UMA019

OPERATIONS RESEARCH

Submitted To: Dr.Navdeep Kailey

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

BY: Divija 102018056 CSBS-3

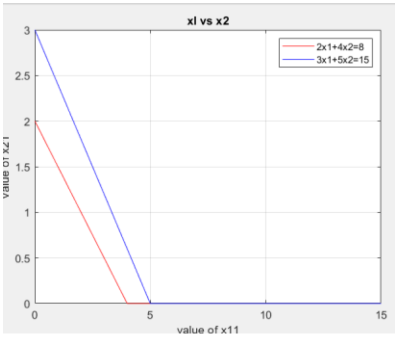
INDEX

ASSIGNMENT 1:

Question 1:

```
clc
clear all
format short
%Phase I: To input Parameter
A=[2 4;3 5];
B=[8; 15];
c = [3 2];
%Phage II: To Plot the lines on the graph
x1= 0:max (B)
x21= (B(1)-A(1,1).*x1)./A(1,2)
x22= (B(2)-A(2,1).*x1)./A(2,2)
x21=max(0,x21)
x22=max(0,x22)
plot (x1, x21, 'r' ,x1,x22,'b' )
title('xl vs x2')
xlabel ('value of x11');
ylabel ('value of x21');
legend('2x1+4x2=8', '3x1+5x2=15');
grid on
%Phase 3
cx1 = find(x1==0)
c1 = find(x21==0)
line1 = [x1(:,[c1 cx1]); x21(:,[c1 cx1]);]';
c2 = find(x22==0)
line2 = [x1(:,[c2 cx1]); x22(:,[c2 cx1]);]';
corpt = unique([line1;line2],'rows')
%phase 4
 pt = [0;0]
for i=1:size(A,1)
 A1 = A(i,:)
 B1 = B(i,:)
 for j=i+1:size(A,1)
 A2 = A(j,:);
 B2 = B(j,:);
 A4 = [A1;A2];
 B4 = [B1;B2];
 X = A4\B4;
 pt = [pt X]
 end
end
ptt = pt'
%phase 5
allpt = [ptt;corpt];
points = unique(allpt,"rows")
%phase 6
%find the feasible region
PT = constraint(points)
P = unique(PT,"rows")
%phase 7 : find value of objective func
%max z = x1+5x2
for i=1:size(P,1)
 fn(i,:)= (sum(P(i,:).*c))
end
values = [P fn]
%phase 8 : to find optimal sol
```

```
[Optval Optposition] = max(fn)
Optval = values(Optposition,:)
OPTIMAL_BFS = array2table(Optval);
OPTIMAL_BFS.Properties.VariableNames(1:size(Optval,2))={'x1', 'x2', 'z' }
```



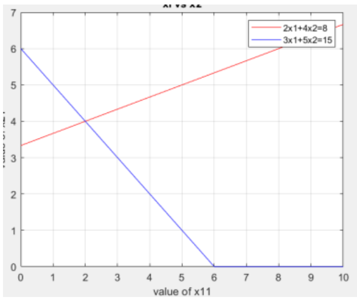

```
OPTIMAL_BFS =

  1×3 table

    x1    x2    z
    __    __    _

    0     3     6
```

Question 2:

```
clc
clear all
format short
%Phase I: To input Parameter
A=[2 4;3 5];
B=[8; 15];
c = [3 2];
%Phage II: To Plot the lines on the graph
x1= 0:max (B)
x21= (B(1)-A(1,1).*x1)./A(1,2)
x22= (B(2)-A(2,1).*x1)./A(2,2)
x21=max(0,x21)
x22=max(0,x22)
plot (x1, x21, 'r' ,x1,x22,'b' )
title('xl vs x2')
xlabel ('value of x11');
ylabel ('value of x21');
legend('2x1+4x2=8', '3x1+5x2=15');
grid on
%Phase 3
cx1 = find(x1==0)
c1 = find(x21==0)
line1 = [x1(:,[c1 cx1]); x21(:,[c1 cx1]);]';
c2 = find(x22==0)
```

```
line2 = [x1(:,[c2 cx1]); x22(:,[c2 cx1]);]';
corpt = unique([line1;line2],'rows')
%phase 4
 pt = [0;0]
for i=1:size(A,1)
 A1 = A(i,:)
 B1 = B(i,:)
 for j=i+1:size(A,1)
 A2 = A(j,:);
 B2 = B(j,:);
 A4 = [A1;A2];
 B4 = [B1;B2];
 X = A4\B4;
 pt = [pt X]
 end
end
ptt = pt'
%phase 5
allpt = [ptt;corpt];
points = unique(allpt,"rows")
%phase 6
%find the feasible region
PT = constraint1(points)
P = unique(PT,"rows")
%phase 7 : find value of objective func
%max z = x1+5x2
for i=1:size(P,1)
 fn(i,:)= (sum(P(i,:).*c))format rat
c = [3,2];
a = [2 4;3 5];
b = [8; 15];
p=max(b);
x1 = 0:1:max(b)
x12 = (b(1)
-a(1,1).*x1)./a(1,2)
x22 = (b(2)
-a(2,1).*x1)./a(2,2)
x12 = max(0,x12)
x22 = max(0,x22)
%%x32 = max(0,x32)
plot( x1,x12,'r',x1,x22,'b')
cx1=find(x1==0)
c1=find(x12==0)
line1 = [x1(:,[c1 cx1]); x12(:,[c1 cx1])]'
c2=find(x22==0);
line2= [x1(:,[c2 cx1]); x22(:,[c2 cx1])]'
corpt = unique([line1;line2],'rows')
pt =[0;0];
for i=1:size(a,1)
 a1 = a(i,:);
 b1 = b(i,:);
 for j =i+1:size(a,1)
 a2 = a(j,:);
 b2 = b(j,:);
 a4 = [a1;a2];
 b4 = [b1;b2];
 x = a4
\b4;
 pt = [pt x];
```

```
 end
end
pt = [pt x]
ptt = pt'
allpt=[ptt;corpt]
points=unique(allpt,'rows')
PT = constraint(points)
p=unique(PT,'rows')
for i=1:size(PT,1)
 fx(i,:)=sum(PT(i,:).*c)
end
P a g e | 4
vert_fns=[PT fx];
[fxval,indfx] = max(fx)
optval=vert_fns(indfx,:)
optimalbfs=array2table(optval)
optimalbfs.Properties.VariableNames(1:3) = {'x1', 'x2','z'}
end
values = [P fn]
%phase 8 : to find optimal sol
[Optval Optposition] = max(fn)
Optval = values(Optposition,:)
OPTIMAL_BFS = array2table(Optval);
OPTIMAL_BFS.Properties.VariableNames(1:size(Optval,2))={'x1', 'x2', 'z' }
```

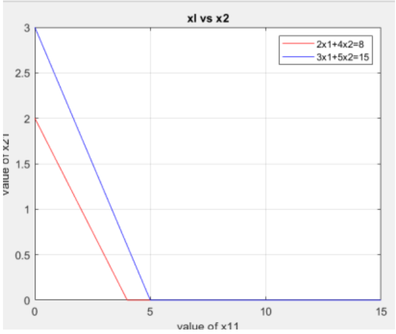


Question 4:

```
clc
clear all
format short
%Phase I: To input Parameter
A=[2 4;3 5];
B=[8; 15];
c = [3 2];
%Phage II: To Plot the lines on the graph
x1= 0:max (B)
x21= (B(1)-A(1,1).*x1)./A(1,2)
x22= (B(2)-A(2,1).*x1)./A(2,2)
x21=max(0,x21)
x22=max(0,x22)
plot (x1, x21, 'r' ,x1,x22,'b' )
title('xl vs x2')
xlabel ('value of x11');
ylabel ('value of x21');
legend('2x1+4x2=8', '3x1+5x2=15');
grid on
%Phase 3
cx1 = find(x1==0)
c1 = find(x21==0)
line1 = [x1(:,[c1 cx1]); x21(:,[c1 cx1]);]';
c2 = find(x22==0)
```

```
line2 = [x1(:,[c2 cx1]); x22(:,[c2 cx1]);]';
corpt = unique([line1;line2],'rows')
%phase 4
 pt = [0;0]
for i=1:size(A,1)
 A1 = A(i,:)
 B1 = B(i,:)
 for j=i+1:size(A,1)
 A2 = A(j,:);
 B2 = B(j,:);
 A4 = [A1;A2];
 B4 = [B1;B2];
 X = A4\B4;
 pt = [pt X]
 end
end
ptt = pt'
%phase 5
allpt = [ptt;corpt];
points = unique(allpt,"rows")
%phase 6
%find the feasible region
PT = constraint1(points)
P = unique(PT,"rows")
%phase 7 : find value of objective func
%max z = x1+5x2
for i=1:size(P,1)
 fn(i,:)= (sum(P(i,:).*c))format rat
c = [3,2];
a = [2 4;3 5];
b = [8; 15];
p=max(b);
x1 = 0:1:max(b)
x12 = (b(1)
-a(1,1).*x1)./a(1,2)
x22 = (b(2)
-a(2,1).*x1)./a(2,2)
x12 = max(0,x12)
x22 = max(0,x22)
%%x32 = max(0,x32)
plot( x1,x12,'r',x1,x22,'b')
cx1=find(x1==0)
c1=find(x12==0)
line1 = [x1(:,[c1 cx1]); x12(:,[c1 cx1])]'
c2=find(x22==0);
line2= [x1(:,[c2 cx1]); x22(:,[c2 cx1])]'
corpt = unique([line1;line2],'rows')
pt =[0;0];
for i=1:size(a,1)
 a1 = a(i,:);
 b1 = b(i,:);
 for j =i+1:size(a,1)
 a2 = a(j,:);
 b2 = b(j,:);
 a4 = [a1;a2];
 b4 = [b1;b2];
 x = a4
\b4;
 pt = [pt x];
```

```
 end
end
pt = [pt x]
ptt = pt'
allpt=[ptt;corpt]
points=unique(allpt,'rows')
PT = constraint(points)
p=unique(PT,'rows')
for i=1:size(PT,1)
 fx(i,:)=sum(PT(i,:).*c)
end
P a g e | 4
vert_fns=[PT fx];
[fxval,indfx] = max(fx)
optval=vert_fns(indfx,:)
optimalbfs=array2table(optval)
optimalbfs.Properties.VariableNames(1:3) = {'x1', 'x2','z'}
end
values = [P fn]
%phase 8 : to find optimal sol
[Optval Optposition] = max(fn)
Optval = values(Optposition,:)
OPTIMAL_BFS = array2table(Optval);
OPTIMAL_BFS.Properties.VariableNames(1:size(Optval,2))={'x1', 'x2', 'z' }
```



ASSIGNMENT 2:

Question 1:

```
clc
clear all
format short
c=[1,2]
a = [-1 1; 1 1]
b = [1; 2]

s = eye (size(a,1))
I = [0,0]
index= find(I==1)
s(index,index)= -s(index,index)
mat=[a s b]
obj=array2table(c);
obj.Properties.VariableNames(1:size(c,2))={'x_1','x_2'}
cons=array2table(mat);
cons.Properties.VariableNames(1:size(mat,2))={'x_1','x_2','s1','s2','b'}
n=size(a,2)
m=size(a,1)
if(n<m)
 disp("invalid")
else
 ans = nchoosek(n,m)
 pairs= nchoosek(1:n,m)
 sol = []
 for i = 1:ans
 y = zeros(n,1)
 X = a(:,pairs(i,:))\b
 if all(X>=0 & X ~= inf)

 Y(pairs(i,:))= X
 sol = [sol, y]
 end
 end
end
```

```
X =

    0.5000
    1.5000


Y =

    0.5000    1.5000
```

Question 2:

```
clc
```

```
clear all
format short
c = [1,3,7]
a = [1 1 0; 0 -1 1]
b = [1;0]
n=size(a,2)
m=size(a,1)
if(n<m)
 disp("invalid")
else
 ans = nchoosek(n,m)
 pairs= nchoosek(1:n,m)
 sol = []
 for i = 1:ans

 y = zeros(n,1)
 X = a(:,pairs(i,:))\b
 if all(X>=0 & X ~= inf)

 Y(pairs(i,:))= X
 sol = [sol, y]
 end
 end
end
```

Question 3:

```
clc
clear all
format short
% phase1: Input Parameter
A=[1 0 0 1 0 0 0; 0 1 0 0 1 0 0; -1 1 0 0 0 1 0; -1 0 2 0 0 0 1];
B=[4;4;6;4];
C=[-1 2 -1 0 0 0 0];
% phase2: Set of all Basic solutions
m=size(A,1);
n=size(A,2);
if (n>m)
 nCm=nchoosek(n,m);
 pair=nchoosek(1:n,m);
 sol=[];
for i=1:nCm
 y=zeros(n,1);
 temp=pair(i,:);
 P=A(:,temp);
 x=inv(P)*B;
if (x>=0 & x~=inf & x~=-inf)
 y(temp)=x;
 sol=[sol, y]
end
end
else
 error('nCm does not exist')
end
% phase3: Basic feasible solution
Z=C*sol
[Zmax , Zindex]=max(Z);
bfs1=sol(:,Zindex);
optimal_value=[bfs1' Zmax]
optimal_bfs=array2table(optimal_value);
```

```
optimal_bfs.Properties.VariableNames(1:size(optimal_bfs,2))={'x1', 'x_2' ,
'x_3','s_1' , 's_2', 's_3', 's_4', 'Z' }
fprintf('Optimal solution:');
disp(Zmax);
```

```
  1×8 table

    x1    x_2    x_3    s_1    s_2    s_3    s_4    Z
    __    ___    ___    ___    ___    ___    ___    _

    0      4      0      4      0      2      4     8

Optimal solution:     8
```

Question 4:

```
clc
clear all
format short
c = [1,3,7]
a = [1 1 0; 0 -1 1]
b = [1;0]
n=size(a,2)
m=size(a,1)
if(n<m)
 disp("invalid")
else
 ans = nchoosek(n,m)
 pairs= nchoosek(1:n,m)
 sol = []
 for i = 1:ans

 y = zeros(n,1)
 X = a(:,pairs(i,:))\b
 if all(X>=0 & X ~= inf)

 Y(pairs(i,:))= X
 sol = [sol, y]
 end
 end
end
```

```
X =

     1
     1


Y =

     1     1     1
```

ASSIGNMENT 3:

Question 1:

```
%% Max. z = x1 + 2x2,
%%subject to ? x1 + x2 ? 1, x1 + x2 ? 2,
%%x1, x2 ? 0.
clc
clear all
format short
noofvariables=2;
c = [1 2]
a = [-1 1; 1 1]
b = [1;2]
s= eye(size(a,1));
A = [a s b]
cost = zeros(1,size(A,2));
cost(1:noofvariables)=c;
bv = noofvariables+1:size(A,2)-1;
zjcj = cost(bv)*A -cost;
zcj = [zjcj; A];
simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))= {'x_1','x_2','s1','s2','b'}
run = true;
while(run)
 zc=zjcj(1:end-1);
 if any(zc<0);
 fprintf('bfs is not optimal')
 [Enter_val,pvt_col]=min(zc)
 if all(A(:,pvt_col)<=0)
 error('lpp is unbounded')
 else
 sol = A(:,end)
 col= A(:,pvt_col)
 for i=1:size(A,1)
 if(col(i)>0)
 ratio(i)=sol(i)/col(i)
 else
 ratio(i) = inf
 end
 end
 [leaving_variable,pvt_row]=min(ratio)
 end


 pvt_key = A(pvt_row, pvt_col)
 bv(pvt_row) = pvt_col
 A(pvt_row,:)=A(pvt_row,:)/pvt_key
 for i=1:size(A,1)
 if i~= pvt_row
 A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
 end
 end
 zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
 zcj1=[zjcj;A]
 table = array2table(zcj1)
 table.Properties.VariableNames(1:size(zcj1,2))= {'x_1','x_2','s1','s2','b'}
```

```
 BFS=zeros(1,size(A,2));
 BFS(bv)=A(:,end);
 BFS(end)=sum(BFS.*cost);
 CurrentBFS=array2table(BFS);

CurrentBFS.Properties.VariableNames(1:size(CurrentBFS,2))={'x1','x2','s1','s2','So
l'}
 else
 run = false;
 fprintf("the current bfs is optimal")
 end
end
```

```
CurrentBFS =

  1×5 table

    x1     x2     s1    s2    Sol

    ___    ___    __    __    ___

    0.5    1.5    0     0     3.5
```

Question 2:

```
%%M ax. z = 4x1 + 6x2 + 3x3 + x4,
%%subject to
%%x1 + 4x2 + 8x3 + 6x4 ? 11, 4x1 + x2 + 2x3 + x4 ? 7,
%%2x1 + 3x2 + x3 + 2x4 ? 2, x1, x2, x3 ? 0.
clc
clear all

format short
noofvariables=4;
c = [4 6 3 1]
a = [1 4 8 6; 4 1 2 1; 2 3 1 2]
b = [11; 7; 2]
s= eye(size(a,1));
A = [a s b]
cost = zeros(1,size(A,2));
cost(1:noofvariables)=c;
bv = noofvariables+1:size(A,2)-1;
zjcj = cost(bv)*A -cost;
zcj = [zjcj; A];
simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))=
{'x_1','x_2','x_3','x_4','s1','s2','s3','b'}
run = true;
while(run)
 zc=zjcj(1:end-1);
 if any(zc<0);
 fprintf('bfs is not optimal')
 [Enter_val,pvt_col]=min(zc)
 if all(A(:,pvt_col)<=0)
 error('lpp is unbounded')
 else
 sol = A(:,end)
```

```
col= A(:,pvt_col)
for i=1:size(A,1)
if(col(i)>0)
ratio(i)=sol(i)/col(i)
else
ratio(i) = inf
end
end
[leaving_variable,pvt_row]=min(ratio)
end

pvt_key = A(pvt_row, pvt_col)
bv(pvt_row) = pvt_col
A(pvt_row,:)=A(pvt_row,:)/pvt_key
for i=1:size(A,1)
if i~= pvt_row
A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
end
end
zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
zcj1=[zjcj;A]

table = array2table(zcj1)
table.Properties.VariableNames(1:size(zcj1,2))=
{'x_1','x_2','x_3','x_4','s1','s2','s3','b'}
BFS=zeros(1,size(A,2));
BFS(bv)=A(:,end);
BFS(end)=sum(BFS.*cost);
CurrentBFS=array2table(BFS);

CurrentBFS.Properties.VariableNames(1:size(CurrentBFS,2))={'x_1','x_2','x_3','x_4'
,'s1','s2','s3','b'}
else
run = false;
fprintf("the current bfs is optimal")
end
end
```

```
CurrentBFS =

  1×8 table
```

x_1	x_2	x_3	x_4	s1	s2	s3	b
0.33333	0	1.3333	0	0	3	0	5.3333

```
the current bfs is optimal>>
```

Question 3:

```
clc
clear all
format short
noofvariables=7;
c = [0 0 0 3/4 -20 1/2 -6]
a = [1 0 0 1/4 -8 -1 9; 0 1 0 1/2 -12 -1/6 3; 0 0 1 0 0 1 0]
b = [0; 0; 1]
%%s= eye(size(a,1));
A = [a b]
cost = zeros(1,size(A,2));

cost(1:noofvariables)=c;
%%bv = noofvariables+1:size(A,2)-1;
bv = 1:3
zjcj = cost(bv)*A -cost;
zcj = [zjcj; A];
simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))=
{'x_1','x_2','x_3','x_4','x_5','x_6','x_7','b'}
run = true;
while(run)
 zc=zjcj(1:end-1);
 if any(zc<0);
 fprintf('bfs is not optimal')
 [Enter_val,pvt_col]=min(zc)
 if all(A(:,pvt_col)<=0)
 error('lpp is unbounded')
 else
 sol = A(:,end)
 col= A(:,pvt_col)
 for i=1:size(A,1)
 if(col(i)>0)
 ratio(i)=sol(i)/col(i)
 else
 ratio(i) = inf
 end
 end
 [leaving_variable,pvt_row]=min(ratio)
 end

 pvt_key = A(pvt_row, pvt_col)
 bv(pvt_row) = pvt_col
 A(pvt_row,:)=A(pvt_row,:)/pvt_key
 for i=1:size(A,1)
 if i~= pvt_row
 A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
 end
 end
 zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
 zcj1=[zjcj;A]
 table = array2table(zcj1)
 table.Properties.VariableNames(1:size(zcj1,2))=
{'x_1','x_2','x_3','x_4','x_5','x_6','x_7','b'}
 BFS=zeros(1,size(A,2));
 BFS(bv)=A(:,end);
 BFS(end)=sum(BFS.*cost);
 CurrentBFS=array2table(BFS);
```

```
CurrentBFS.Properties.VariableNames(1:size(CurrentBFS,2))={'x_1','x_2','x_3','x_4'
,'x_5','x_6','x_7','Sol'}
 else
 run = false;
 fprintf("the current bfs is optimal")
 end
end
```

```
CurrentBFS =

  1×8 table
```

x_1	x_2	x_3	x_4	x_5	x_6	x_7	Sol
0.91667	0	0	0.33333	0	1	0	0.75

```
the current bfs is optimal>>
```

ASSIGNMENT 4:

QUESTION 1:

```
%code for big M
clc
clear
format short
x_1 = 10000
cost=[-3 -5 0 0 -x_1 -x_1 0]
b=[3;
 2]
a= [1 3 -1 0 1 0 ;
    1 1 0 -1 0 1 ]
A=[a b]

noofvar=2;
%bv are index of starting basic variables
%starting basic variable n + 1 se start hokr size -1 wale honge
bv=noofvar+3:1:size(A,2)-1

%cost(bv) gives cost of basic variables
zjcj=cost(bv)*A-cost;
zcj = [zjcj; A]

simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))=
{'x_1','x_2','s1','s2','A1','A2','b'}
run= true
while run
    zc=zjcj(1:end-1)
if any(zc<0);
    fprintf('bfs is not optimal')
    %to find minimum with its position(most negative entering variable)
    [Enter_val,pvt_col]=min(zc)
    fprintf('the most neagtive element in Zrow is %d corresponding to column
%d',Enter_val,pvt_col)

    if all(A(:,pvt_col)<=0)
        error('lpp is unbounded')
    else
        %to find leaving variable
        sol = A(:,end)
        col= A(:,pvt_col)

        %now we will find the minimum ratio between pivot col and sol col
        for i=1:size(A,1)
            if(col(i)>0)
                ratio(i)=sol(i)/col(i)
            else
                %inf stands for infinity (MAX)
                ratio(i) = inf
            end
        end
        [leaving_variable,pvt_row]=min(ratio)
    end
        %to display new basic variables in next iteration
        bv(pvt_row) = pvt_col;
        disp(bv)
        % to indentify pivot element
```

```
        pvt_key = A(pvt_row, pvt_col)
        %updating table for next iteration
        %updating pivot row first
        A(pvt_row,:)=A(pvt_row,:)./pvt_key

        % for updation of other rows
         for i=1:size(A,1)
             % ab isme pivot row nhi leni
            if i~= pvt_row
                A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
            end
         end
         %now updating the z row
         zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
         zcj1=[zjcj;A]
         % to print the table
        table = array2table(zcj1)
        table.Properties.VariableNames(1:size(zcj1,2))=
{'x_1','x_2','s1','s2','A1','A2','b'}

        BFS = zeros(1,size(A,2));
        BFS(bv) = A(:,end)
        % to find objective func wala sol
        BFS(end) = sum(BFS.*cost)

        curr_Bfs = array2table(BFS);
        curr_Bfs.Properties.VariableNames(1:size(curr_Bfs,2)) =
{'x_1','x_2','s1','s2','A1','A2','b'}
else
    run = false;
end
end
```

curr_Bfs =

1×7 table

x_1	x_2	s1	s2	A1	A2	b
1.5	0.5	0	0	0	0	-7

QUESTION 2:

```
%code for big M
clc
clear
format short
x_1 = 10000
cost=[-12 -10 0 0 0 -x_1 -x_1 -x_1 0]
b=[10;
 30;
```

```
 8]
a= [5 1 -1 0 0 1 0 0;
 6 5 0 -1 0 0 1 0;
 1 4 0 0 -1 0 0 1 ]
A=[a b]

noofvar=5;
%bv are index of starting basic variables
%starting basic variable n + 1 se start hokr size -1 wale honge
bv=noofvar+1:1:size(A,2)-1

%cost(bv) gives cost of basic variables
zjcj=cost(bv)*A-cost;
zcj = [zjcj; A]

simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))=
{'x_1','x_2','s1','s2','s3','A1','A2','A3','b'}
run= true
while run
    zc=zjcj(1:end-1)
if any(zc<0);
    fprintf('bfs is not optimal')
    %to find minimum with its position(most negative entering variable)
    [Enter_val,pvt_col]=min(zc)
    fprintf('the most neagtive element in Zrow is %d corresponding to column
%d',Enter_val,pvt_col)

    if all(A(:,pvt_col)<=0)
        error('lpp is unbounded')
    else
        %to find leaving variable
        sol = A(:,end)
        col= A(:,pvt_col)

        %now we will find the minimum ratio between pivot col and sol col
        for i=1:size(A,1)
            if(col(i)>0)
                ratio(i)=sol(i)/col(i)
            else
                %inf stands for infinity (MAX)
                ratio(i) = inf
            end
        end
        [leaving_variable,pvt_row]=min(ratio)
    end
        %to display new basic variables in next iteration
        bv(pvt_row) = pvt_col;
        disp(bv)
        % to indentify pivot element
        pvt_key = A(pvt_row, pvt_col)
        %updating table for next iteration
        %updating pivot row first
        A(pvt_row,:)=A(pvt_row,:)./pvt_key

        % for updation of other rows
         for i=1:size(A,1)
             % ab isme pivot row nhi leni
            if i~= pvt_row
```

```
                A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
            end
         end
         %now updating the z row
         zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
         zcj1=[zjcj;A]
         % to print the table
        table = array2table(zcj1)

table.Properties.VariableNames(1:size(zcj1,2))={'x_1','x_2','s1','s2','s3','A1','A
2','A3','b'}

        BFS = zeros(1,size(A,2));
        BFS(bv) = A(:,end)
        % to find objective func wala sol
        BFS(end) = sum(BFS.*cost)

        curr_Bfs = array2table(BFS);
        curr_Bfs.Properties.VariableNames(1:size(curr_Bfs,2)) =
{'x_1','x_2','s1','s2','s3','A1','A2','A3','b'}
else
    run = false;
end
end
```

curr_Bfs =

1×9 table

x_1	x_2	s1	s2	s3	A1	A2	A3	b
4.2105	0.94737	12	0	0	0	0	0	-60

--

Question 3:

```
%code for big M
clc
clear
format short
x_1 = 10000
cost=[3 2 0 0 -x_1 0]
b=[2;
 3;
 1]
a= [1 1 1 0 0;
 1 3 0 1 0;
 1 -1 0 0 1]
A=[a b]
noofvar=2;
%bv are index of starting basic variables
%starting basic variable n + 1 se start hokr size -1 wale honge
bv=noofvar+1:1:size(A,2)-1
```

```
%cost(bv) gives cost of basic variables
zjcj=cost(bv)*A-cost;
zcj = [zjcj; A]

simptable = array2table(zcj)
simptable.Properties.VariableNames(1:size(zcj,2))=
{'x_1','x_2','s1','s2','A3','b'}

run= true
while run
    zc=zjcj(1:end-1)
if any(zc<0);
    fprintf('bfs is not optimal')
    %to find minimum with its position(most negative entering variable)
    [Enter_val,pvt_col]=min(zc)
    fprintf('the most neagtive element in Zrow is %d corresponding to column
%d',Enter_val,pvt_col)

    if all(A(:,pvt_col)<=0)
        error('lpp is unbounded')
    else
        %to find leaving variable
        sol = A(:,end)
        col= A(:,pvt_col)

        %now we will find the minimum ratio between pivot col and sol col
        for i=1:size(A,1)
            if(col(i)>0)
                ratio(i)=sol(i)/col(i)
            else
                %inf stands for infinity (MAX)
                ratio(i) = inf
            end
        end
        [leaving_variable,pvt_row]=min(ratio)
    end
        %to display new basic variables in next iteration
        bv(pvt_row) = pvt_col;
        disp(bv)
        % to indentify pivot element
        pvt_key = A(pvt_row, pvt_col)
        %updating table for next iteration
        %updating pivot row first
        A(pvt_row,:)=A(pvt_row,:)./pvt_key

        % for updation of other rows
         for i=1:size(A,1)
             % ab isme pivot row nhi leni
            if i~= pvt_row
                A(i,:)= A(i,:)- A(i,pvt_col).*A(pvt_row,:)
            end
         end
         %now updating the z row
         zjcj = zjcj - zjcj(pvt_col).* A(pvt_row,:)
         zcj1=[zjcj;A]
         % to print the table
        table = array2table(zcj1)
```

```
        table.Properties.VariableNames(1:size(zcj1,2))=
{'x_1','x_2','s1','s2','A3','b'}


        BFS = zeros(1,size(A,2));
        BFS(bv) = A(:,end)
        % to find objective func wala sol
        BFS(end) = sum(BFS.*cost)

        curr_Bfs = array2table(BFS);
        curr_Bfs.Properties.VariableNames(1:size(curr_Bfs,2)) =
{'x_1','x_2','s1','s2','A3','b'}

else
    run = false;
end
end
```

1×6 table

x_1	x_2	s1	s2	A3	b
1.5	0.5	0	0	0	5.5

ASSIGNMENT 5:

QUESTION 1:

```
clc
clear all
format short
Variables= {'x1','x2','s1','s2','a1','a2','sol'};
OVariables={'x1','x2','s1','s2','sol'};

Origc=[-3 -5 0 0  -1 -1 0]
A=[1 3 -1 0 1 0 3;
   1 1 0 -1 0 1 2]
bv=[5 6]

%%phase-1
Cost= [0 0 0 0 -1 -1 0]
startbv=find(Cost<0);
%calling of function to find optimal table for arbitary z function
[BFS,A]=simp(A,bv,Cost,Variables);


%%phase-2
%dropping the artifical variable in phase 2
A(:,startbv)=[];
Origc(startbv)=[];
%calling of function for optimal sol for original functiom
[optbfs,optA]=simp(A,BFS,Origc,OVariables);

final_bfs=zeros(1,size(A,2));
final_bfs(optbfs)=optA(:,end)
final_bfs(end)=sum(final_bfs.*Origc)
optimalbfs=array2table(final_bfs)
optimalbfs.Properties.VariableNames(1:size(optimalbfs,2))=OVariables
```

```
optimalbfs =

  1×5 table

    x1     x2     s1    s2    sol
    ___    ___    __    __    ___

    1.5    0.5    0     0     -7
```

QUESTION 2:

```
clc
clear all
max = 0;
Variables = {'x1', 'x2', 's1', 's2', 's3', 'A1', 'A2', 'A3', 'sol'}
OVariables = {'x1', 'x2', 's1', 's2', 's3', 'sol'}
```

```
info = [5 1 -1 0 0 1 0 0 10;
 6 5 0 -1 0 0 1 0 30;
 1 4 0 0 -1 0 0 1 8]
origC = [-12 -10 0 0 0 -1 -1 -1 0]
bv4 = [6 7 8]
A = info
cost = [0 0 0 0 0 -1 -1 -1 0]
zjcj = (cost(bv4)*A) - cost
[bv2,A] = simp(A,bv4,cost,Variables)
if bv2==0
 fprintf('\n UNBOUNDED SOLUTION ')
else
%PHASE2
 A(:,bv4) = []
 origC(:,bv4)=[]
 [ opt_bfs,optA] = simp(A,bv2,origC,OVariables)
 if (opt_bfs == 0)
 fprintf('\n UNBOUNDED SOLUTION ')
 else
 bfss = zeros(1,size(A,2))
 bfss(opt_bfs) = A(:,end)
 %bfss(end) = sum(bfss.*origC)
 if max==1
 bfss(end) = sum(bfss.*origC)
 else
 bfss(end) = -sum(bfss.*origC)
 end
 currentbfs = array2table(bfss)

 currentbfs.Properties.VariableNames(1:size(currentbfs,2)) = OVariables
 end
end
```

currentbfs =

1×6 table

x1	x2	s1	s2	s3	sol
4.2105	0.94737	12	0	0	60

QUESTION :3

```
clc
clear all
format short
Variables= {'x1','x2','s1','s2','a1','sol'};
OVariables={'x1','x2','s1','s2','sol'};


Origc=[3 2 0 0 -1 0]
A=[1 1 1 0 0 2;1 3 0 1 0 3; 1 -1 0 0 1 1]
bv=[3 4 5]

%%phase-1
Cost= [0 0 0 0 -1 0]
startbv=find(Cost<0);
%calling of function to find optimal table for arbitary z function
[BFS,A]=simp(A,bv,Cost,Variables);
```

```
%%phase-2
%dropping the artifical variable in phase 2
A(:,startbv)=[];
Origc(startbv)=[];
%calling of function for optimal sol for original functiom
[optbfs,optA]=simp(A,BFS,Origc,OVariables);

final_bfs=zeros(1,size(A,2));
final_bfs(optbfs)=optA(:,end)
final_bfs(end)=sum(final_bfs.*Origc)
optimalbfs=array2table(final_bfs)
optimalbfs.Properties.VariableNames(1:size(optimalbfs,2))=OVariables
```

```
optimalbfs =

  1×5 table

    x1     x2     s1    s2    sol
    ___    ___    __    __    ___

    1.5    0.5    0     0     5.5

>>
```

ASSIGNMENT 6:

QUESTION 1:

```
clc
clear all
format short
%Phase I: To input Parameter
A=[2 4;3 5];
B=[8; 15];
c = [3 2];
%Phage II: To Plot the lines on the graph
x1= 0:max (B)
x21= (B(1)-A(1,1).*x1)./A(1,2)
x22= (B(2)-A(2,1).*x1)./A(2,2)
x21=max(0,x21)
x22=max(0,x22)
plot (x1, x21, 'r' ,x1,x22,'b' )
title('xl vs x2')
xlabel ('value of x11');
ylabel ('value of x21');
legend('2x1+4x2=8', '3x1+5x2=15');
grid on
%Phase 3
cx1 = find(x1==0)
c1 = find(x21==0)
line1 = [x1(:,[c1 cx1]); x21(:,[c1 cx1]);]';
c2 = find(x22==0)
line2 = [x1(:,[c2 cx1]); x22(:,[c2 cx1]);]';
corpt = unique([line1;line2],'rows')
%phase 4
 pt = [0;0]
for i=1:size(A,1)
 A1 = A(i,:)
 B1 = B(i,:)
 for j=i+1:size(A,1)
 A2 = A(j,:);
 B2 = B(j,:);
 A4 = [A1;A2];
 B4 = [B1;B2];
 X = A4\B4;
 pt = [pt X]
 end
end
ptt = pt'
%phase 5
allpt = [ptt;corpt];
points = unique(allpt,"rows")
%phase 6
%find the feasible region
PT = constraint1(points)
P = unique(PT,"rows")
%phase 7 : find value of objective func
%max z = x1+5x2
for i=1:size(P,1)
 fn(i,:)= (sum(P(i,:).*c))format rat
c = [3,2];
a = [2 4;3 5];
```

```
b = [8; 15];
p=max(b);
x1 = 0:1:max(b)
x12 = (b(1)
-a(1,1).*x1)./a(1,2)
x22 = (b(2)
-a(2,1).*x1)./a(2,2)
x12 = max(0,x12)
x22 = max(0,x22)
%%x32 = max(0,x32)
plot( x1,x12,'r',x1,x22,'b')
cx1=find(x1==0)
c1=find(x12==0)
line1 = [x1(:,[c1 cx1]); x12(:,[c1 cx1])]'
c2=find(x22==0);
line2= [x1(:,[c2 cx1]); x22(:,[c2 cx1])]'
corpt = unique([line1;line2],'rows')
pt =[0;0];
for i=1:size(a,1)
 a1 = a(i,:);
 b1 = b(i,:);
 for j =i+1:size(a,1)
 a2 = a(j,:);
 b2 = b(j,:);
 a4 = [a1;a2];
 b4 = [b1;b2];
 x = a4
\b4;
 pt = [pt x];
 end
end
pt = [pt x]
ptt = pt'
allpt=[ptt;corpt]
points=unique(allpt,'rows')clear all
clc
format short
c = [3 5 0 0 0]
A = [-1 -3 1 0 -3;
 -1 -1 0 1 -2;]

bv = [3,4]
cost = zeros(1, size(A,2))
cost(1:5) = c
zjcj = cost(bv)*A - cost
zcj = [zjcj;A]
soln = A(:,end)
run = true
while(run == true)
 if(any(soln<0))
 negIND = find(soln<0)
 [leaving_var, pivot_row] = min(soln(negIND))
 ratio = []
 for i = 1:size(A,2)-1
 if A(pivot_row,i) <0
 ratio(i) = abs(zjcj(i)/A(pivot_row,i))
 else
 ratio(i) = inf;
 end
```

```
 end
 [entering_var, pivot_col] = min(ratio)
 pvt_key = A(pivot_row, pivot_col)
 bv(pivot_row) = pivot_col;
 A(pivot_row,:) = A(pivot_row,:)/pvt_key;
 for i = 1 : size(A,1)
 if i ~= pivot_row
 A(i,:) = A(i,:) - (A(i,pivot_col).*A(pivot_row,:));
 end
 end
 zjcj = zjcj - (zjcj(pivot_col).*A(pivot_row,:))
 zc = zjcj(1:end-1)
 soln = A(:,end)
 else
 run = false
 fprintf("Current BFS is Optimal")
 zcj = [zjcj;A]
 optimum_simplex_table = array2table(zcj)
P a g e | 27
 optimum_simplex_table.Properties.VariableNames(1:size(zcj,2)) = {'x1', 'x2',
's1', 's2', 'soln'}
 optimal_solution = zjcj(end)
 solns = [bv' A(:,end)]
 end
end
PT = constraint(points)
p=unique(PT,'rows')
for i=1:size(PT,1)
 fx(i,:)=sum(PT(i,:).*c)
end
P a g e | 4
vert_fns=[PT fx];
[fxval,indfx] = max(fx)
optval=vert_fns(indfx,:)
optimalbfs=array2table(optval)
optimalbfs.Properties.VariableNames(1:3) = {'x1', 'x2','z'}
end
values = [P fn]
%phase 8 : to find optimal sol
[Optval Optposition] = max(fn)
Optval = values(Optposition,:)
OPTIMAL_BFS = array2table(Optval);
OPTIMAL_BFS.Properties.VariableNames(1:size(Optval,2))={'x1', 'x2', 'z' }
```

Question 2:

```
clear all
clc
format short
%taking input
%the z function sholud be max always
c = [12 10 0 0 0 0]
%the info matrix with identity matrix and sol col included
A = [-5 -1 1 0 0 -10;
     -6 -5 0 1 0 -30;
     -1 -4 0 0 1 -8]
%index of starting basic variables
noofvar = 2
%bv = [3,4,5]
```

```
bv = noofvar+1:1:size(A,2)-1
cost = zeros(1, size(A,2));
cost(1:6) = c

zjcj = cost(bv)*A - cost
zcj = [zjcj;A]

%taking end wala col of A matrix
soln = A(:,end)

run = true
while(run == true)

    %because feasibilty disturb hoti toh sol col dekhenge
    %most negative choose krna
    if(any(soln<0))
        %finding the index of all the negative values in sol col
    negIND = find(soln<0)

    %finding leaving variable
    [leaving_var, pivot_row] = min(soln(negIND))

    ratio = []
    for i = 1:size(A,2)-1
        %z row mei se negative enteries ki ratio leni hai
        if A(pivot_row,i) <0
            ratio(i) = abs(zjcj(i)/A(pivot_row,i))
        else
            ratio(i) = inf;
        end
    end
    %row ki jgha col lena yaha
    [entering_var, pivot_col] = min(ratio)

    %updation of pivot key
    pvt_key = A(pivot_row, pivot_col)

    %updating basic variable
    bv(pivot_row) = pivot_col;

    %updation of pivot row
    A(pivot_row,:) = A(pivot_row,:)/pvt_key;

    %updation of other rows
    for i = 1 : size(A,1)
        % ab isme pivot row nhi leni
        if i ~= pivot_row
              A(i,:) = A(i,:) - (A(i,pivot_col).*A(pivot_row,:));
        end
    end
    %updation of z row
    zjcj = zjcj - (zjcj(pivot_col).*A(pivot_row,:))
    zc = zjcj(1:end-1)
    soln = A(:,end)
    else
        run = false
        fprintf("Current BFS is Optimal")

        zcj = [zjcj;A]
```

```
        dual_simpl_table = array2table(zcj);
        dual_simpl_table.Properties.VariableNames(1:size(zcj,2)) = {'x1', 'x2',
's1', 's2', 's3', 'soln'}
        optimal_solution = zjcj(end)
    end
end
```

```
    x1    x2       s1          s2       s3     soln
    __    __    ________    ________    __    ______

    0     0            0          -2    0         60
    0     1      0.31579    -0.26316    0     4.7368
    1     0     -0.26316    0.052632    0     1.0526
    0     0            1          -1    1         12


optimal_solution =

    60
```

QUESTION 3:

```
clc
clear all
format short
Variables={'x1','x2','s1','s2','sol'};
Cost=[-3 2 0 0 0];
info=[-1 -1 ;-1 -2 ];
s = eye(size(info,1))
b=[-1; -3];
A = [info s b]
BV = [];
for j=1:size(s,2)
 for i=1:size(A,2)
 if A(:,i)==s(:,j)
 BV = [BV i];
 end
 end
end
fprintf('Basic Variables (BV)=')
disp(Variables(BV));
ZjCj = Cost(BV)*A-Cost;
ZCj = [ZjCj;A];
Simptable = array2table(ZCj);
Simptable.Properties.VariableNames(1:size(ZCj,2))=Variables
RUN = true;
while RUN
sol = A(:,end);
if any(sol<0)
 fprintf("Current BFS is not feasible \n");
%Finding the leaving variable
 [LV, pivot_row] = min(sol);
 fprintf("Leaving row =%d\n",pivot_row);
%finding entering variable
 Row = A(pivot_row,1:end-1);
 ZJ = ZjCj(:,1:end-1);
```

```
for i=1:size(Row,2)
 if Row(i)<0
 ratio(i)= abs(ZJ(i)./Row(i));
 else
 ratio(i)=inf;
 end
end
 [minVal, pvt_col]=min(ratio);
 fprintf("Entering Variable = %d\n",pvt_col);
%Updation
 BV(pivot_row)=pvt_col;
 fprintf('Basic Variables (BV) =')
 disp(Variables(BV));
 pvt_key = A(pivot_row,pvt_col);
 A(pivot_row,:)=A(pivot_row,:)./pvt_key;
for i=1:size(A,1)
 if i~=pivot_row
 A(i,:)=A(i,:)-A(i,pvt_col).*A(pivot_row,:);
 end
end
 ZjCj = Cost(BV)*A-Cost
else
 RUN = false
 fprintf("Current BFS is feasible");
 ZCj = [ZjCj;A];
 SimpTable= array2table(ZCj);
 SimpTable.Properties.VariableNames(1:size(ZCj,2)) =Variables
end
end
solution=ZCj(1,end);
fprintf('\noptimal solution reached');
fprintf('\n\n optimal solution is %d',-1*solution)
```

x1	x2	s1	s2	sol
4	0	0	-1	3
-0.5	0	1	-0.5	0.5
0.5	1	0	-0.5	1.5

```
optimal solution reached

 optimal solution is -3>>
```

ASSIGNMENT 7

```
QUESTION 1:
%every balanced problem has feasible sol
%basic m+n-1
format short
clear all
clc

%obtain the intital bfs
cost = [2 10 4 5 ;
        6 12 8 11 ;
        3 9 5 7];
%supply
A = [ 12 25 20];
%demad
B = [ 25 10 15 5];

%check if balanced or not
if sum(A) == sum(B)
    fprintf('given transportation problem is balanced\n');
else
    fprintf('given transportation problem is not balanced\n');
    if sum(A) < sum(B)
        %ek row add krenge
        cost(end + 1,:) = zeros(1,size(cost,2));
        A(end+1) = sum(B) - sum(A);
    elseif sum (B) < sum(A)
        %column add krenge
        cost(:,end + 1) = zeros(size(cost,1),1);
        B(end+1) = sum(A) - sum(B);
    end
end

Icost = cost;
%initial allocation
X = zeros(size(cost));
%finding no of rows and cols
[m , n] = size(cost);
Bfs = m+n-1;

%finding the cell with min cost
for i = 1: size(cost , 1)
    for j = 1:size(cost,2)
hh = min(cost(:));
[rowind , colind] = find(hh==cost);

%to give allocations

x11 = min(A(rowind) , B(colind));
%find max allocation
[val , ind] = max(x11);
%identify the row and col position
ii = rowind(ind);
jj = colind(ind);

y11 = min(A(ii) , B(jj));
%assign allocation
```

```
X(ii , jj) = y11;
%reducing the values
A(ii) = A(ii) - y11;
B(jj) = B(jj) -y11;

cost(ii, jj) = Inf;
    end
end
%print inital bfs
fprintf('Intial bfs = \n')
ib = array2table(X);
disp(ib);

%check for degenerate
totalbfs = length(nonzeros(X));
if totalbfs == Bfs
    fprintf('intial bfs is non degenerate\n');
else
    fprintf('degenerate');
end
%computing the cost
initialcost = sum(sum(Icost.*X));
fprintf('intial bfs cost %d\n' , initialcost);
```

```
Intial bfs =
    X1     X2     X3     X4     X5
    __     __     __     __     __

    10      0      0     0      2
     0     10     10     5      0
    15      0      5     0      0

intial bfs is non degenerate
intial bfs cost 345
```

QUESTION 2:

```
%every balanced problem has feasible sol
%basic m+n-1
format short
clear all
clc

%obtain the intital bfs
cost = [3 11 4 14 15;
        6 16 18 2 28 ;
        10 13 15 19 17;
        7 12 5 8 9];
%supply
A = [ 15 25 10 15];
%demad
B = [20 10 15 15 5];

%check if balanced or not
```

```
if sum(A) == sum(B)
    fprintf('given transportation problem is balanced\n');
else
    fprintf('given transportation problem is not balanced\n');
    if sum(A) < sum(B)
        %ek row add krenge
        cost(end + 1,:) = zeros(1,size(cost,2));
        A(end+1) = sum(B) - sum(A);
    elseif sum (B) < sum(A)
        %column add krenge
        cost(:,end + 1) = zeros(size(cost,1),1);
        B(end+1) = sum(A) - sum(B);
    end
end

Icost = cost;
%initial allocation
X = zeros(size(cost));
%finding no of rows and cols
[m , n] = size(cost);
Bfs = m+n-1;

%finding the cell with min cost
for i = 1: size(cost , 1)
    for j = 1:size(cost,2)
hh = min(cost(:));
[rowind , colind] = find(hh==cost);

%to give allocations

x11 = min(A(rowind) , B(colind));
%find max allocation
[val , ind] = max(x11);
%identify the row and col position
ii = rowind(ind);
jj = colind(ind);

y11 = min(A(ii) , B(jj));
%assign allocation
X(ii , jj) = y11;
%reducing the values
A(ii) = A(ii) - y11;
B(jj) = B(jj) -y11;

cost(ii, jj) = Inf;
    end
end
%print inital bfs
fprintf('Intial bfs = \n')
ib = array2table(X);
disp(ib);

%check for degenerate
totalbfs = length(nonzeros(X));
if totalbfs == Bfs
    fprintf('intial bfs is non degenerate\n');
else
    fprintf('degenerate');
end
```

```
%computing the cost
initialcost = sum(sum(Icost.*X));
fprintf('intial bfs cost %d\n' , initialcost);
```

```
Intial bfs =
    X1    X2    X3    X4    X5
    __    __    __    __    __

    15     0     0     0    0
     5     0     0    15    5
     0    10     0     0    0
     0     0    15     0    0

degenerateintial bfs cost 450
```