```matlab
%graphics
clc;
f1 = @(x1, x2) x1 + x2 - 40;      %constraint 1
f2 = @(x1, x2) 3*x1 + x2 - 30;    %constraint 2
f3 = @(x1, x2) 4*x1 + 3*x2 - 60;  %constraint 3
z = @(x1, x2) 20*x1 + 10*x2;
A = [1 1; 3 1; 4 3; 1 0; 0 1];
B = [40; 30; 60; 0; 0];

% plot
% n = 3;
% x = 0:max(B);
% for i = 1:n
%     y = (B(i) - A(i, 1) * x) / A(i, 2);
%     y_positive = max(zeros(1, length(y)), y);
%     plot(x, y_positive)
%     hold on
% end

%calculating intersection points
pt = [];
for i = 1:size(A)
    for j = i + 1:size(A)
        AA = [A(i, :); A(j, :)];
        BB = [B(i, :); B(j, :)];
        X = AA\BB;
        if (X >= 0)     %removing negative values
            pt = [pt X];
        end
    end
end
pt

%finding points which satisfies constraints (feasible points)
final_pt = [];
for i = 1:length(pt)
    if (f1(pt(1, i), pt(2, i)) <= 0 && f2(pt(1, i), pt(2, i)) <= 0 && f3(pt(1, i),
pt(2, i)) <= 0)
        final_pt = [final_pt pt(:, i)];
    end
end
final_pt

%finding maximum Z from final_pt
z_max = z(final_pt(1, 1), final_pt(2, 1));
z_pt = final_pt(:, 1);
for i = 2:length(final_pt)
    temp = z(final_pt(1, i), final_pt(2, i));
    if (z_max < temp)
        z_max = temp;
        z_pt = final_pt(:, i);
    end
end
z_max
z_pt
```

```
%plotting graph
n = 3;
x = 0:max(B);
for i = 1:n
    y = (B(i) - A(i, 1) * x) / A(i, 2);
    y_positive = max(zeros(1, length(y)), y);
    plot(x, y_positive)
    hold on
end

%marking feasible points
plot(final_pt(1, :), final_pt(2, :), '.', 'markersize', 20, 'color', 'green')

%giving different colour to z_pt (feasible point with max value)
plot(z_pt(1, :), z_pt(2, :), '.', 'markersize', 20, 'color', 'red')




%Bfs
% z = -x1 + 2x3 -x3
% x1 <= 4
% x2 <= 4
% -x1 + x2 <=6
% -x1 + 2x3 <=4
C=[-1 2 -1 0 0 0 0 ];
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];
n = size(A, 2);
m = size(A, 1);
if (n >= m)
    nv = nchoosek(n, m);
    t = nchoosek(1:n, m);
    sol = [];
     for i = 1: nv
          y = zeros(n, 1);
          if round(det(A(:,t(i,:))),3) ~= 0
              x = A(:,t(i,:)) \ b;
            if all(x >= 0 & x ~=inf & x ~= -inf )
                 y(t(i, :)) = x;
                 sol = [sol y];

%                   if any(x==0)
%                       disp('Degenerate BFS');
%                   else
%                       disp('Non-degenerate BFS');
%                   end
%             else
%                 disp('Infeaseible solution');
             end
%           else
%               disp('Basis matrix inverse does not exist');
           end
      end
```

```
% else
%     error('No solution exists for this system because number of constraints are
greater');
end
z = C*sol;
[zmax, zind] = max(z);
BFS = sol (:,zind);
optval=[BFS', zmax];
optimal_BFS = array2table(optval);
disp(optimal_BFS)
%  % optimal_BFS.Properties.VariableNames(1:size(optimal_BFS,
2))={'x1','x2','s1','s2','value_of_z'};




%simplex
clc
clear
format rat


% Max Z=3x1+2x2
% 2x1+x2<=18
% 2x1+3x2<=42
% 3x1+x2<=24

%% PHASE 1: INPUT PARAMETER

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
Cj(1:n)=Z;

%% PHASE 3: FIRST TABLE

bv=n+1:size(A,2)-1;
zjcj=Cj(bv)*A-Cj;
fprintf("INITIAL TABLE:\n");
ZjC=[zjcj; A];
simpTable=array2table(ZjC);
simpTable.Properties.VariableNames(1:size(ZjC,2))={'x1','x2','s1','s2','s3','Sol'};
disp(simpTable);
```

```
%% PHASE 4: SIMPLEX TABLES

table=1;
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("TABLE %d:\n",table);
        table=table+1;
        disp(simpTable);
    else
        RUN=false;
    end
end

%% PHASE 5: OPTIMAL SOLUTION

if optimal==true
    fprintf("FINAL TABLE:\n");
    disp(simpTable);
```

```
    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 all
%%%%%%%%%%%%% Converting to standard form and adding artificials %%%%%
%%%%%%%%%%%%%
% Inputs: Coefficient matrix A, Sign matrix for constraints, number of
% constraints, rhs of constraints, cost
A = [3,2;1,4;1,1];
m = 3;
nv = 2;
sign = [-1,-1,1];
b = [3;4;5]
c = [5,8]
S = eye(m) % Initializing to identity adds slack variable to all constraints
% Check sign of each constraints and add variables (slack, surplus, artificial)
cnew = c
nart = 0
index_bv = []
for i = 1:m
dummy = zeros(m,1)
if sign(i) < 0
S(i,i) = -S(i,i);
nart = nart + 1
dummy(i) = 1;
S = [S dummy]; % add a column when '>=' constarint
cnew(nv + i) = 0
cnew(nv + m + nart) = -1000000
else if sign(i) == 0
S(i,i) = 1;
cnew(nv + i) = -1000000
else if sign(i) > 0
cnew(nv + i) = 0
end
end
end
end
```

```
Anew = [A S]
n = length(Anew)
nart = 0
for i = 1:m
index_bv(i) = nv + i
if sign(i) < 0
nart = nart + 1
index_bv(i) = nv + m + nart
end
end
cb = cnew(index_bv)
%%%%%%%%%%%%%Simplex Algorithim%%%%%%%%%%%%%%%%%%%%%%%%
%first convert the problem to standard form and then input the following
% values
n=length(Anew);% Number of variables
c= cnew;
A= Anew;
index = index_bv
% RHS vector b and number of constraints m is used from before.
%index is used from BFS calculation done before.
for s=1:100 % number of iterations
B=[];
for j=1:m
B=[ B A(:,index(j))]; % computing basis matrix
end
tolerance=10^-7;
if abs(det(B))<tolerance
disp('change basis matrix');
end
cb=c(index); % cost of basic variables
Xb=inv(B)*b; % basic solution
if nnz(Xb)< m % checking degeneracy
disp('degenerate bfs')
Xb
index
end
z=cb*Xb; %computing objective function value
Y=inv(B)*A; % computing column vectors of all variables
NE=cb*Y-c; % computing net evaluations
%optimality check
if NE>=0 % for max problem, sign will be changed for min problem
disp('optimality declared');
Xb
index
z
nzeros=length(NE)-nnz(NE); % checking alternate optimal solution
if nzeros>m
disp('alternate optimal solution');
NE(index)=10;
for j=1:n
if NE(j)==0
EV=j;
break
end
end
```

```
for j=1:m
if Y(j,EV)>0
ratio(j)=Xb(j)/Y(j,EV);
else
ratio(j)= 10^8;
end
end
[k,LV]=min(ratio);
index(LV)=EV;
B=[];
for j=1:m
B=[ B A(:,index(j))];
end
cb=c(index);
Xb=inv(B)*b
z=cb*Xb
index
Y=inv(B)*A;
NE=cb*Y-c;
else
disp(' no alt optimal solution');
end
break
else % if optimality not declared
%choose entering variable
[a,EV]= min(NE);
%choose leaving variable
for j=1:m
if Y(j,EV)>0
ratio(j)=Xb(j)/Y(j,EV); % selecting only positive pivots
else
ratio(j)= 10^8;
end
end
if min(ratio)==10^8 % this would happen if no positive pivots are there in column of
entering variable
disp('Lpp is unbounded')
else
[k,LV]=min(ratio);
index(LV)=EV;% replacing leaving variable with entering variable
end
end
end


%2 phase
clear all
clc
Variables= {'x_1','x_2','s_1','s_2','A_2','A_3','sol'};
OVariables={'x_1','x_2','s_1','s_2','sol'};
OrigC = [-4 -5 0 0 -1 -1 0];

Info = [3 1 1 0 0 0 27; 3 2 0 -1 1 0 3; 5 5 0 0 0 1 60];

BV=[3 5 6];
```

```
%PHASE-1
fprintf('********** PHASE-1 ********** \n')
Cost=[0 0 0 0 -1 -1 0]
A=Info;
StartBV=find(Cost<0); %define the artificial variables

% compute zj-cj
ZjCj=Cost(BV)*A-Cost;

RUN= true;
while RUN
ZC=ZjCj(:,1:end-1);
if any(ZC<0)
fprintf(' The current BFS is not optimal\n')
[ent_col,pvt_col]=min(ZC);
fprintf('Entering Col =%d \n' , pvt_col);
sol=A(:,end);
Column=A(:,pvt_col);
if Column<=0
error('LPP is unbounded');
else
for i=1:size(A,1)
    if Column(i)>0
ratio(i)=sol(i)./Column(i);
    else
ratio(i)=inf;
    end
end
[MinRatio,pvt_row]=min(ratio);
fprintf('leaving Row=%d \n', pvt_row);
end
BV(pvt_row)=pvt_col;
pvt_key=A(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,:)
ZCj=[ZjCj;A]
TABLE=array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2))=Variables
BFS(BV)=A(:,end)
else RUN=false;
fprintf(' Current BFS is Optimal \n');
fprintf('Phase 1 End \n')
BFS=BV;
end
end

%PHASE-2
fprintf('********** PHASE-2 ********** \n')
```

```matlab
A(:,StartBV)=[]; %Removing Artificial var by giving them empty value
OrigC(:,StartBV)=[]; %Removing Artificial var cost by giving them empty value
ZjCj=OrigC(BV)*A-OrigC;
RUN= true;
while RUN
ZC=ZjCj(:,1:end-1);
if any(ZC<0)
fprintf(' The current BFS is not optimal\n')
[ent_col,pvt_col]=min(ZC);
fprintf('Entering Col =%d \n' , pvt_col);
sol=A(:,end);
Column=A(:,pvt_col);
if Column<=0
error('LPP is unbounded');
else
for i=1:size(A,1)
    if Column(i)>0
ratio(i)=sol(i)./Column(i);
    else
ratio(i)=inf;
    end
end
[MinRatio,pvt_row]=min(ratio);
fprintf('leaving Row=%d \n', pvt_row);
end
BV(pvt_row)=pvt_col;
pvt_key=A(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,:)
ZCj=[ZjCj;A]
TABLE=array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2))=OVariables
BFS(BV)=A(:,end)
else RUN=false;
fprintf(' Current BFS is Optimal \n');

fprintf('Phase End \n')
BFS=BV;
end
end

FINAL_BFS=zeros(1,size(A,2)); FINAL_BFS(BFS)=A(:,end);
FINAL_BFS(end)=sum(FINAL_BFS.*OrigC); OptimalBFS= array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2))= OVariables
```