 1.32471795, which seems to be correct as these approximations are close to the roots given. Values 3 and 4 give unreasonable second values and do not converge as they should.

N=10

N =

10

x=0.6

x =

0.600000000000000

root=newtons\_1(N,x)

x =

17.899999999999984

x =

11.946802328608761

x =

7.985520351936208

x =

5.356909314795458

x =

3.624996032946096

x =

2.505589190106631

x =

1.820129422319469

x =

1.461044109887682

x =

1.339323224262526

x =

1.324912867718656

root =

1.324912867718656

x=0.57

x =

0.570000000000000

root=newtons\_1(N,x)

x =

-54.165454545454324

x =

-36.114292524925531

x =

-24.082094252098212

x =

-16.063387407817846

x =

-10.721483416797254

x =

-7.165534466881882

x =

-4.801703812712865

x =

-3.233425234527273

x =

-2.193674204844573

x =

-1.496866569237556

root =

-1.496866569237556

x=0.6

x =

0.600000000000000

N=100

N =

100

root=newtons\_1(N,x)

root =

1.324717957244746

x=0.57

x =

0.570000000000000

root=newtons\_1(N,x)

root =

1.324717957244746

%For the initial value in part 3, x converges somewhat quickly, though slower than for approximations like 1.0 and 1.5. For part 4, it does eventually converge, but very slowly. This is likely because both values are near a minimum of the function. When f'(x) is near 0, Newtons equation gives numbers that are far from the actual value.

diary off