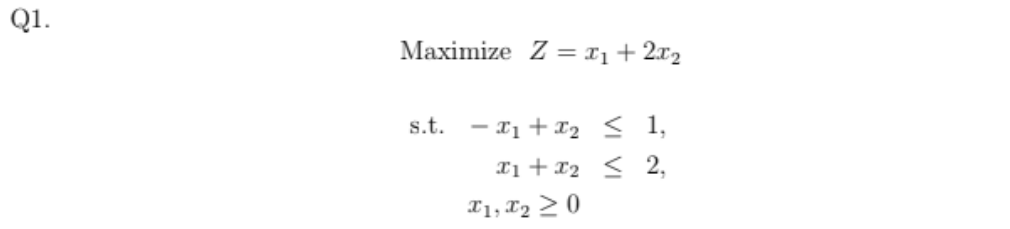
**EXPERIMENT 5**



**Code:**

clc

clear all

Noofvariables=2;

variables={'x1','x2','s1','s2','sol'};

c=[1 2]; % cost of objective func

Abar=[-1 1;1 1];% const coeff

B=[1;2]; %RHS of constraints

s=eye(size(Abar,1));

A=[Abar s B];

Cost=zeros(1,size(A,2));

Cost(1:Noofvariables)=c;

% Contraints BV

BV=Noofvariables+1:1:size(A,2)-1;

% To calculate Zj-Cj

ZjCj=Cost(BV)\*A-Cost;

% For printing 1st simplex table

ZCj=[ZjCj;A];

simplextable=array2table(ZCj);

simplextable.Properties.VariableNames(1:size(ZCj,2))=variables;

% Start simplex Algorithm

Run=true;

while Run

if any(ZjCj<0) % to check if any negative value there

fprintf('The current BFS is not optimal\n')

fprintf('Next iteration required \n')

disp('Old basic variable (BV)=')

disp(BV)

% For finding entering variable

Zc=ZjCj(1:end-1);

[Ent\_col pvt\_col]=min(Zc);

fprintf('The most negative value in Zj-Cj row is %d and coresponding to column %d \n',Ent\_col,pvt\_col)

fprintf('Entering variable is %d \n',pvt\_col)

%For finding the leaving variable

sol=A(:,end);

column=A(:,pvt\_col);

if all(column<=0)

error('The LPP has unbounded solution \n since all enteries are <=0 in %d \n',pvt\_col)

else

for i=1:size(column,1)

if column(i)>0

ratio(i)=sol(i)./column(i)

else

ratio(i)=inf

end

end

% To finding minimmum ratio

[minratio pvt\_row]=min(ratio);

fprintf('The minimum ratio corresponding to pivot row %d \n ',pvt\_row)

fprintf('leaving variable is %d \n ',BV(pvt\_row))

BV(pvt\_row)=pvt\_col;

disp('New basic variable(BV)==')

disp(BV)

pvt\_key=A(pvt\_row,pvt\_col)

% To update table for next iteration

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

ZjCj=ZjCj-ZjCj(pvt\_col).\*A(pvt\_row,:);

end

end

else

Run= false;

ZCj=[ZjCj;A]

FinalTable=array2table(ZCj);

FinalTable.Properties.VariableNames(1:size(ZCj,2))=variables

FinalTable.Properties.RowNames(1:size(ZCj,1))={'Zj-Cj','x1','x2'}

BFS=zeros(1,size(A,2));

BFS(BV)=A(:,end)

BFS(end)=sum(BFS.\*Cost);

currentBFS=array2table(BFS);

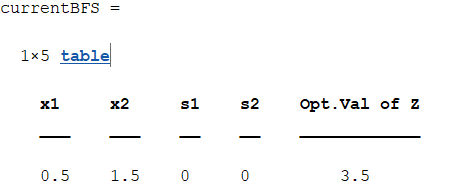
currentBFS.Properties.VariableNames(1:size(currentBFS,2))={'x1','x2','s1','s2','Opt.Val of Z'}

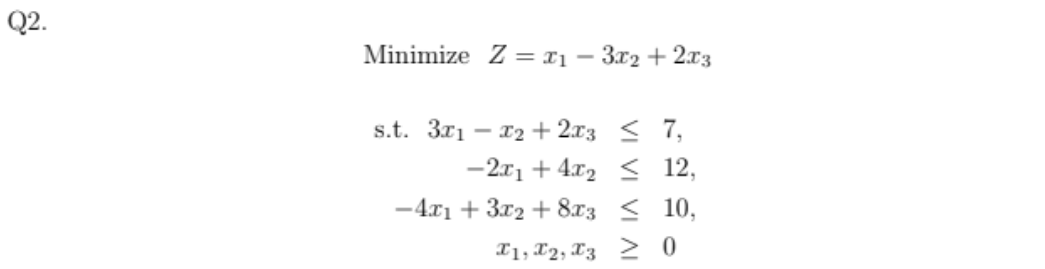
disp('Optimal sol is reached')

end

end

**Final Solution:**

****

****

**Code:**

clc

clear all

Noofvariables=3;

variables={'x1','x2','x3','s1','s2','s3','sol'};

c=[-1 3 -2]; % cost of objective func

Abar=[3 -1 2;-2 4 0;-4 3 8];% const coeff

B=[7;12;10]; %RHS of constraints

s=eye(size(Abar,1));

A=[Abar s B];

Cost=zeros(1,size(A,2));

Cost(1:Noofvariables)=c;

% Contraints BV

BV=Noofvariables+1:1:size(A,2)-1;

% To calculate Zj-Cj

ZjCj=Cost(BV)\*A-Cost;

% For printing 1st simplex table

ZCj=[ZjCj;A];

simplextable=array2table(ZCj);

simplextable.Properties.VariableNames(1:size(ZCj,2))=variables;

% Start simplex Algorithm

Run=true;

while Run

if any(ZjCj<0) % to check if any negative value there

fprintf('The current BFS is not optimal\n')

fprintf('Next iteration required \n')

disp('Old basic variable (BV)=')

disp(BV)

% For finding entering variable

Zc=ZjCj(1:end-1);

[Ent\_col pvt\_col]=min(Zc);

fprintf('The most negative value in Zj-Cj row is %d and coresponding to column %d \n',Ent\_col,pvt\_col)

fprintf('Entering variable is %d \n',pvt\_col)

%For finding the leaving variable

sol=A(:,end);

column=A(:,pvt\_col);

if all(column<=0)

error('The LPP has unbounded solution \n since all enteries are <=0 in %d \n',pvt\_col)

else

for i=1:size(column,1)

if column(i)>0

ratio(i)=sol(i)./column(i)

else

ratio(i)=inf

end

end

% To finding minimmum ratio

[minratio pvt\_row]=min(ratio);

fprintf('The minimum ratio corresponding to pivot row %d \n ',pvt\_row)

fprintf('leaving variable is %d \n ',BV(pvt\_row))

BV(pvt\_row)=pvt\_col;

disp('New basic variable(BV)==')

disp(BV)

pvt\_key=A(pvt\_row,pvt\_col)

% To update table for next iteration

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

ZjCj=ZjCj-ZjCj(pvt\_col).\*A(pvt\_row,:);

end

end

else

Run= false;

ZCj=[ZjCj;A]

FinalTable=array2table(ZCj);

FinalTable.Properties.VariableNames(1:size(ZCj,2))=variables

FinalTable.Properties.RowNames(1:size(ZCj,1))={'Zj-Cj','x1','s2','x3'}

BFS=zeros(1,size(A,2));

BFS(BV)=A(:,end)

BFS(end)=0-sum(BFS.\*Cost);

currentBFS=array2table(BFS);

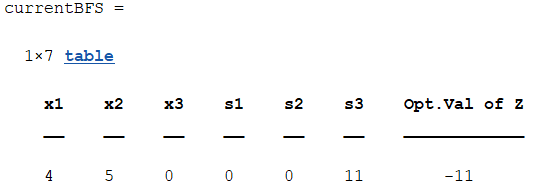
currentBFS.Properties.VariableNames(1:size(currentBFS,2))={'x1','x2','x3','s1','s2','s3','Opt.Val of Z'}

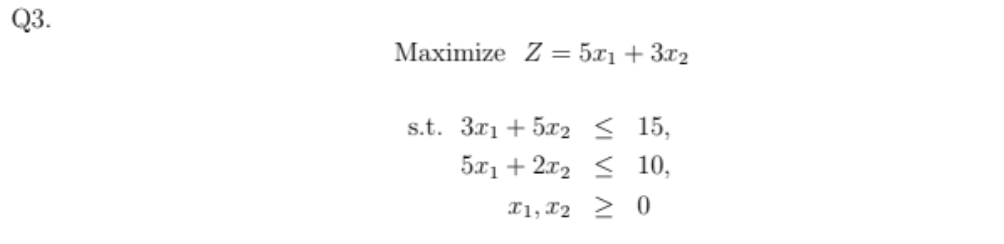
disp('Optimal sol is reached')

end

end

**Final Solution:**





**Code:**

clc

clear all

Noofvariables=2;

variables={'x1','x2','s1','s2','sol'};

c=[5 3]; % cost of objective func

Abar=[3 5;5 2];% const coeff

B=[15;10]; %RHS of constraints

s=eye(size(Abar,1));

A=[Abar s B];

Cost=zeros(1,size(A,2));

Cost(1:Noofvariables)=c;

% Contraints BV

BV=Noofvariables+1:1:size(A,2)-1;

% To calculate Zj-Cj

ZjCj=Cost(BV)\*A-Cost;

% For printing 1st simplex table

ZCj=[ZjCj;A];

simplextable=array2table(ZCj);

simplextable.Properties.VariableNames(1:size(ZCj,2))=variables;

% Start simplex Algorithm

Run=true;

while Run

if any(ZjCj<0) % to check if any negative value there

fprintf('The current BFS is not optimal\n')

fprintf('Next iteration required \n')

disp('Old basic variable (BV)=')

disp(BV)

% For finding entering variable

Zc=ZjCj(1:end-1);

[Ent\_col pvt\_col]=min(Zc);

fprintf('The most negative value in Zj-Cj row is %d and coresponding to column %d \n',Ent\_col,pvt\_col)

fprintf('Entering variable is %d \n',pvt\_col)

%For finding the leaving variable

sol=A(:,end);

column=A(:,pvt\_col);

if all(column<=0)

error('The LPP has unbounded solution \n since all enteries are <=0 in %d \n',pvt\_col)

else

for i=1:size(column,1)

if column(i)>0

ratio(i)=sol(i)./column(i)

else

ratio(i)=inf

end

end

% To finding minimmum ratio

[minratio pvt\_row]=min(ratio);

fprintf('The minimum ratio corresponding to pivot row %d \n ',pvt\_row)

fprintf('leaving variable is %d \n ',BV(pvt\_row))

BV(pvt\_row)=pvt\_col;

disp('New basic variable(BV)==')

disp(BV)

pvt\_key=A(pvt\_row,pvt\_col)

% To update table for next iteration

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

ZjCj=ZjCj-ZjCj(pvt\_col).\*A(pvt\_row,:);

end

end

else

Run= false;

ZCj=[ZjCj;A]

FinalTable=array2table(ZCj);

FinalTable.Properties.VariableNames(1:size(ZCj,2))=variables

FinalTable.Properties.RowNames(1:size(ZCj,1))={'Zj-Cj','x1','x2'}

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','Opt.Val of Z'}

disp('Optimal sol is reached')

end

end

**Final Solution:**

