# Contents

## Pseudocode for simplex solver

(Implement Bland's rule while finding pivot index).

1. Input A, b, and c of the minimization LP in standard form –
   $Min$
   $C'x$
   $St$:
   $Ax = b$
   $x, b \geq 0$

2. Introduce artificial variables and start phase I.
3. In the optimal tableau, if cost > 0, the problem is infeasible. Display and exit.
4. In the optimal tableau, if cost =0, the problem is feasible. Display and continue.
5. If artificial variables are there in the final basis, then there are redundant rows.
6. Remove the redundant rows by driving the artificial variables out of the basis.

   Driving artificial variables out of the basis:

   i. If all the columns of $B^{-1}A$ corresponding to the row of the artificial variable in basis is zero, simply delete that row.
   ii. Else, pivot at the first non-zero column.
7. After removing the redundant rows, go to phase II using the final basis and tableau from phase I.
8. Compute the reduced costs in phase II and start phase II.
9. While finding pivots, keep checking for unbounded condition. If unbounded, display and exit.
10. When an optimal tableau is found, display the optimum cost and the solution.

# Model 17

## LP formulation

Problem:

| | Fertilizer 1 | Fertilizer 2 | Fertilizer 3 | Fertilizer 4 | Fertilizer 5 | | | |
|---|---|---|---|---|---|---|---|---|
| Shop 1 | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | | <= | 350 |
| Shop 2 | $X_6$ | $X_7$ | $X_8$ | $X_9$ | $X_{10}$ | | <= | 225 |
| Shop 3 | $X_{11}$ | $X_{12}$ | $X_{13}$ | $X_{14}$ | $X_{15}$ | | <= | 195 |
| Shop 4 | $X_{16}$ | $X_{17}$ | $X_{18}$ | $X_{19}$ | $X_{20}$ | | <= | 275 |
| | | | | | | | | |
| | >= | >= | >= | >= | >= | | | |
| | 185 | 50 | 50 | 200 | 185 | | | |

$x_1, x_2, \ldots\ldots x_{20}$ represent tons of fertilizer type from each shop.

Slack variables:

$$x_{21}, x_{22}, \ldots \ldots x_{29} \geq 0$$

Minimize

$$45x_1 + 13.9x_2 + 29.9x_3 + 31.9x_4 + 9.9x_5 + 42.5\,x_6 + 17.8\,x_7 + 31x_8 + 35x_9 + 12.3x_{10} + 47.5x_{11} + 19.9x_{12} + 24x_{13} + 32.5x_{14} + 12.4x_{15} + 41.3x_{16} + 12.5x_{17} + 31.2x_{18} + 29.8x_{19} + 11x_{20}$$

Subject to:

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_{21} = 350$$

$$x_6 + x_7 + x_8 + x_9 + x_{10} + x_{22} = 225$$

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{23} = 195$$

$$x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{24} = 275$$

$$x_1 + x_6 + x_{11} + x_{16} - x_{25} = 185$$

$$x_2 + x_7 + x_{12} + x_{17} - x_{26} = 50$$

$$x_3 + x_8 + x_{13} + x_{18} - x_{27} = 50$$

$$x_4 + x_9 + x_{14} + x_{19} - x_{28} = 200$$

$x_5 + x_{10} + x_{15} + x_{20} - x_{29} = 185$

$x_1, x_2 \dots \dots \dots \dots .., x_{29} \geq 0$

## Simplex Solver

Solution:

lin_solve(A,b,c) is my LP simplex solver.

```
%model_17.m



%problem initialization - A, b, and c, in standard form
%Ax =b
%problem should be minimization form

A = zeros(9,29);
A(1,1:5) = 1;
A(1,21) =1;

A(2,6:10)=1;
A(2,22) =1;

A(3,11:15)=1;
A(3,23) =1;

A(4,15:20)=1;
A(4,24)=1;
k =25;

for i=5:9
    for j=i-4:5:i+11
        A(i,j)=1;
    end
    A(i,k)=-1;
    k = k+1;
end

b =[350;225;195;275;185;50;50;200;185];

c
=[45;13.9;29.9;31.9;9.9;42.5;17.8;31.0;35.0;12.3;47.5;19.9;24.0;32.5;12.4;41.3;12.5;31.2;29.8;11.
0];
c(21:29)=0;


%lin_solve is my LP solver. It returns the optimum cost and the optimum x value.
```

```
[o_c,x] = lin_solve(A,b,c)

%o_c is the optimum cost and x is the solution.


%In lin_solve, each part of the model is handeled by a function.


function [o_c,x] = lin_solve(A,b,c)
m = size(A,1);
n=size(A,2);

[T,B] = initialtab(A,b); %initialtab() adds the artificial variables and initializes
                         %the initial tableau.
                         % T is the tableau and B is an array which has the indices of
                         %the basic variables.

[T,B,f] =ph_one(T,B);   %phase one of the two phase approach. It also checks for
                        %feasibility and returns f=1 when feasible

if f==1
[T,B] = red_remove(T,B,m,n); % checks and removes the redundant constraints if any.
                             %It reurns the tableau without the artificial variables

[T,B] = phtwoinitialize(c,T,B); %initializes the reduced costs of the tableau for the phase 2.

[T,B,u] = ph_two(T,B); %phase two of the 2-phase approach. It checks for unboundedness
                       %If unbounded, it displays a message and returns u=1.
if u==1
    o_c=[];x=[];return;
end
[o_c,x] = disp_sol(T,B); % finally displays the optimum cost and the optimum x.
end
end






function [T,B] = initialtab(A,b) %adds the artificial variables and initializes
                                 %the initial tableau.
m = size(A,1);
n=size(A,2);
i = eye(m); %artificial variables
A = [A,i];
c_b = ones(m,1);
c = [zeros(n,1);c_b];

top = (c_b)'*A - c';
f = [top;A];
cost = (c_b)'*b;
r =[cost;b];
T =[f,r];
```

```
B = n+1 : n+m;
end




function [T,B,f]= ph_one(T,B)
o =0;
while o==0
    o=chkopt(T);
    if (o==0)
        index = findpivot(T,B);
        [T,B] = pivot(T,index,B);

    else
        if(T(1,end)<1e-6)  %similar to T(1,end)==0, but only till 10^-6.
            disp("LP is Feasible"); %check for feasibility, cost=0 at the end of phase 1.
            f=1;
        else
            disp("LP is not Feasible");
            f=0;
        end


    end
end
end






 function [T,B] = red_remove(T,B,m,n) % checks for redundant constraints and removes them.
art = n+1:m+n;
k = ismember(B,art);                   % if artificial variables in basis, then redundant
                                       %constraints exist.
if k==0
    disp("No redundant Constraints");
    disp("A has full row rank");
else
    disp("Redundant Constraints found");

    l = find(k==1);                   % removing redundant constraints
    for dum=1:length(l)
        p = ismember(B,art);
        i = find(p==1,1);

        if T(i+1,1:n)==0           %if all elememts of (B^-1 * A) in the row corresponding
                                   %to artificial variable
            T(i+1,:)=[];           %in the basis =0, then eliminate the row
            B(i)=[];
        else
            Y=T(i+1,:);
            j = find(Y~=0,1);      %else, pivot at the first non-zero element,
```

```
            index = [i+1,j];       %to drive the artificial variable out.
            [T,B] = pivot(T,index,B);
        end
    end
    disp("Redundant Constraints Removed"); %The redundant constraints are removed.
    disp(T);disp(B);
end

T(:,n+1:end-1)=[];              %removing the artificial variables.
 end




function [T,B] = phtwoinitialize(c,T,B) %phase 2 initialization - reduced costs
T(1,1:end-1) = -1*c';
T(1,end) = 0;
for j = B
    i =find(j==B(:));
    x = T(1,j)/T(i+1,j);
    T(1,:) = T(1,:)-x*T(i+1,:);
end
end




function [T,B,u] = ph_two(T,B) % phase 2
o = 0;
while o==0
    o=chkopt(T);              %chkopt() checks if the tableau is optimal and returns 1
    if o==0                   %when optimal.
        [index,u] = findpivot_ptwo(T,B); %findpivot_ptwo() returns the index of the pivot element
        if u==1                          %It also checks if the problem is unbounded,
            disp("Unbounded Problem");   %if unbounded,it returns u=1.
            return;
        end
        [T,B] = pivot(T,index,B);
    else
        disp("Optimal Solution found.") %when tableau is optimal

    end
end
end




function [o_c,x]= disp_sol(T,B) % displays the solution
o_c = T(1,end);
n = size(T,2)-1;
x = zeros(n,1);
for k = B
    i = find(k==B(:));
```

```
    x(k) = T(i+1,end);
end
disp("Optimal Cost:");  %optimal cost
disp(o_c);
disp("solution:");      %optimal x value
for i = 1:n
       fprintf('x%d = %d',i,x(i));    %prints x1, x2 format
    fprintf('\n');
end
end




function [T,B] = pivot(T,index,B) %does the pivot operation on the tableau.
i = index(1);
j = index(2);
m = size(T,1);
B(i-1) = j;  %because there is an extra reduced cost row in the tableau
a = T(i,j);
T(i,:) = T(i,:)/a;
for l = 1:m
    if l~=i
        x = T(l,j);
        T(l,:) = T(l,:) - x*T(i,:);
    end
end

end




%Finding pivots index using bland's rule. B is the xi's in the basis.
function [index] = findpivot(T,B) %finds the pivot element based on Bland's rule.
A = T(1,1:end-1);
j = find(A>0,1);

R = T(2:end,end)./T(2:end,j);
R(R<=0) = inf;
k = find(R==min(R(:)));
if length(k)>1
    i = find(B==min(B(k)));

else
    i=k;
end

index = [i+1,j];
end
```

```matlab
function [index,u] = findpivot_ptwo(T,B) %finds the pivot element based on Bland's rule.
                                            %also checks for unboundedness.
A = T(1,1:end-1);
j = find(A>0,1);

W=T(2:end,j);
if W<=0                      %checking for unboundedness. expression evaluates the whole vector.
    u =1;index=[1,1];return; % u, to check if unbounded.
end

R = T(2:end,end)./T(2:end,j);
R(R<=0) = inf;
k = find(R==min(R(:)));
if length(k)>1
    i = find(B==min(B(k)));

else
    i=k;
end
u =0;
index = [i+1,j];
end




function o = chkopt(T) %checks if the tableau is optimal or not.
c = T(1,1:end-1);
if all(c(:)<=0)
    o =1;              %returns 1 if optimal
else
    o =0;              %returns 0 if not optimal
end
end
```

```
LP is Feasible
No redundant Constraints
A has full row rank
Optimal Solution found.
Optimal Cost:
       17449

solution:
x1 = 0
x2 = 0
x3 = 0
x4 = 0
x5 = 185
x6 = 160
x7 = 0
x8 = 0
x9 = 0
x10 = 0
```

```
x11 = 0
x12 = 0
x13 = 50
x14 = 0
x15 = 0
x16 = 25
x17 = 50
x18 = 0
x19 = 200
x20 = 0
x21 = 165
x22 = 65
x23 = 145
x24 = 0
x25 = 0
x26 = 0
x27 = 0
x28 = 0
x29 = 0

o_c =

       17449


x =

     0
     0
     0
     0
   185
   160
     0
     0
     0
     0
     0
     0
    50
     0
     0
    25
    50
     0
   200
     0
   165
    65
   145
     0
     0
     0
     0
```

```
0
0
```

*[Published with MATLAB® R2017b]*

## Linprog Verification

```
%model_17_linprog.m

%problem initialization - A, b, and c, in standard form
%Ax =b

A = zeros(9,29);
A(1,1:5) = 1;
A(1,21) =1;

A(2,6:10)=1;
A(2,22) =1;

A(3,11:15)=1;
A(3,23) =1;

A(4,15:20)=1;
A(4,24)=1;
k =25;

for i=5:9
    for j=i-4:5:i+11
        A(i,j)=1;
    end
    A(i,k)=-1;
    k = k+1;
end

b =[350;225;195;275;185;50;50;200;185];

c
=[45;13.9;29.9;31.9;9.9;42.5;17.8;31.0;35.0;12.3;47.5;19.9;24.0;32.5;12.4;41.3;12.5;31.2;29.8;11.
0];
c(21:29)=0;
%problem initialization same as before

S=[]; %for A
r=[]; %for b
      %llinprog takes Aeq, beq separately from A and b.
lb=zeros(29,1);
```

```
%linprog command
[x,fval] = linprog(c,S,r,A,b,lb)
```

```
Optimal solution found.


x =

     0
     0
     0
     0
   185
   160
     0
     0
     0
     0
     0
     0
    50
     0
     0
    25
    50
     0
   200
     0
   165
    65
   145
     0
     0
     0
     0
     0
     0


fval =

       17449
```

*Published with MATLAB® R2017b*

## Interior Point Method Solver

inp_solve() is my interior point method solver.

Centered interior point - barrier method implemented.

Separate Primal and Dual steps implemented

Infeasible starting point implemented. Only condition x>0 and s>0.

```
%model_17_inp

%problem initialization Ax = b. Same as before

A = zeros(9,29);
A(1,1:5) = 1;
A(1,21) =1;

A(2,6:10)=1;
A(2,22) =1;

A(3,11:15)=1;
A(3,23) =1;

A(4,15:20)=1;
A(4,24)=1;
k =25;

for i=5:9
    for j=i-4:5:i+11
        A(i,j)=1;
    end
    A(i,k)=-1;
    k = k+1;
end

b =[350;225;195;275;185;50;50;200;185];

c
=[45;13.9;29.9;31.9;9.9;42.5;17.8;31.0;35.0;12.3;47.5;19.9;24.0;32.5;12.4;41.3;12.5;31.2;29.8;11.
0];
c(21:29)=0;
%problem intialization done


[o_c,x] = inp_solve(A,b,c)



%inp_solve() is my interior point method solver.
%Centered interior point - barrier method implemented.
%Separate Primal and Dual steps implemented
%Infeasible starting point implemented. Only condition x>0 and s>0.
```

```
function [o_c,x] = inp_solve(A,b,c)

m = size(A,1);
n = size(A,2);



 x = ones(n,1); %infeasible starting point x>0
 s = ones(n,1); %infeasible starting point s>0

 y = linsolve(A',(c-s));


%INP parameters intialization
alpha = 0.995;
beta = 0.1;
e = 1e-6;
check =1;
iter = 0;

% Start of INP method
while(check==1)

u = beta * (s'*x)/n;

v = -1 *(s./x);

d = diag(v);

mat = zeros(m+n);


mat(1:n,:) = [d,A'];
mat(n+1:end,1:n) = A; % mat is the (n+m) x (n+m) matrix



r = (c-(A'*y)) - u * (1./x);
k= b - A*x;

coeff = [r;k];


sol = linsolve(mat,coeff); %the main linear equations solving
                           %step in INP



d_x = sol(1:n);          %delta x
d_y = sol(n+1:end);      %delta y

d_s = (c-(A'*y)) - s - (A'*d_y); %delta s
```

```
l_x = find(d_x<0);
l_s = find(d_s<0);

r_x = -1*(x(l_x)./d_x(l_x));
theta_x = min(r_x);

r_s = -1*(s(l_s)./d_s(l_s));
phi_s = min(r_s);

theta = min([1,(alpha*theta_x)]);  %theta

phi = min([1,(alpha*phi_s)]);      %phi



x = x + theta * d_x;   %next iteration
y = y + phi * d_y;
s = s + phi * d_s;

ch = x .* s;          %ch used for checking condition



iter = iter +1;      %number of iteraions



if ch(:) < e         % checking condition.
                        %e initialized as 1e-6.
       check = 0;
end

end

disp("Optimal Solution Found");

x= round(x,4);      %solution


cost = c'*x;

o_c = cost;         %optimum cost

end
```

```
Optimal Solution Found

o_c =

       17449


x =
```

```
     0
     0
     0
     0
   185
   160
     0
     0
     0
     0
     0
     0
    50
     0
     0
    25
    50
     0
   200
     0
   165
    65
   145
     0
     0
     0
     0
     0
     0
```

*Published with MATLAB® R2017b*

## Conclusion

The minimum cost: 17449

Solution:

| | Fertilizer 1 | Fertilizer 2 | Fertilizer 3 | Fertilizer 4 | Fertilizer 5 | | | |
|---|---|---|---|---|---|---|---|---|
| **Shop 1** | 0 | 0 | 0 | 0 | **185** | | <= | 350 |
| **Shop 2** | **160** | 0 | 0 | 0 | 0 | | <= | 225 |
| **Shop 3** | 0 | 0 | **50** | 0 | 0 | | <= | 195 |
| **Shop 4** | **25** | **50** | 0 | **200** | 0 | | <= | 275 |
| | | | | | | | | |
| | >= | >= | >= | >= | >= | | | |
| | 185 | 50 | 50 | 200 | 185 | | | |

The same solution is obtained from my simplex method and interior point method solvers.

The LP is verified with *linprog,* and the solution is same in all the three cases.

## Model 18*

All the solvers are same as used in the previous problem.

Only the initialization of A, b, and c is different.

### LP Formulation

Coefficient table:

| | Soy | Corn | Oats | Cows | Hens | Overtime_w | Overtime_s |
|---|---|---|---|---|---|---|---|
| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
| **Funds** | 0 | 0 | 0 | 1200 | 9 | 0 | 0 |
| **Land** | 1 | 1 | 1 | 1.5 | 0 | 0 | 0 |
| **Barn** | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **House** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **Winter** | 20 | 35 | 10 | 100 | 0.6 | 1 | 0 |
| **Summer** | 50 | 75 | 40 | 50 | 0.3 | 0 | 1 |

Problem:

$x_1, x_2, \ldots . x_7$ amount of each item in respective units.

Slack variables:

$x_8, x_9, \ldots . x_{13} \geq 0$

Maximize

$500x_1 + 750x_2 + 350x_3 + 1000x_4 + 5x_5 + 5x_6 + 6x_7$

Or

Minimize

$$-500x_1 - 750x_2 - 350x_3 - 1000x_4 - 5x_5 - 5x_6 - 6x_7$$

Subject to:

$$1200x_4 + 9x_5 + x_8 = 40000$$

$$x_1 + x_2 + x_3 + 1.5x_4 + x_9 = 125$$

$$x_4 + x_{10} = 32$$

$$x_5 + x_{11} = 3000$$

$$20x_1 + 35x_2 + 10x_3 + 100x_4 + 0.6x_5 + x_6 + x_{12} = 3500$$

$$50x_1 + 75x_2 + 40x_3 + 50x_4 + 0.3x_5 + x_7 + x_{13} = 3500$$

$$x_1, x_2, \dots \dots x_{13} \geq 0$$

## Simplex Solver

```
%model_18.m



%problem initialization - A, b, and c, in standard form
%Ax =b
%problem should be minimization form

A=[0    0       0       1200    9       0       0;1     1       1.5     0       0       0;0
        0       0       1       0       0       0;0     0       0       0       1       0       0;20
        35      10      100     0.6     1       0;50    75      40      50      0.3     0       1];
c=[500  750     350     1000    5       5       6]';
c = -1*c; %converting to minimization form
b=[40000;125;32;3000;3500;4000];

z = eye(6);
c(8:13)=0;
A =[A,z];


%lin_solve is my LP solver. It returns the optimum cost and the optimum x value.

[o_c,x] = lin_solve(A,b,c)

%o_c is the optimum cost and x is the solution.


%In lin_solve, each part of the model is handeled by a function.


function [o_c,x] = lin_solve(A,b,c)
```

```
m = size(A,1);
n=size(A,2);

[T,B] = initialtab(A,b); %initialtab() adds the artificial variables and initializes
                         %the initial tableau.
                         % T is the tableau and B is an array which has the indices of
                         %the basic variables.

[T,B,f] =ph_one(T,B);   %phase one of the two phase approach. It also checks for
                        %feasibility and returns f=1 when feasible

if f==1
[T,B] = red_remove(T,B,m,n); % checks and removes the redundant constraints if any.
                             %It reurns the tableau without the artificial variables

[T,B] = phtwoinitialize(c,T,B); %initializes the reduced costs of the tableau for the phase 2.

[T,B,u] = ph_two(T,B); %phase two of the 2-phase approach. It checks for unboundedness
                       %If unbounded, it displays a message and returns u=1.
if u==1
    o_c=[];x=[];return;
end
[o_c,x] = disp_sol(T,B); % finally displays the optimum cost and the optimum x.
end
end





function [T,B] = initialtab(A,b) %adds the artificial variables and initializes
                                 %the initial tableau.
m = size(A,1);
n=size(A,2);
i = eye(m); %artificial variables
A = [A,i];
c_b = ones(m,1);
c = [zeros(n,1);c_b];

top = (c_b)'*A - c';
f = [top;A];
cost = (c_b)'*b;
r =[cost;b];
T =[f,r];
B = n+1 : n+m;
end




function [T,B,f]= ph_one(T,B)
o =0;
while o==0
```

```
    o=chkopt(T);
    if (o==0)
        index = findpivot(T,B);
        [T,B] = pivot(T,index,B);

    else
        if(T(1,end)<1e-6)  %similar to T(1,end)==0, but only till 10^-6.
            disp("LP is Feasible"); %check for feasibility, cost=0 at the end of phase 1.
            f=1;
        else
            disp("LP is not Feasible");
            f=0;
        end


    end
end
end





 function [T,B] = red_remove(T,B,m,n) % checks for redundant constraints and removes them.
art = n+1:m+n;
k = ismember(B,art);                  % if artificial variables in basis, then redundant
                                      %constraints exist.
if k==0
    disp("No redundant Constraints");
    disp("A has full row rank");
else
    disp("Redundant Constraints found");

    l = find(k==1);                  % removing redundant constraints
    for dum=1:length(l)
        p = ismember(B,art);
        i = find(p==1,1);

        if T(i+1,1:n)==0           %if all elememts of (B^-1 * A) in the row corresponding
                                   %to artificial variable
            T(i+1,:)=[];           %in the basis =0, then eliminate the row
            B(i)=[];
        else
            Y=T(i+1,:);
            j = find(Y~=0,1);      %else, pivot at the first non-zero element,
            index = [i+1,j];       %to drive the artificial variable out.
            [T,B] = pivot(T,index,B);
        end
    end
    disp("Redundant Constraints Removed"); %The redundant constraints are removed.
    disp(T);disp(B);
end

T(:,n+1:end-1)=[];              %removing the artificial variables.
```

```
 end



function [T,B] = phtwoinitialize(c,T,B) %phase 2 initialization - reduced costs
T(1,1:end-1) = -1*c';
T(1,end) = 0;
for j = B
    i =find(j==B(:));
    x = T(1,j)/T(i+1,j);
    T(1,:) = T(1,:)-x*T(i+1,:);
end
end



function [T,B,u] = ph_two(T,B) % phase 2
o = 0;
while o==0
    o=chkopt(T);              %chkopt() checks if the tableau is optimal and returns 1
    if o==0                   %when optimal.
        [index,u] = findpivot_ptwo(T,B); %findpivot_ptwo() returns the index of the pivot element
        if u==1                          %It also checks if the problem is unbounded,
            disp("Unbounded Problem");   %if unbounded,it returns u=1.
            return;
        end
        [T,B] = pivot(T,index,B);
    else
        disp("Optimal Solution found.") %when tableau is optimal

    end
end
end



function [o_c,x]= disp_sol(T,B) % displays the solution
o_c = T(1,end);
n = size(T,2)-1;
x = zeros(n,1);
for k = B
    i = find(k==B(:));
    x(k) = T(i+1,end);
end
disp("Optimal Cost:");  %optimal cost
disp(o_c);
disp("solution:");      %optimal x value
for i = 1:n
       fprintf('x%d = %d',i,x(i));    %prints x1, x2 format
    fprintf('\n');
end
```

```
end




function [T,B] = pivot(T,index,B) %does the pivot operation on the tableau.
i = index(1);
j = index(2);
m = size(T,1);
B(i-1) = j;  %because there is an extra reduced cost row in the tableau
a = T(i,j);
T(i,:) = T(i,:)/a;
for l = 1:m
    if l~=i
        x = T(l,j);
        T(l,:) = T(l,:) - x*T(i,:);
    end
end

end




%Finding pivots index using bland's rule. B is the xi's in the basis.
function [index] = findpivot(T,B) %finds the pivot element based on Bland's rule.
A = T(1,1:end-1);
j = find(A>0,1);

R = T(2:end,end)./T(2:end,j);
R(R<=0) = inf;
k = find(R==min(R(:)));
if length(k)>1
    i = find(B==min(B(k)));

else
    i=k;
end

index = [i+1,j];
end




function [index,u] = findpivot_ptwo(T,B) %finds the pivot element based on Bland's rule.
                                            %also checks for unboundedness.
A = T(1,1:end-1);
j = find(A>0,1);

W=T(2:end,j);
if W<=0                      %checking for unboundedness. expression evaluates the whole vector.
    u =1;index=[1,1];return; % u, to check if unbounded.
end
```

```
R = T(2:end,end)./T(2:end,j);
R(R<=0) = inf;
k = find(R==min(R(:)));
if length(k)>1
    i = find(B==min(B(k)));

else
    i=k;
end
u =0;
index = [i+1,j];
end




function o = chkopt(T) %checks if the tableau is optimal or not.
c = T(1,1:end-1);
if all(c(:)<=0)
    o =1;              %returns 1 if optimal
else
    o =0;              %returns 0 if not optimal
end
end
```

```
LP is Feasible
No redundant Constraints
A has full row rank
Optimal Solution found.
Optimal Cost:
       -51875

solution:
x1 = 5.625000e+01
x2 = 0
x3 = 0
x4 = 2.375000e+01
x5 = 0
x6 = 0
x7 = 0
x8 = 1.150000e+04
x9 = 3.312500e+01
x10 = 8.250000e+00
x11 = 3000
x12 = 0
x13 = 0

o_c =


       -51875
```

```
x =

        56.25
            0
            0
        23.75
            0
            0
            0
        11500
       33.125
         8.25
         3000
            0
            0
```

*Published with MATLAB® R2017b*

## Linprog Verification

```
%model_18_linprog.m

%problem initialization - A, b, and c, in standard form
%Ax =b

A=[0    0       0       1200    9       0       0;1     1       1       1.5     0       0       0;0
        0       0       1       0       0       0;0     0       0       0       1       0       0;20
        35      10      100     0.6     1       0;50    75      40      50      0.3     0       1];
c=[500  750     350     1000    5       5       6]';
c = -1*c; %converting to minimization form
b=[40000;125;32;3000;3500;4000];

z = eye(6);
c(8:13)=0;
A =[A,z];


S=[]; %for A
r=[]; %for b
      %llinprog takes Aeq, beq separately from A and b.
lb=zeros(13,1);

%linprog command
[x,fval] = linprog(c,S,r,A,b,lb)
```

```
Optimal solution found.


x =
```

```
        56.25
            0
            0
        23.75
            0
            0
            0
        11500
       33.125
         8.25
         3000
            0
            0


fval =

      -51875
```

*Published with MATLAB® R2017b*

## Interior Point Method Solver

inp_solve() is my interior point method solver.

Centered interior point - barrier method implemented.

Separate Primal and Dual steps implemented

Infeasible starting point implemented. Only condition x>0 and s>0.

```
%model_17_inp

%problem initialization Ax = b. Same as before

A=[0    0       0       1200    9       0       0;1     1       1       1.5     0       0       0;0
        0       0       1       0       0       0;0     0       0       0       1       0       0;20
        35      10      100     0.6     1       0;50    75      40      50      0.3     0       1];
c=[500  750     350     1000    5       5       6]';
c = -1*c;
b=[40000;125;32;3000;3500;4000];

z = eye(6);
c(8:13)=0;
A =[A,z];



[o_c,x] = inp_solve(A,b,c)
```

```
%inp_solve() is my interior point method solver.
%Centered interior point - barrier method implemented.
%Separate Primal and Dual steps implemented
%Infeasible starting point implemented. Only condition x>0 and s>0.


function [o_c,x] = inp_solve(A,b,c)

m = size(A,1);
n = size(A,2);



 x = ones(n,1); %infeasible starting point x>0
 s = ones(n,1); %infeasible starting point s>0

 y = linsolve(A',(c-s));


%INP parameters intialization
alpha = 0.995;
beta = 0.1;
e = 1e-6;
check =1;
iter = 0;

% Start of INP method
while(check==1)

u = beta * (s'*x)/n;

v = -1 *(s./x);

d = diag(v);

mat = zeros(m+n);


mat(1:n,:) = [d,A'];
mat(n+1:end,1:n) = A; % mat is the (n+m) x (n+m) matrix



r = (c-(A'*y)) - u * (1./x);
k= b - A*x;

coeff = [r;k];


sol = linsolve(mat,coeff); %the main linear equations solving
                           %step in INP
```

```
d_x = sol(1:n);          %delta x
d_y = sol(n+1:end);      %delta y

d_s = (c-(A'*y)) - s - (A'*d_y); %delta s


l_x = find(d_x<0);
l_s = find(d_s<0);

r_x = -1*(x(l_x)./d_x(l_x));
theta_x = min(r_x);

r_s = -1*(s(l_s)./d_s(l_s));
phi_s = min(r_s);

theta = min([1,(alpha*theta_x)]);  %theta

phi = min([1,(alpha*phi_s)]);      %phi



x = x + theta * d_x;   %next iteration
y = y + phi * d_y;
s = s + phi * d_s;

ch = x .* s;          %ch used for checking condition



iter = iter +1;      %number of iteraions



if ch(:) < e         % checking condition.
                        %e initialized as 1e-6.
       check = 0;
end

end

disp("Optimal Solution Found");

x= round(x,4);      %solution


cost = c'*x;

o_c = cost;         %optimum cost

end
```

```
Optimal Solution Found

o_c =

     -51875


x =

       56.25
           0
           0
       23.75
           0
           0
           0
       11500
      33.125
        8.25
        3000
           0
           0
```

[*Published with MATLAB® R2017b*](http://www.mathworks.com/products/matlab/)

## Conclusion

The optimum cost of the minimization problem is -51875.

The optimum cost of the original maximization problem is 51875.

The optimum solution:

| | Soy | Corn | Oats | Cows | Hens | Overtime_w | Overtime_s |
|---|---|---|---|---|---|---|---|
| | **56.25** | **0** | **0** | **23.75** | **0** | **0** | **0** |
| **Funds** | 0 | 0 | 0 | 1200 | 9 | 0 | 0 |
| **Land** | 1 | 1 | 1 | 1.5 | 0 | 0 | 0 |
| **Barn** | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **House** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **Winter** | 20 | 35 | 10 | 100 | 0.6 | 1 | 0 |
| **Summer** | 50 | 75 | 40 | 50 | 0.3 | 0 | 1 |

56.25 acres of Soy and 23.75 cows*.

The same solution is obtained from my simplex method and interior point method solvers.

The LP is verified with *linprog,* and the solution is same in all the three cases.

*Note: Integrality constraint is not implemented. Hence, decimal number of cows.