### **BFS**

```
%Find all Basic feasible solutions for the LPP
%Max z=3x1+4X2+0X3+0X4
%1x1+1x2<=450, or 1x1+1x2+1x3+0x4=450
%2x1+1x2<=600 or 2x1+1x2+0x3+1x4=600
% x1>=0, x2>=0
clc
clear all
format short
A=[1 1 1 0; 2 1 0 1] % write in standard form
C=[3 4 0 0]
b=[450; 600]
n=size(A,2) % column in A matrix or no. of variables
m=size(A,1) % rows in A matrix or constraints
if(n>m)
nCm=nchoosek(n,m) %nchoosek command for n choose k (or binomial coeff)
pair=nchoosek(1:n,m) % all possible cases (x1&x2;x1&x3;and so on)
sol=[];
 for i=1:nCm
  y=zeros(n,1)% n-no of variables, initialize with all var(s)=0
  x=A(:,pair(i,:))\b %pair(i,:)ith row , including all columns
  if all(x \ge 0 \& x = \inf \& x = -\inf)
  y(pair(i,:))=x
```

```
sol=[sol, y]
 end
 end
else
  error('nCm does not exists')
end
Z=C*sol
[Zmax, Zindex]=max(Z)
bfs=sol(:,Zindex)
optimal_value=[bfs' Zmax]
optimal_bfs=array2table(optimal_value)
optimal_bfs.Properties.VariableNames(1:size(optimal_bfs,2))={'x_1','x_2','x_3','x_4','Z'}
                                         GRAPHICAL
clc
clear all
%% Problem:Solve the LPP with Graphical Method
       % Maximize Z=2x1 + x2,
% Subject to x1 + 2x2 \le 10,
          % x1 + x2 \le 6,
          % x1 - x2 \le 2,
          % x1 - 2x2 \le 1,
          % x1, x2 \geq 0
%% Phase 1: Insert the coefficient matrix and right hand side matrix
A=[1 2; 1 1; 1 -1;1 -2;1 0;0 1];
B=[10; 6; 2;1;0;0];
```

```
%% phase 2: Plotting the graph
P=max(B);
x1=0:1:max(B)
x21=(B(1)-A(1,1).*x1)./A(1,2);
x22=(B(2)-A(2,1).*x1)./A(2,2);
x23=(B(3)-A(3,1).*x1)./A(3,2);
x24=(B(4)-A(4,1).*x1)./A(4,2);
x21=max(0,x21);
x22=max(0,x22)
x23=max(0,x23)
x24=max(0,x24)
E=[x1(:,[A(1,1) A(1,2)])]
%plot(x1,x21,'r',x1,x22,'b',x1,x23,'g',x1,x24,'m');
% title('x1 vs x2');
% xlabel('value of x1');
% ylabel('value of x2');
% grid on
% hold on
%% phase 3: find corner point with axes, that is line intercept, or
%finding line intersections with axes
position_x1=find(x1==0) %points with x1 axis (index or position)
position_x21=find(x21==0) %points with x2 axis
Line1=[x1(:,[position_x1 position_x21]); x21(:,[position_x1 position_x21])]';
```

```
position_x22=find(x22==0) %points with x2 axis
Line2=[x1(:,[position_x1 position_x22]); x22(:,[position_x1 position_x22])]';
position_x23=find(x23==0) %points with x2 axis
Line3=[x1(:,[position_x1 position_x23]); x23(:,[position_x1 position_x23])]';
position_x24=find(x24==0) %points with x2 axis
Line4=[x1(:,[position_x1 position_x24]); x24(:,[position_x1 position_x24])]';
intersection_pts_axes=unique([Line1;Line2;Line3;Line4],'rows')
%% Phase 4: finding intersection of lines with each other
 pt=[0;0]
for i=1:size(A,1)
  A1=A(i,:);% first constraint for i=1
  B1=B(i,:);
  for j=i+1:size(A,1)
  A2=A(j,:);% second constraint for i+1=j=2
  B2=B(j,:);
  A4=[A1; A2];
  B4=[B1; B2];
  X=A4\B4 % inverse of matrix
  pt=[pt X]
end
end
 ptt=pt'
```

%%

```
% %% Phase5: Write all corner points
cor_pts=[intersection_pts_axes;ptt]
P=unique(cor_pts,'rows')
size(P)
%%
% %% Phase 6: Feasible region points
 b1=P(:,1); % write first column of matrix
 b2=P(:,2);
% % %write 1st Constraint % all constraints are of <= sign
cons1=round(b1+(2.*b2)-10);
s1=find(cons1>0);
P(s1,:)=[];
% % % write 2nd Constraint % all constraints are of <= sign
 b1=P(:,1);
 b2=P(:,2);
 cons2=round((b1+b2)-6);
 s2=find(cons2>0);
P(s2,:)=[];
% % %write 3rd Constraint % all are of <= sign
 b1=P(:,1);
 b2=P(:,2);
 cons3=round((b1-b2)-2);
 s3=find(cons3>0);
```

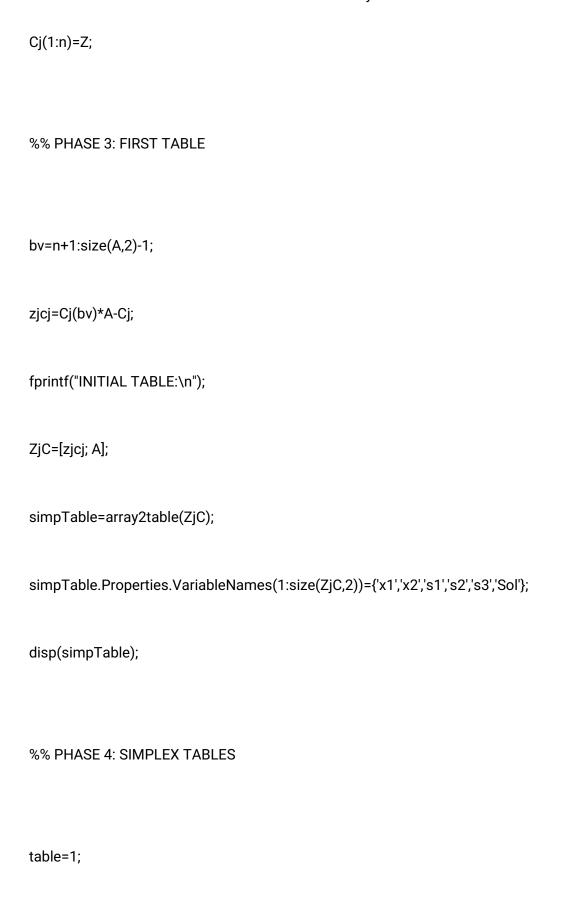
```
P(s3,:)=[];
% % %write 4th Constraint % all are of <= sign
b1=P(:,1);
 b2=P(:,2);
 cons4=round((b1-(2.*b2))-1);
 s4=find(cons4>0);
P(s4,:)=[];
 % % %write 5th Constraint % all are of <= sign
 b1=P(:,1);
 b2=P(:,2);
 cons5=round(-b1);
 s5=find(cons5>0);
P(s5,:)=[];
 % % %write 6th Constraint % all are of <= sign
b1=P(:,1);
 b2=P(:,2);
 cons6=round(-b2);
s6=find(cons6>0);
 P(s6,:)=[];
f_points=P
%
%%
% %% Phase 7:Objective function value
c=[2,1];
```

for i=1:size(P,1) fn(i,:)=(sum(P(i,:).\*c)); optim=max(fn) end **SIMPLEX** clc clear format rat % Max Z=3x1+2x2 % 2x1+x2<=18 % 2x1+3x2<=42

%% PHASE 1: INPUT PARAMETER

% 3x1+x2<=24

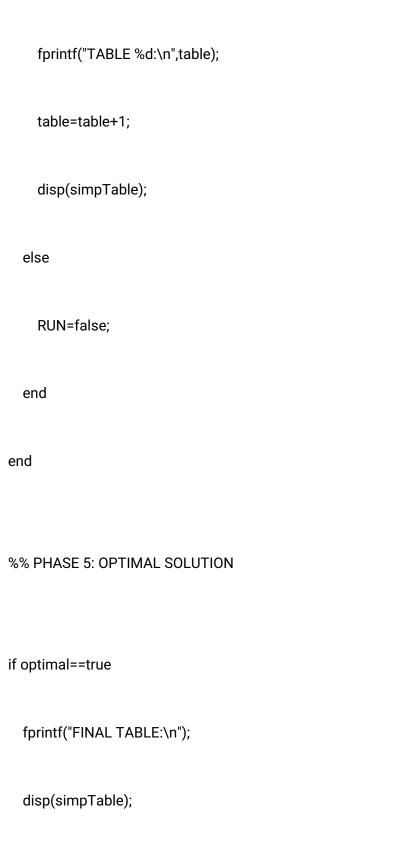
Z=[3 2];
A=[2 1; 2 3; 3 1];
B=[18; 42; 24];
%% PHASE 2: COMPLETE MATRIX AND COST MATRIX
s=eye(size(A,1));
m=size(A,1);
n=size(A,2);
col=size(A,2);
A=[A s B];
Cj=zeros(1,size(A,2));
% Cost Matrix



```
optimal=true;
RUN=true;
zc=zjcj(1:size(A,2)-1);
while RUN
  if any(zc<0)
    zc=zjcj(1:size(A,2)-1);
    [minvalzjcj, minindzjcj]=min(zc);
    pivot_col_ind=minindzjcj;
    pivot_col=A(:,pivot_col_ind);
    if all(pivot_col<=0)
      print('LPP is unbounded');
      optimal=false;
```

```
break;
else
  for i=1:size(pivot_col,1)
    if pivot_col(i)>0
      ratio(i)=B(i)./pivot_col(i);
    else
      ratio(i)=inf;
    end
  end
  [min_ratio, ratio_ind]=min(ratio);
  pivot_row_ind=ratio_ind;
  bv(ratio_ind)=pivot_col_ind;
```

```
pivot_value=A(pivot_row_ind,pivot_col_ind);
  A(pivot_row_ind,:)=A(pivot_row_ind,:)./pivot_value;
  for i=1:size(A,1)
    if i~=pivot_row_ind
       A(i,:)=A(i,:)-(pivot\_col(i)*A(pivot\_row\_ind,:));
    end
  end
  zjcj=zjcj-(zjcj(pivot_col_ind)*A(pivot_row_ind,:));
end
ZjC=[zjcj; A];
B(1:m)=(A(:,size(A,2)))';
simpTable=array2table(ZjC);
simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','s3','Sol'};
```



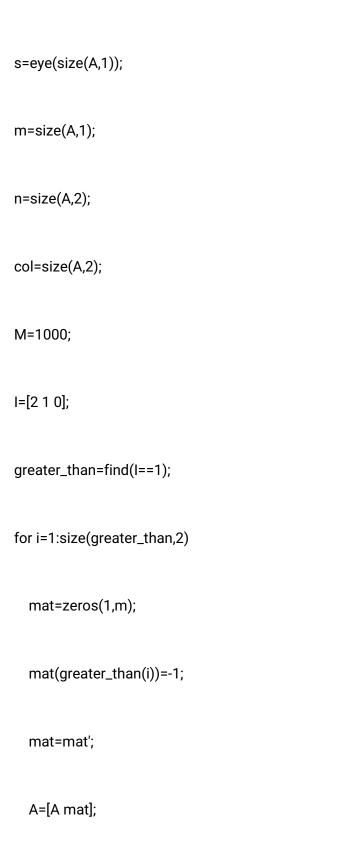
```
fprintf("OPTIMAL SOLUTION:\n");
  for i=1:col
    index=find(bv==i);
    if(index==0)
      fprintf("x%d = 0\n",i);
    else
      fprintf("x%d = %.3f\n",i,A(index,size(A,2)));
    end
  end
  fprintf("Max Z = %f",ZjC(1,size(ZjC,2)));
end
```

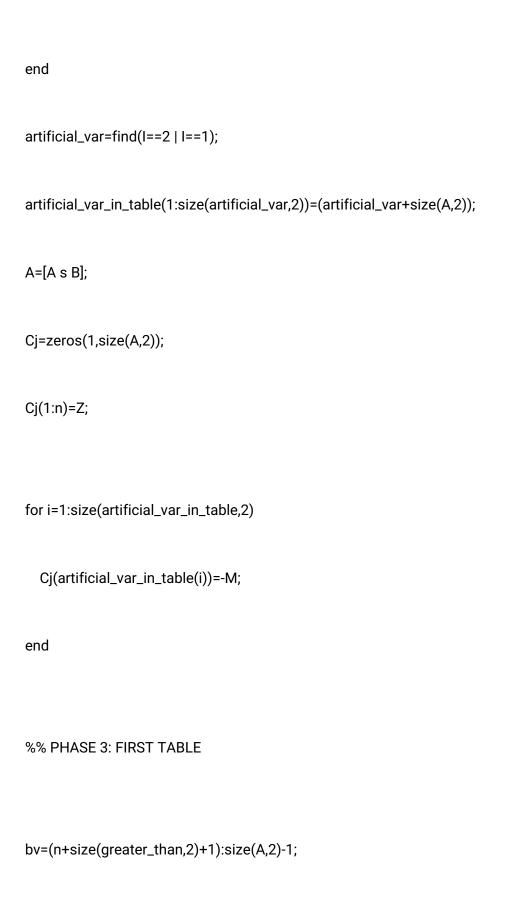
**BIG M** 

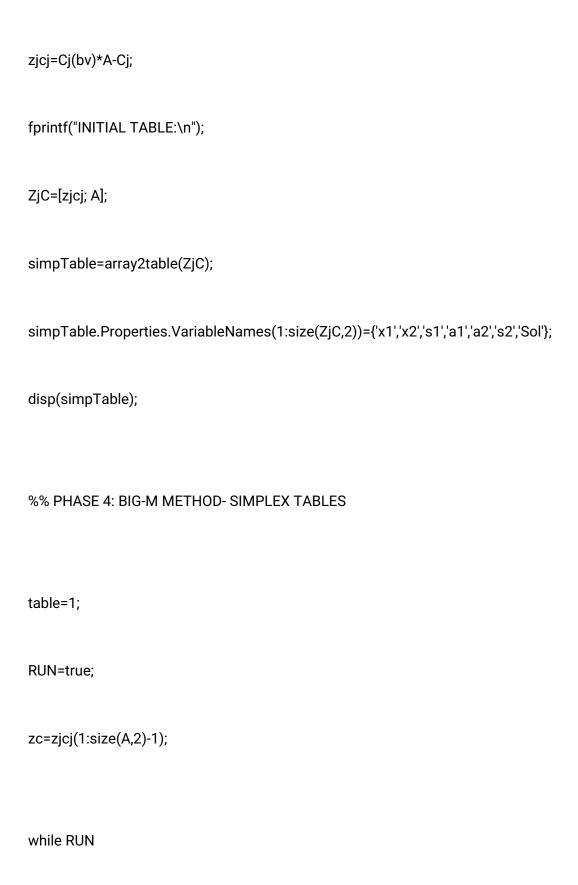
clc

clear
format rat
% MIN Z=2x1+x2
% 3x1+x2=3;
% 4x1+3x2>=6
% x1+2x2<=3
%% PHASE 1: INPUT PARAMETER
A=[3 1; 4 3; 1 2];
B=[3; 6; 3];
Z=[-2 -1];

%% PHASE 2: STANDARD FORM

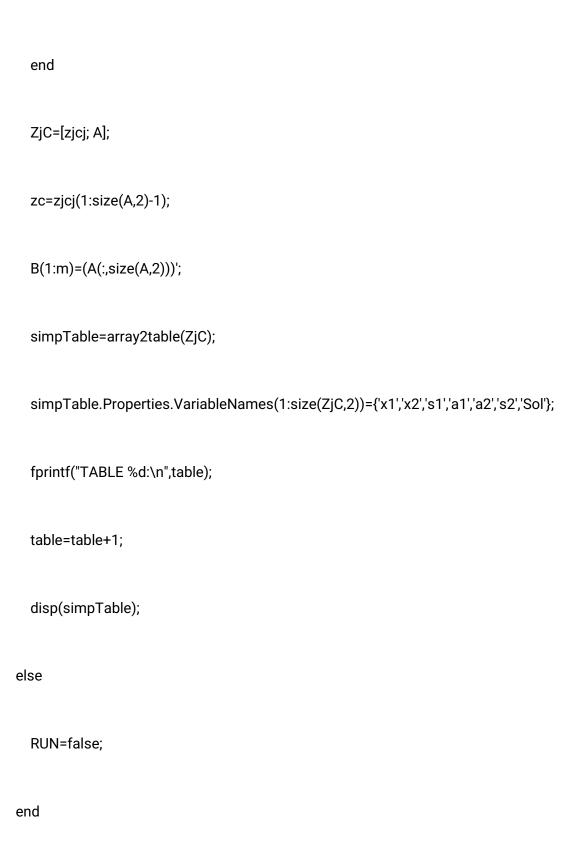


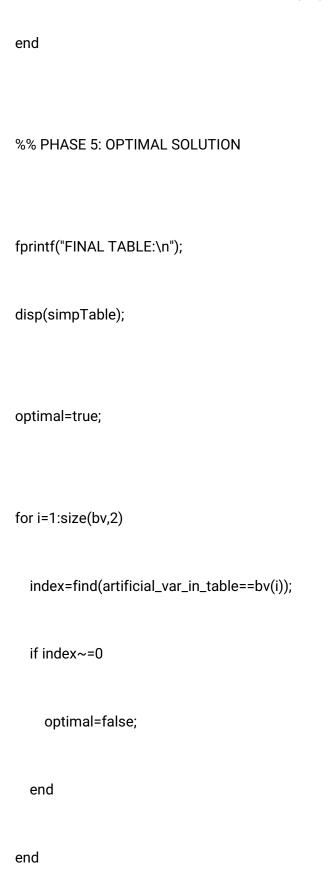




```
if any(zc<0)
  [minvalzjcj, minindzjcj]=min(zc);
  pivot_col_ind=minindzjcj;
  pivot_col=A(:,pivot_col_ind);
  if all(pivot_col<=0)
    print('LPP is unbounded');
  else
    for i=1:size(pivot_col,1)
       if pivot_col(i)>0
         ratio(i)=B(i)./pivot_col(i);
       else
         ratio(i)=inf;
```

end end [min\_ratio, ratio\_ind]=min(ratio); pivot\_row\_ind=ratio\_ind; bv(ratio\_ind)=pivot\_col\_ind; pivot\_value=A(pivot\_row\_ind,pivot\_col\_ind); A(pivot\_row\_ind,:)=A(pivot\_row\_ind,:)./pivot\_value; for i=1:size(A,1) if i~=pivot\_row\_ind  $A(i,:)=A(i,:)-(pivot\_col(i)*A(pivot\_row\_ind,:));$ end end zjcj=zjcj-(zjcj(pivot\_col\_ind)\*A(pivot\_row\_ind,:));





```
if optimal==false
  disp('INFEASIBLE SOLUTION');
else
  fprintf("OPTIMAL SOLUTION:\n");
  for i=1:col
    index=find(bv==i);
    if(index>0)
      fprintf("x%d = %.3f\n",i,A(index,size(A,2)));
    else
      fprintf("x%d = 0\n",i);
    end
  end
```

end

# TWO PHASE

clear all

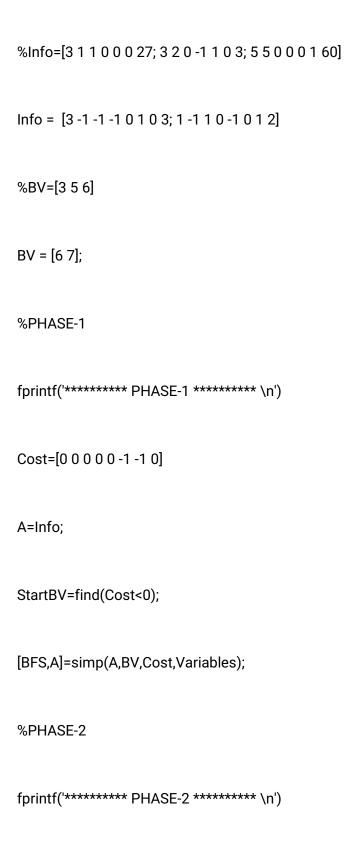
clc

Variables={'x\_1','x\_2','x\_3','s\_1','s\_2','A\_1','A\_2','sol'};

OVariables={'x\_1','x\_2','x\_3','s\_1','s\_2','sol'};

OrigC=[-7.5 3 0 0 0 -1 -1 0]

%OrigC=[-4 -5 0 0 -1 -1 0]



A(:,StartBV)=[]; %Removing Artificial var by giving them empty value

OrigC(:,StartBV)=[]; %Removing Artificial var cost by giving them empty value

[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);

Optimal BFS. Properties. Variable Names (1: size (Optimal BFS, 2)) = OVariables