clc

clear all

C = [19 17 23 21 25];

A = [60 25 45 20 50;10 15 45 50 40;30 60 10 30 10];

b = [40;35;25];

% m - number of constraints

% n - number of variables

[m n] = size(A);

% Checking whether the constraint matrix has redundant rows or not

if rank(A)==m

fprintf('The constraint matrix has full row rank \n')

% If it has redundant rows, the code below will remove them

else

fprintf('The constraint matrix doesnt have full row rank and there are redundant rows \n')

for i=1:m

for j=i+1:m

if rank([A(i,:);A(j,:)]) < m-1

% Combining the rows before and after the redundant row

% This is same as removing the redundant row

A = A([1:j-1,j+1:end],:);

end

end

end

% The constraint matrix is updated after removing redundant rows

fprintf('The updated constrained matrix after removing redundant rows is \n')

disp(A)

end

found = 0;

% The below code checks if the LP has an initial basis

% It checks all possible m x m matrices

for i=1:n

for j=1:n

for k=1:n

R = [A(:,i) A(:,j) A(:,k)];

% Condition to check for initial basis

% diagonal elements must be 1 and determinant must be 1

if (isequal(diag(R),[1 1 1]') && det(R)==1)

fprintf('initial basis found')

found = 1;

basis =[ i j k ];

end

end

end

end

% Code for Phase-1

% This is executed only if initial basis not found

if found == 0

fprintf('initial basis not found \n\n')

% Adding artificial variables

A = [A eye(m)];

% Cost function for phase 1

c1 = [ zeros(1,n) -max(eye(m))];

% Initial basis

basis = [n+1 n+2 n+3];

fprintf('The current variables in the basis are \n')

display(basis)

% creating the tableau format

tab1 = [c1 0];

for i=1:m

tab1 = [ tab1;A(i,:) b(i,:)];

end

fprintf('initial tableau for phase 1 iteration 1 is')

display(tab1)

% making the z values of the basic variables 'zero' initially

for j=2:m+1

tab1(1,:)= tab1(1,:)+tab1(j,:);

end

optimal = 0;

iter = 0;

% Loop for running iterations until the optimal solution is reached

while(optimal==0)

iter = iter+1;

% Choosing the maximum z value which will enter the basis

% Here bland's rule automatically applied

% Since 'max' will return the value with least index incase of tie

[M,I] = max(tab1(1,1:m+n));

% ratio test for choosing the minimum positive value

% choosing which variable is leaving the basis

ratiot = tab1(2:m+1,m+n+1)./tab1(2:m+1,I);

% Here bland's rule automatically applied

% Since 'min' will return the value with least index incase of tie

[H,G] = min(ratiot(ratiot>0));

basis(G)=I;

fprintf('The current variables in the basis are \n')

display(basis)

% Row transformation of the pivot variable to make it '1'

tab1(G+1,:)=tab1(G+1,:)./tab1(G+1,I);

% Row transformations to make other rows '0' using the pivot row

for i=1:m+1

% Transforming rows other than pivot row

% Since pivot row is already transformed

if(i~=G+1)

tab1(i,:)=tab1(i,:)-(tab1(G+1,:)\*tab1(i,I));

end

end

fprintf('The tableau for iteration %d',iter+1)

display(tab1)

% Checking if all 'z' are negative to obtain an optimal solution

if isempty(tab1(tab1(1,1:m+n)>0.000000001))==1

optimal=1;

end

end

% If at optimal solution, RHS not equal to zero

% It becomes infeasible

if tab1(1,m+n+1)~=0

fprintf('The LP is infeasible')

% Otherwise, we can proceed to Phase-2

else

fprintf('The LP is feasible and proceed to phase 2')

end

end

% If we have an initial basis we will directly come to Phase-2

% If we have a feasible solution after Phase-1, we come to Phase-2

% Beginning of Phase-2

fprintf('The current variables in the basis are \n')

display(basis)

fprintf('initial tableau for phase 2 iteration 1 is')

% Creating the initial tableau for Phase-2

d = tab1(2:end,m+n+1);

A = tab1(2:end,1:n);

tab2 = [-C 0];

for i=1:m

tab2 = [ tab2;A(i,:) d(i,:)];

end

% Making the z-values of initial basis as 0

for j=1:m

tab2(1,:)=tab2(1,:)-tab2(j+1,:)\*tab2(1,basis(j));

end

display(tab2)

optimal = 0;

iter = 0;

% Checking if the solution is optimal before proceeding

% This is done by checking if all z values are positive

if isempty(tab2(tab2(1,1:n)>0.000000001))==1

optimal=1;

end

% The loop goes on until an optimal solution is reached

while(optimal==0)

iter = iter+1;

% Choosing the maximum z value which will enter the basis

% Here bland's rule automatically applied

% Since 'max' will return the value with least index incase of tie

[M,I] = max(tab2(1,1:n));

% Ratio test for choosing the minimum positive value

ratiot = tab2(2:m+1,n+1)./tab2(2:m+1,I);

% If there is no positive value in ratio test, the LP is unbounded

if isempty(ratiot(ratiot>0))==1

fprintf('The LP is unbounded \n')

break;

end

% Here bland's rule automatically applied

% Since 'min' will return the value with least index incase of tie

[H,G] = min(ratiot(ratiot>0));

basis(G)=I;

fprintf('The current variables in the basis are \n')

display(basis)

% Row transformation of the pivot variable to make it '1'

tab2(G+1,:)=tab2(G+1,:)./tab2(G+1,I);

% Row transformations to make other rows '0' using the pivot row

for i=1:m+1

% Transforming rows other than pivot row

% Since pivot row is already transformed

if(i~=G+1)

tab2(i,:)=tab2(i,:)-(tab2(G+1,:)\*tab2(i,I));

end

end

fprintf('The tableau for iteration %d',iter+1)

display(tab2)

% Checking if all 'z' are negative to obtain an optimal solution

if isempty(tab2(tab2(1,1:n)>0.000000001))==1

optimal=1;

end

end

% If we reach an optimal solution, printing the optimal obj function value

if optimal==1

fprintf('The optimal objective function value is \n')

display(tab2(1,n+1))

fprintf('The optimal solution \n')

% To print the optimal solution

for i=1:n

u=0;

fprintf('x%d = ',i)

for j=1:m

if i==basis(j)

fprintf('%d \n',tab2(j+1,n+1))

u=1;

end

end

if u==0

fprintf('%d \n',u)

end

end

% Printing the extreme direction incase of unboundedness

else

fprintf('The extreme direction for the LP which is unbounded is \n')

for k=1:n

v=0;

for j=1:m

if k==basis(j)

dir(k,1)= -tab2(j+1,I);

v=1;

end

end

if v==0

if k==I

dir(k,1)=1;

else

dir(k,1)=0;

end

end

end

end