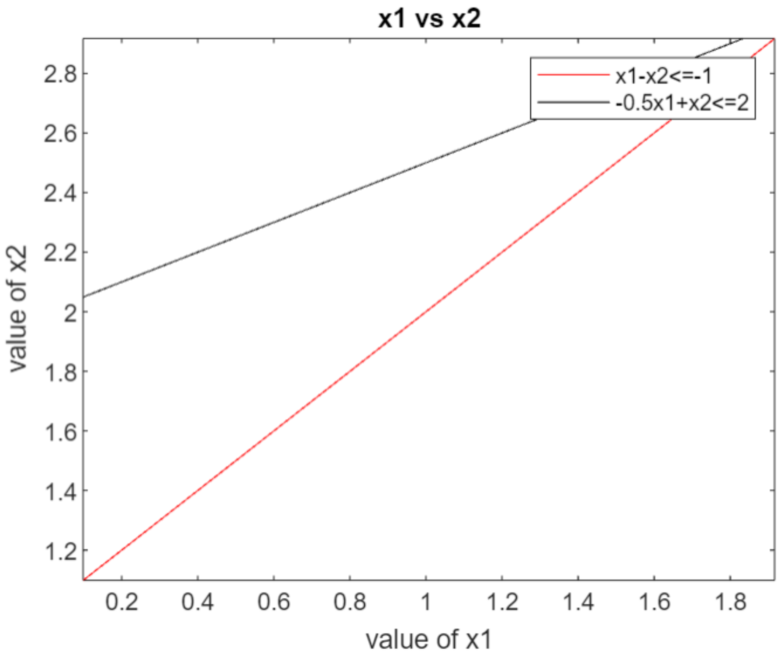
MAT2003 (Optimization Techniques) Final Lab

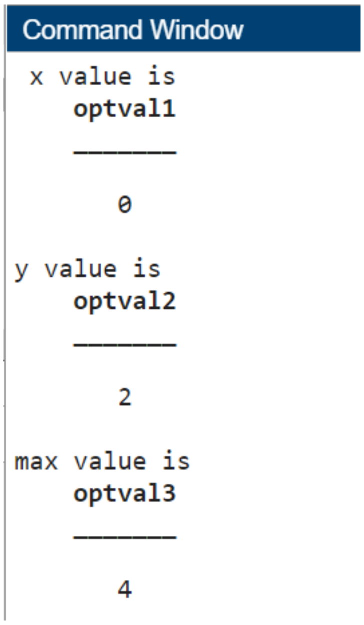
KHAN MOHD OWAIS RAZA
20BCD7138

Code-1 :-

```
%% KHAN MOHD OWAIS RAZA
%% 20BCD7138
%% MAT2003 FINAL LAB
%% CODE-1
clc
clear
format short
%%%%%% Stage 1: %%%%%%%
C= [-1 2];
A= [ 1 -1 ; -0.5 1 ];
b= [-1; 2];
%%%%%%%%% Stage 2: ploting the constraints in 2d graph%%%%%%%%%%
y1= 0:1: max(b);
X21= (b(1) - A(1,1) .*y1)./A(1,2);
X22= (b(2) - A(2,1) .*y1)./A(2,2);
X21= max(0,X21);
X22= max(0,X22);
plot(y1,X21, 'r', y1,X22,'k');
xlabel( 'value of x1');
ylabel( 'value of x2');
title('x1 vs x2');
legend('x1-x2<=-1', '-0.5x1+x2<=2')
%%%%%%%%%%%%Phase 3 Find the corner point i.e., pt of intersections
Cx1=find(y1==0);
C1 = find(X21==0);
Line1= [y1(:, [C1 Cx1]) ; X21(:, [C1 Cx1])]';
C2 = find(X22==0);
Line2= [y1(:, [C2 Cx1]) ; X22(:, [C2 Cx1])]';
Corpt= unique([Line1;Line2],'row');
%%%%%%%%Stage 4 Find the intersection points%%%%%%%%
HG=[0;0];
for i=1:size(A,1)
Hg1=A(i,: );
B1=b(i,: ) ;
for j=i+1: size(A,1)
Hg2= A(j,: );
B2= b (j, : );
Aa= [Hg1; Hg2];
Bb= [B1;B2];
Xx= Aa\Bb;
HG=[ HG Xx];
end
end
Pt = HG';
```

```
%%%%%%%%%%%%%%%% Stage 5 write all points, i.e., corner + intersection
points **
Allpt = [Pt; Corpt];
%%%%%%%%%%%%%%% stage 6: find the feasible region %%%%%%%%%
PT= constraint(Allpt);
PT= unique(PT,'row');
%%%%%%% stage 7***
for i=1: size(PT,1)
FX(i, : ) = sum (PT (i,: ).*C);
end
%%%%%%%% final Stage optimal Solution %%%%%%%%%%%
vert_fns = [PT FX];
[fxval, indfx]= max(FX);
optval = vert_fns(indfx, :);
optimal_bfs= array2table( optval);
disp(" x value is")
disp(optimal_bfs(1,[1]))
disp("y value is")
disp(optimal_bfs(1,[2]))
disp("max value is")
disp(optimal_bfs(1,[3]))

function X = constraint(X)
%%%%%%% write the first constraint here%%%%%%%
X1= X(: , 1);
X2= X(: , 2);
Cons1 = X1-1*X2+1;
H1=find(Cons1>0);
X(H1,: )=[];
%%%%%%% write the Second constraint here%%%%%%%
X1= X(: , 1);
X2= X(: , 2);
Cons2 = -0.5*X1+X2-2;
H2=find(Cons2>0);
X(H2,: )=[];
%%%%%%% write the Third constraint here%%%%%%%
X1= X(: , 1);
X2= X(: , 2);
end
```

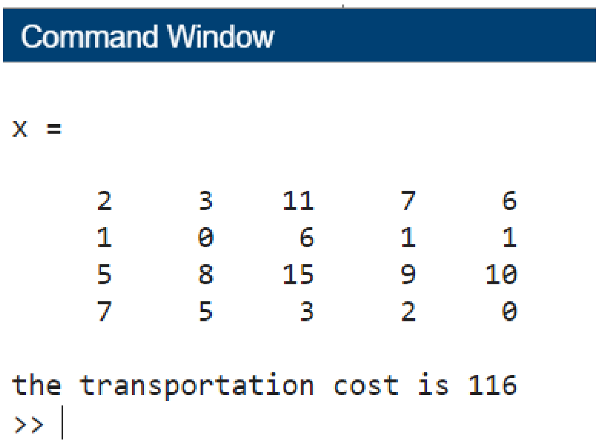




```
 x value is
    optval1
    _______

       0

y value is
    optval2
    _______

       2

max value is
    optval3
    _______

       4
```

Code-2 :-

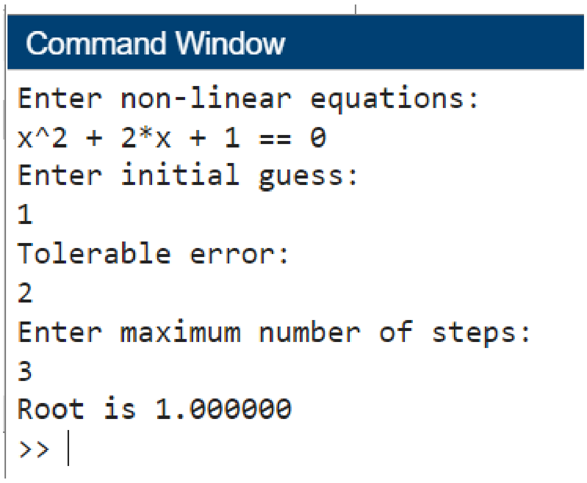
```
%% KHAN MOHD OWAIS RAZA
%% 20BCD7138
%% MAT2003 FINAL LAB
%% CODE-2
clc;
clear all;
close all;
x=[2 3 11 7 6;1 0 6 1 1;5 8 15 9 10;7 5 3 2 0]
[m n]=size(x);
x1=zeros(m,n);
sumc=0;
sumr=0;
for i=1:m-1
sumc=sumc+x(i,n);
end
for j=1:n-1
sumr=sumr+x(m,j);
end
if(sumc == sumr)
for i=1:m
for j=1:n
x11=min(x(i,n),x(m,j));
x1(i,j)=x11;
x(i,n)=x(i,n)-x11;
x(m,j)=x(m,j)-x11;
end
end
else disp('unbalanced transportation');
end
xre=0;
for i=1:m-1
for j=1:n-1
xre=xre+(x(i,j).*x1(i,j));
end
end
disp(['the transportation cost is ',num2str(xre)]);
```



```
x =

     2     3    11     7     6
     1     0     6     1     1
     5     8    15     9    10
     7     5     3     2     0

the transportation cost is 116
>> 
```

Code-3 :-

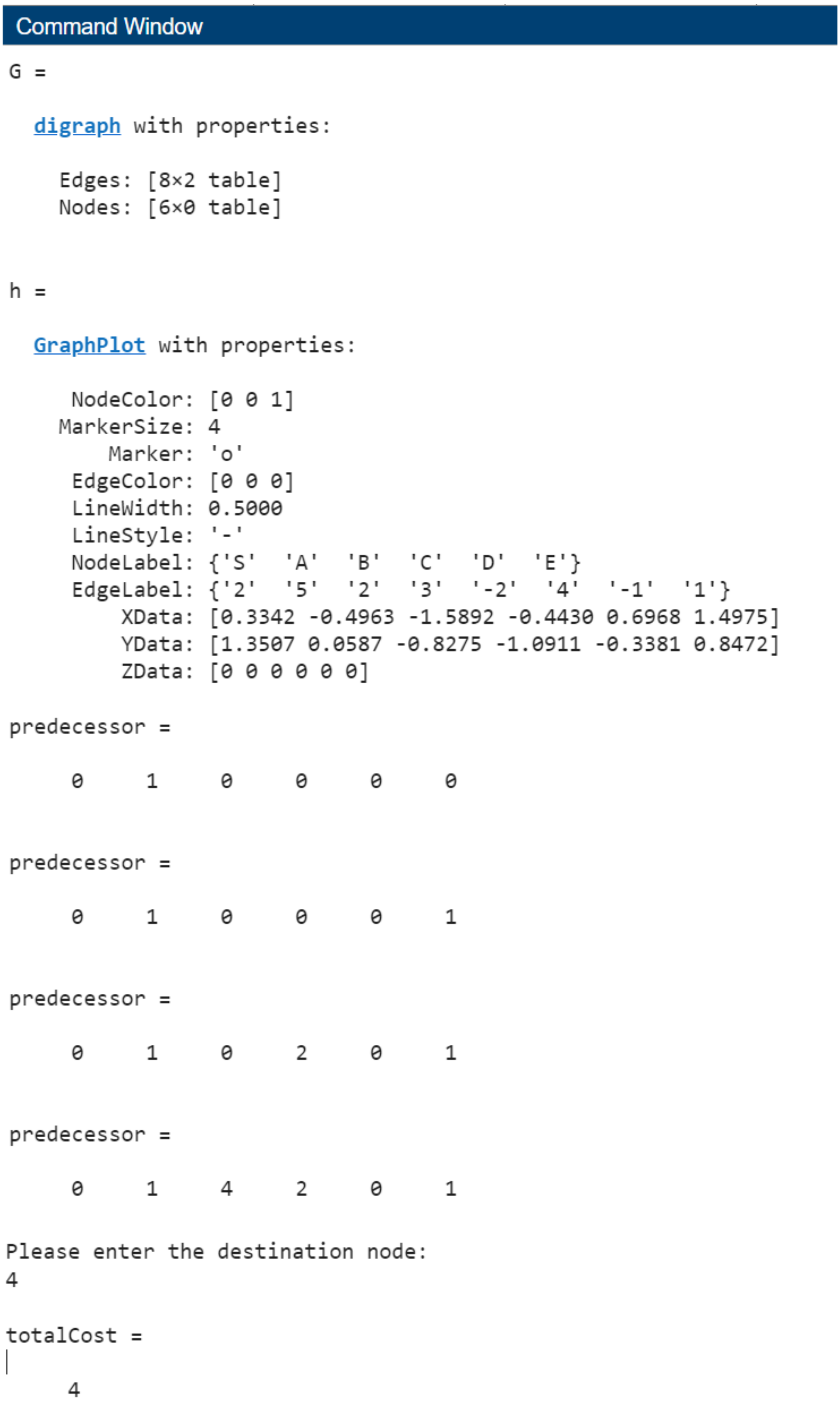
```
%% KHAN MOHD OWAIS RAZA
%% 20BCD7138
%% MAT2003 FINAL LAB
%% CODE-3
clc
clear all;
% Setting x as symbolic variable
syms x;
% Input Section
y = input('Enter non-linear equations: ');
a = input('Enter initial guess: ');
e = input('Tolerable error: ');
N = input('Enter maximum number of steps: ');
% Initializing step counter
step = 1;
% Finding derivate of given function
g = diff(y,x);
% Finding Functional Value
fa = eval(subs(y,x,a));
while abs(fa)> e
fa = eval(subs(y,x,a));
ga = eval(subs(g,x,a));
if ga == 0
disp('Division by zero.');
break;
end
b = a - fa/ga;
fprintf('step=%d\ta=%f\tf(a)=%f\n',step,a,fa);
a = b;
if step>N
disp('Not convergent');
break;
end
step = step + 1;
end
fprintf('Root is %f\n', a);
```



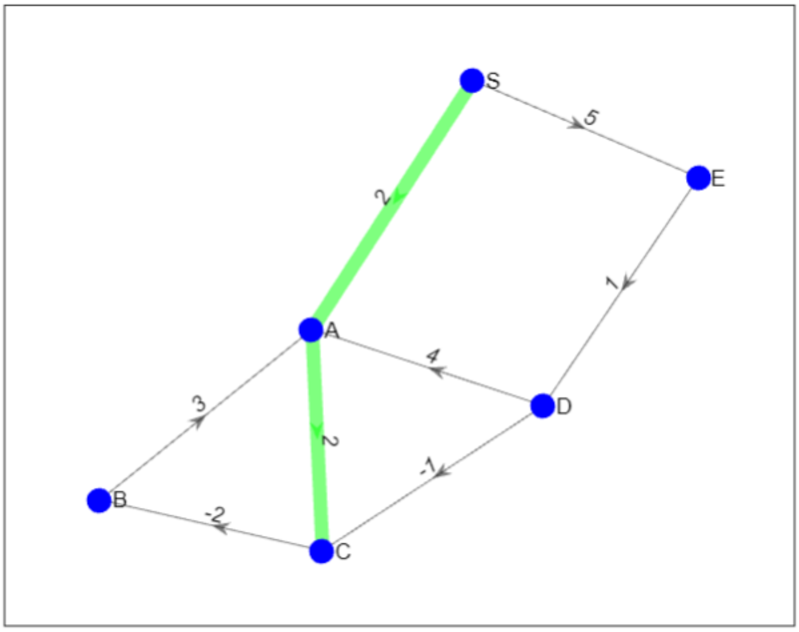
```
Enter non-linear equations:
x^2 + 2*x + 1 == 0
Enter initial guess:
1
Tolerable error:
2
Enter maximum number of steps:
3
Root is 1.000000
>>
```

Code-4 :-

```
%% KHAN MOHD OWAIS RAZA
%% 20BCD7138
%% MAT2003 FINAL LAB
%% CODE-4
N=6;
e_num=8;
s=1;
w=[2 5 2 3 -2 -1 4 1];
m=[1 1 2 3 4 5 5 6 ];
n=[2 6 4 2 3 4 2 5 ];
names={'S','A','B','C','D','E'};
G=digraph(m,n,w)
h=plot(G,'EdgeLabel',G.Edges.Weight,'Nodelabel',names,'EdgeColor','k','NodeColor','b')
h.MarkerSize=8;
S=sparse(m',n',w');
distance(1:N)=Inf;
distance(s)=0;
predecessor(1:N)=0;
for i = 1 : N - 1
for j = 1 : e_num
v = n(j);
u = m(j);
t = distance(u) + w(j);
if (t < distance(v) )
distance(v) = t;
predecessor(v) = u
end
end
end

for j = 1 : e_num
u = m(j);
v = n(j);
if ( distance(u) + w(j) < distance(v) )
fprintf ( 1, '\n' );
fprintf ( 1, 'BELLMAN_FORD - Fatal error!\n' );
fprintf ( 1, ' Graph contains a cycle with negative weight.\n' );
error ( 'BELLMAN_FORD - Fatal error!' );
end
end
for i=1:(N-1)
d=input('Please enter the destination node:');
totalCost = distance(d)
TR=shortestpathtree(G,1,d);
p=plot(G,'EdgeLabel',G.Edges.Weight,'Nodelabel',names,'EdgeColor','k','NodeColor','b')
p.MarkerSize=8;
highlight(p,TR,'EdgeColor','g','LineWidth',5);
end
```



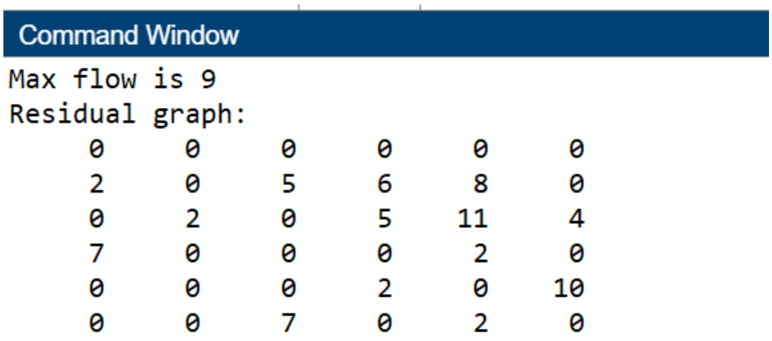
```
G =

  digraph with properties:

    Edges: [8×2 table]
    Nodes: [6×0 table]


h =

  GraphPlot with properties:

     NodeColor: [0 0 1]
    MarkerSize: 4
        Marker: 'o'
     EdgeColor: [0 0 0]
     LineWidth: 0.5000
     LineStyle: '-'
     NodeLabel: {'S'  'A'  'B'  'C'  'D'  'E'}
     EdgeLabel: {'2'  '5'  '2'  '3'  '-2'  '4'  '-1'  '1'}
         XData: [0.3342 -0.4963 -1.5892 -0.4430 0.6968 1.4975]
         YData: [1.3507 0.0587 -0.8275 -1.0911 -0.3381 0.8472]
         ZData: [0 0 0 0 0 0]

predecessor =

     0     1     0     0     0     0


predecessor =

     0     1     0     0     0     1


predecessor =

     0     1     0     2     0     1


predecessor =

     0     1     4     2     0     1

Please enter the destination node:
4

totalCost =

     4
```



Code-5 :-

```
%% KHAN MOHD OWAIS RAZA
%% 20BCD7138
%% MAT2003 FINAL LAB
%% CODE-5
function fordfulkerson
clc; clear;
s = 1; t = 6; f = 0;
cap = [ 0 2 0 7 0 0 ;
0 0 7 6 8 0;
0 0 0 0 11 11;
0 0 5 0 4 0;
0 0 0 0 0 12;
0 0 0 0 0 0];
len = length(cap);
while true
p = findPath(cap);
if p(1) == 0, break; end
flow = max(max(cap));
for j = 2:length(p)
flow = min(flow,cap(p(j),p(j-1)));
end
for j = 2:length(p)
a = p(j); b = p(j-1);
cap(a,b) = cap(a,b) - flow;
cap(b,a) = cap(b,a) + flow;
end
f = f + flow;
end
disp(['Max flow is ' num2str(f)]);
disp('Residual graph:');
disp(cap);
function F = findPath(A)
q = zeros(1,len);
pred = zeros(1,len);
front = 1; back = 2;
```

```
pred(s) = s; q(front) = s;
while front ~= back
v = q(front);
front = front + 1;
for i = 1:len
if pred(i) == 0 && A(v,i) > 0
q(back) = i;
back = back + 1;
pred(i) = v;
end
end
end
path = zeros(1,len);
if pred(t) ~= 0
i = t; c = 1;
while pred(i) ~= i
path(c) = i;
c = c + 1;
i = pred(i);
end
path(c) = s;
path(c+1:len) = [];
end
F = path;
end
end
```



```
Max flow is 9
Residual graph:
     0     0     0     0     0     0
     2     0     5     6     8     0
     0     2     0     5    11     4
     7     0     0     0     2     0
     0     0     0     2     0    10
     0     0     7     0     2     0
```