Search and Optimization 21/22

Assignment

LIVESTOCK FEEDING PROBLEM

Claire DELGOVE – S359263

Date: October 2021 **Department**: MSc AAI

Contents

1 Context

The diet problem is a complex case, not only dedicated to farmers feeding their cattle. The search for a minimal cost diet originates from World War II.
Indeed, the first studies searching for diet solutions started with Jerry Cornfield who introduced it during WW2 (1941-1945). He formulated '*The Diet Problem'* for the Army searching for a low-cost diet that would meet the nutritional needs of soldiers.
During this period, the Royal Air Force and other parts of the army were hiring mathematicians to solve the important diet problem and to plan affordable meals. Among the researchers involved in solving this problem was George Dantzig. He proposed a new algorithm he had developed. It took him until 1947, being the first to deliver the correct mathematical result. Dantzig tested his model on his own diet, constructing a database with 50 types of food. He wanted to reduce his caloric intake to 1,500 kcal and programmed an objective function to maximize the feeling of being full. The solution he found was a weird diet with 200 bouillon cubes per day. This was possible because the former nutritional requirements didn't show a limit to the amount of salt. These results led to upper bounds being added to linear programming for the first time. Until now the approach has been used in many ways to design individual diets as well as population diets. The problem of the diet is interesting, because it is difficult to optimize the function of phenomenon like the diet, as it is composed of several variables: energy density, water content, macronutrients, micronutrients, etc.

Most solutions have been developed in 2000 or later with the rise of calculation capacities, becoming available to solve linear problems. Meanwhile, new nutritional, social, cost, and environmental constraints appeared.

The objective of this assignment is to study a diet problem, illustrating linear and non-linear programming. Using this topic, different analysis will be performed throughout the study.

2 Linear Programming Problem

2.1 Problem Description

LIVESTOCK FEEDING PROBLEM

A farmer decides to prepare food every day for his 200 cows. Due to his background, he acquired some techniques about food preparation, and he figured out that the best way to do that was to plan the quantity of food per portion, considering one portion per animal and per day. In a daily portion, he mixes up two different products: corn and fodder.

The recipe is easy, but he must respect some rules. First, a portion must weigh at least 400 grams, and contains at least 30% of proteins, and 15% of fiber maximum. Secondly, the portions must be enough nutritive but not too much for the cows. The farmer considers that a portion must contain between 600 and 1000 calories.

Apart from that, he tries to spend less and less money for the food. Indeed, due to the Covid crisis, prices skyrocketed. Therefore, his main goal is to minimize the total price each day. The final objective is to find the appropriate repartition of corn and fodder per daily portion to minimize the total daily price. To prepare his portion, he uses the following table reporting all the information about quantities and prices:

Product	**Quantity of *protein* (g)**	**Quantity of *fiber* (g)**	**Cost (£)**	**Calories (cal)**
Corn (1g)	0.09	0.02	0.0015	1.3
Fodder (1g)	0.6	0.06	0.0045	1.4

Each data is for 1g of product.

2.2 Decision variables

Let's x_1 be the quantity of corn in gram per daily portion, and x_2 the one of fodder also in gram per daily portion. These variables $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are the **decision variables.** Indeed, the objective is to find the appropriate repartition of corn and fodder per portion to minimize the total daily price.

2.3 Objective function

As the goal is to minimize the total daily price of food, let's Z be this total daily food price for all the cattle (200 cows).

To write it, it is needed to sum the prices for each product, considering the needed quantity per day (the decisions variables). Then, the factor 200 is there to consider all the cattle.

$$Z = 200\ (0.0015x_1\ + 0.0045\ x_2\ )$$

then $\boldsymbol{Z = 0.30x_1 + 0.90\ x_2}$

2.4 Problem constraints

Constraints associated to the problem can be written mathematically with the decisions variables:

1) *Portion weight constraint:*

$$x_1 + x_2 \geq 400$$
$$=> -\boldsymbol{x_1} - \boldsymbol{x_2} \leq \mathbf{-400}$$

2) *Calories lower constraint:*

$$1.3\, x_1 + 1.4\, x_2 \geq 600$$
$$=> -\mathbf{1.3}\, \boldsymbol{x_1} - \mathbf{1.4}\, \boldsymbol{x_2} \leq \mathbf{-600}$$

3) *Calories upper constraint:*

$$\mathbf{1.3}\, \boldsymbol{x_1} + \mathbf{1.4}\, \boldsymbol{x_2} \leq \mathbf{1000}$$

4) *Protein amount constraint:*

$$0.09\, x_1 + 0.6\, x_2 \geq 0.3(x_1 + x_2)$$
$$=> -0.21\, x_1 + 0.3\, x_2 \geq 0$$
$$=> \mathbf{0.21}\, \boldsymbol{x_1} - \mathbf{0.3}\, \boldsymbol{x_2} \leq \mathbf{0}$$

5) *Fiber amount constraint:*

$$0.02\, x_1 + 0.06\, x_2 \leq 0.05(x_1 + x_2)$$
$$=> -\mathbf{0.03}\, \boldsymbol{x_1} + \mathbf{0.01}\, \boldsymbol{x_2} \leq \mathbf{0}$$

6) *Boundary constraints:*

$$\boldsymbol{x_1} \geq \mathbf{0}\, ;\, \boldsymbol{x_2} \geq \mathbf{0}$$

2.5 Mathematical formulation

Consequently, the linear optimization problem is the following:

$$\textbf{min} \quad \boldsymbol{Z} = \mathbf{0.30}\boldsymbol{x_1} + \mathbf{0.90}\, \boldsymbol{x_2}$$

$$s.t. \begin{cases} -x_1 - x_2 \leq -400 \\ -1.3\, x_1 - 1.4\, x_2 \leq -600 \\ 1.3\, x_1 + 1.4\, x_2 \leq 1000 \\ 0.21 x_1 - 0.3\, x_2 \leq 0 \\ -0.03\, x_1 + 0.01\, x_2 \leq 0 \\ x_1 \geq 0\, ;\, x_2 \geq 0 \end{cases}$$

2.6 Graphical Method

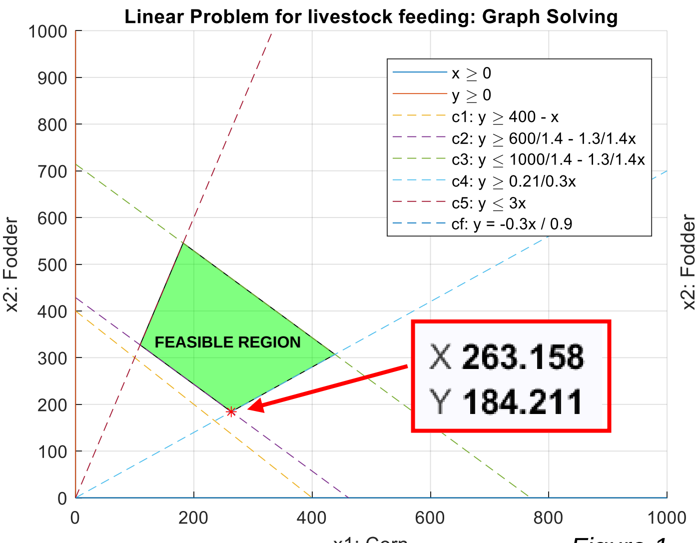


Figure 1

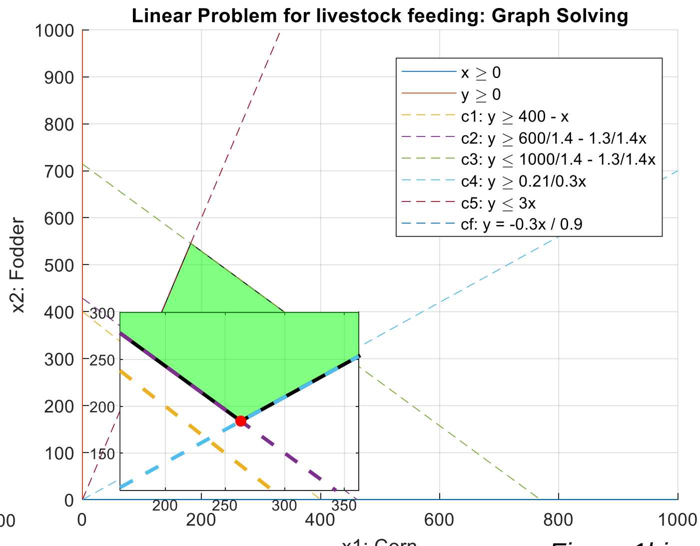


Figure 1bis

Using Matlab, a 2D graph is plotted *Figure 1*, considering all the constraints of the problem. By observation, the **optimal solution** seems to be **263,16 grams** of corn and **184.21 grams** of fodder to buy daily per portion to pay the minimal price possible.
The *magnify* function shows closer the optimal point (*Figure 1 bis*). It is a **corner-point feasible solution** which reveals itself be optimal.
The optimal point is located at the intersection of the second constraint and the fourth constraint. These constraints are **binding constraints** with **sensitive parameters**. Indeed, any sufficiently small change in its coefficients would change the optimal solution.
All the codes used for plotting the graph are presented on the next page.

MATLAB code File 1:

```
% Cost function to minimize
f = [0.30 0.90];

% Constraints matrix
A = [-1     -1    ;
    -1.3   -1.4  ;
     1.3    1.4  ;
     0.21  -0.3  ;
    -0.03   0.01 ;] ;

b = [-400 -600 1000 0 0];


%% Solving by Graphical Method
figure(1)
lb = [0 0]; % lower bounds
ub = [1000 1000]; % upper bound (for the plot)
[val, fval] = linprog(f, A, b, [], [], lb, []);
plotregion(-A,-b,lb,[],'g',0.5) % other file
xlabel('x1: Corn'), ylabel('x2: Fodder')
axis([lb(1) ub(1) lb(2) ub(2)]), grid
title('Linear Problem for livestock feeding: Graph Solving')
hold on

% Constraints lines:
x1 = lb(1):1000;
x2 = lb(2):1000;
x2_1 = 400 - x1;
x2_2 = 600/1.4 - 1.3/1.4*x1 ;
x2_3 = 1000/1.4 - 1.3/1.4*x1;
x2_4 = 0.21/0.3*x1;
x2_5 = 3*x1;

% Optimal cost function line:
x2_cf = -(0.3.*x1) / 0.9;
obj = plot(x1, zeros(size(x1)), zeros(size(x2)), x2, x1, x2_1, x1, x2_2, x1, x2_3, x1, x2_4, x1, x2_5, x1, x2_cf);
obj(3).LineStyle = '--';
obj(4).LineStyle = '--';
obj(5).LineStyle = '--';
obj(6).LineStyle = '--';
obj(7).LineStyle = '--';
obj(8).LineStyle = '--';
% plot optimal solution:
plot(val(1), val(2), 'r*')
hold off % releases graph
% set axes and figure legend:
legend(obj, {'x \geq 0', 'y \geq 0', ...
   'c1: y \geq 400 - x', ...
   'c2: y \geq 600/1.4 - 1.3/1.4x', ...
   'c3: y \leq 1000/1.4 - 1.3/1.4x', ...
   'c4: y \geq 0.21/0.3x', ...
   'c5: y \leq 3x', ...
   'cf: y = -0.3x / 0.9'}, ...
   'Location', 'Best');
```

MATLAB code File 2: (***plotregion*** function from another file)

```
function plotregion(A,b,lb,ub,c,transp,points,linetyp,start_end)
% The function plotregion plots closed convex regions in 2D/3D. The region
% is formed by the matrix A and the vectors lb and ub such that Ax>=b
% and lb<=x<=ub, where the region is the set of all feasible x (in R^2 or R^3).
% An option is to plot points in the same plot.
%
% Usage:   plotregion(A,b,lb,ub,c,transp,points,linetyp,start_end)
%
% Input:
%
% A - matrix. Set A to [] if no A exists
% b - vector. Set b to [] if no b exists
% lb - (optional) vector. Set lb to [] if no lb exists
% ub - (optional) vector. Set ub to [] if no ub exists
% c - (optional) color, example 'r' or [0.2 0.1 0.8], {'r','y','b'}.
%       Default is a random colour.
% transp - (optional) is a measure of how transparent the
%           region will be. Must be between 0 and 1. Default is 0.5.
% points - (optional) points in matrix form. (See example)
% linetyp - (optional) How the points will be marked.
%           Default is 'k-'.
% start_end - (optional) If a special marking for the first and last point
%              is needed.
%
% If several regions must be plotted A ,b, lb, ub and c can be stored as a cell array {}
% containing all sub-set information (see example1.m and example3.m).
%
% Written by Per Bergström 2006-01-16
if nargin<2
    error('Too few arguements for plotregion');
elseif nargin<6
    transp=0.5;
end
if iscell(A)

    if isempty(A{1})
        m=0;
        n=max(length(lb),length(ub));
    else
        [m,n]=size(A{1});
    end
    [lll,p]=size(A);
    for i=1:p
        if size(b{i},1)==1
            b{i}=b{i}';
        end

        if nargin>2
            if not(isempty(lb))
                if not(isempty(lb{i}))
                    if size(lb{i},1)==1
                        lb{i}=lb{i}';
                    end
                    A{i}=[A{i};eye(n)];
                    b{i}=[b{i};lb{i}];
                end
            end
        end
        if nargin>3
            if not(isempty(ub))
                if not(isempty(ub{i}))
```

```
                if size(ub{i},1)==1
                    ub{i}=ub{i}';
                end
                A{i}=[A{i};-eye(n)];
                b{i}=[b{i};-ub{i}];
            end
        end
    end
end
if nargin<5
    c=cell(1,p);
    for i=1:p
        c{i}=[rand rand rand];
    end
end

if nargin>=5
    if isempty(c);
        c=cell(1,p);
        for i=1:p
            c{i}=[rand rand rand];
        end
    else
        for i=1:p
            if isempty(c{i});
                c{i}=[rand rand rand];
            end
        end
    end
end


warning off
if n==2
    eq=cell(1,p);
    X=cell(1,p);
    for pp=1:p
        eq{pp}=zeros(2,1);
        X{pp}=zeros(2,1);
        [m,n]=size(A{pp});

        for i=1:(m-1)
            for j=(i+1):m
                try
                    x=A{pp}([i j],:)\b{pp}([i j]);
                    if and(min((A{pp}*x-b{pp}))>-1e-6,min((A{pp}*x-b{pp}))<Inf)
                        X{pp}=[X{pp},x];
                        eq{pp}=[eq{pp},[i j]'];
                    end
                end
            end
        end
    end

    xmi=min(X{1}(1,2:end));
    xma=max(X{1}(1,2:end));
    ymi=min(X{1}(2,2:end));
    yma=max(X{1}(2,2:end));

    for pp=2:p
        xmi=min(min(X{pp}(1,2:end)),xmi);
        xma=max(max(X{pp}(1,2:end)),xma);
        ymi=min(min(X{pp}(2,2:end)),ymi);
        yma=max(max(X{pp}(2,2:end)),yma);
```

```
    end

    if nargin>=7
        xmi2=min(points(1,:));
        xma2=max(points(1,:));
        ymi2=min(points(2,:));
        yma2=max(points(2,:));

        xmi=min(xmi,xmi2);
        xma=max(xma,xma2);
        ymi=min(ymi,ymi2);
        yma=max(yma,yma2);
    end

    axis([(xmi-0.1) (xma+0.1) (ymi-0.1) (yma+0.1)]);

    if nargin==7
        plot(points(1,:)',points(2,:)','k-');hold on;
    elseif nargin==8
        plot(points(1,:)',points(2,:)',linetyp);hold on;
    elseif nargin==9
        plot(points(1,:)',points(2,:)',linetyp);hold on;
        if length(start_end)==2
            plot(points(1,1)',points(2,1)',start_end);hold on;
            plot(points(1,end)',points(2,end)',start_end);hold on;
        elseif length(start_end)==4
            plot(points(1,1)',points(2,1)',start_end(1:2));hold on;
            plot(points(1,end)',points(2,end)',start_end(3:4));hold on;
        end
    end

    for pp=1:p
        [rad,col]=size(X{pp});
        xm=mean(X{pp}(:,2:end),2);
        Xdiff=X{pp}(:,2:end);

        for j=1:(col-1)
            Xdiff(:,j)=Xdiff(:,j)-xm;
            Xdiff(:,j)=Xdiff(:,j)/norm(Xdiff(:,j));
        end
        costhe=zeros((col-1),1);

        for j=1:(col-1)
            costhe(j)=Xdiff(:,1)'*Xdiff(:,j);
        end

        [cc,ind]=min(abs(costhe));
        ref2=Xdiff(:,ind(1))-(Xdiff(:,ind(1))'*Xdiff(:,1))*Xdiff(:,1);
        ref2=ref2'/norm(ref2);

        for j=1:(col-1)
            if ref2*Xdiff(:,j)<0
                costhe(j)=-2-costhe(j);
            end
        end
        [sooo,ind3]=sort(costhe);
        set(patch(X{pp}(1,ind3+1)',X{pp}(2,ind3+1)',c{pp}),'FaceAlpha',transp);
    end

elseif n==3
    eq=cell(1,p);
    X=cell(1,p);
    for pp=1:p
        eq{pp}=zeros(3,1);
```

```
    X{pp}=zeros(3,1);
    [m,n]=size(A{pp});

    for i=1:(m-2)
        for j=(i+1):(m-1)
            for k=(j+1):m
                try
                    x=A{pp}([i j k],:)\b{pp}([i j k]);
                    if and(min((A{pp}*x-b{pp}))>-1e-6,min((A{pp}*x-b{pp}))<Inf)
                        X{pp}=[X{pp},x];
                        eq{pp}=[eq{pp},[i j k]'];
                    end

                end
            end
        end
    end
end

xmi=min(X{1}(1,2:end));
xma=max(X{1}(1,2:end));
ymi=min(X{1}(2,2:end));
yma=max(X{1}(2,2:end));
zmi=min(X{1}(3,2:end));
zma=max(X{1}(3,2:end));

for pp=2:p
    xmi=min(min(X{pp}(1,2:end)),xmi);
    xma=max(max(X{pp}(1,2:end)),xma);
    ymi=min(min(X{pp}(2,2:end)),ymi);
    yma=max(max(X{pp}(2,2:end)),yma);
    zmi=min(min(X{pp}(3,2:end)),zmi);
    zma=max(max(X{pp}(3,2:end)),zma);
end

if nargin>=7
    xmi2=min(points(1,:));
    xma2=max(points(1,:));
    ymi2=min(points(2,:));
    yma2=max(points(2,:));
    zmi2=min(points(3,:));
    zma2=max(points(3,:));

    xmi=min(xmi,xmi2);
    xma=max(xma,xma2);
    ymi=min(ymi,ymi2);
    yma=max(yma,yma2);
    zmi=min(zmi,zmi2);
    zma=max(zma,zma2);
end

axis([(xmi-0.1) (xma+0.1) (ymi-0.1) (yma+0.1) (zmi-0.1) (zma+0.1)]);

if nargin==7
    plot3(points(1,:)',points(2,:)',points(3,:)','k-');hold on;
elseif nargin==8
    plot3(points(1,:)',points(2,:)',points(3,:)',linetyp);hold on;
elseif nargin==9
    plot3(points(1,:)',points(2,:)',points(3,:)',linetyp);hold on;
    if length(start_end)==2
        plot3(points(1,1)',points(2,1)',points(3,1)',start_end);hold on;
        plot3(points(1,end)',points(2,end)',points(3,end)',start_end);hold on;
    elseif length(start_end)==4
        plot3(points(1,1)',points(2,1)',points(3,1)',start_end(1:2));hold on;
```

```
                plot3(points(1,end)',points(2,end)',points(3,end)',start_end(3:4));hold on;
            end
        end

        for pp=1:p
            [m,n]=size(A{pp});
            for i=1:m
                [ind1,ind2]=find(eq{pp}==i);
                lind2=length(ind2);
                if lind2>0
                    xm=mean(X{pp}(:,ind2),2);
                    Xdiff=X{pp}(:,ind2);
                    for j=1:lind2
                        Xdiff(:,j)=Xdiff(:,j)-xm;
                        Xdiff(:,j)=Xdiff(:,j)/norm(Xdiff(:,j));
                    end
                    costhe=zeros(lind2,1);

                    for j=1:lind2
                        costhe(j)=Xdiff(:,1)'*Xdiff(:,j);
                    end

                    [cc,ind]=min(abs(costhe));
                    ref2=Xdiff(:,ind(1))-(Xdiff(:,ind(1))'*Xdiff(:,1))*Xdiff(:,1);
                    ref2=ref2'/norm(ref2);

                    for j=1:lind2
                        if ref2*Xdiff(:,j)<0
                            costhe(j)=-2-costhe(j);
                        end
                    end
                    [sooo,ind3]=sort(costhe);

set(patch(X{pp}(1,ind2(ind3))',X{pp}(2,ind2(ind3))',X{pp}(3,ind2(ind3))',c{pp}),'FaceAlpha',
transp);
                end
            end

        end
    else
        error('not 2D or 3D');
    end

else      %  A is not a cell

    if isempty(A)
        m=0;
        n=max(length(lb),length(ub));
    else
        [m,n]=size(A);
    end

    if size(b,1)==1
        b=b';
    end

    if nargin>2
        if not(isempty(lb))
            if size(lb,1)==1
                lb=lb';
            end
            A=[A;eye(n)];
            b=[b;lb];
            m=m+n;
```

```
        end
    end
    if nargin>3
        if not(isempty(ub))
            if size(ub,1)==1
                ub=ub';
            end

            A=[A;-eye(n)];
            b=[b;-ub];
            m=m+n;
        end
    end

    if nargin<5
        c=[rand rand rand];
    end

    if nargin>=5
        if isempty(c);
            c=[rand rand rand];
        end
    end

    warning off
    if n==2
        eq=zeros(2,1);
        X=zeros(2,1);
        for i=1:(m-1)
            for j=(i+1):m
                try
                    x=A([i j],:)\b([i j]);
                    if and(min((A*x-b))>-1e-6,min((A*x-b))<Inf)
                        X=[X,x];
                        eq=[eq,[i j]'];
                    end
                end
            end
        end

        xmi=min(X(1,2:end));
        xma=max(X(1,2:end));
        ymi=min(X(2,2:end));
        yma=max(X(2,2:end));

        if nargin>=7
            xmi2=min(points(1,:));
            xma2=max(points(1,:));
            ymi2=min(points(2,:));
            yma2=max(points(2,:));

            xmi=min(xmi,xmi2);
            xma=max(xma,xma2);
            ymi=min(ymi,ymi2);
            yma=max(yma,yma2);
        end
        axis([(xmi-0.1) (xma+0.1) (ymi-0.1) (yma+0.1)]);

        if nargin==7
            plot(points(1,:)',points(2,:)','k-');hold on;
        elseif nargin==8
            plot(points(1,:)',points(2,:)',linetyp);hold on;
        elseif nargin==9
            plot(points(1,:)',points(2,:)',linetyp);hold on;
```

```
        if length(start_end)==2
            plot(points(1,1)',points(2,1)',start_end);hold on;
            plot(points(1,end)',points(2,end)',start_end);hold on;
        elseif length(start_end)==4
            plot(points(1,1)',points(2,1)',start_end(1:2));hold on;
            plot(points(1,end)',points(2,end)',start_end(3:4));hold on;
        end
    end

    [rad,col]=size(X);
    xm=mean(X(:,2:end),2);
    Xdiff=X(:,2:end);

    for j=1:(col-1)
        Xdiff(:,j)=Xdiff(:,j)-xm;
        Xdiff(:,j)=Xdiff(:,j)/norm(Xdiff(:,j));
    end
    costhe=zeros((col-1),1);

    for j=1:(col-1)
        costhe(j)=Xdiff(:,1)'*Xdiff(:,j);
    end

    [cc,ind]=min(abs(costhe));
    ref2=Xdiff(:,ind(1))-(Xdiff(:,ind(1))'*Xdiff(:,1))*Xdiff(:,1);
    ref2=ref2'/norm(ref2);

    for j=1:(col-1)
        if ref2*Xdiff(:,j)<0
            costhe(j)=-2-costhe(j);
        end
    end
    [sooo,ind3]=sort(costhe);
    set(patch(X(1,ind3+1)',X(2,ind3+1)',c),'FaceAlpha',transp);

elseif n==3
    eq=zeros(3,1);
    X=zeros(3,1);

    for i=1:(m-2)
        for j=(i+1):(m-1)
            for k=(j+1):m
                try
                    x=A([i j k],:)\b([i j k]);
                    if and(min((A*x-b))>-1e-6,min((A*x-b))<Inf)
                        X=[X,x];
                        eq=[eq,[i j k]'];
                    end

                end
            end
        end
    end
    xmi=min(X(1,2:end));
    xma=max(X(1,2:end));
    ymi=min(X(2,2:end));
    yma=max(X(2,2:end));
    zmi=min(X(3,2:end));
    zma=max(X(3,2:end));

    if nargin>=7
        xmi2=min(points(1,:));
        xma2=max(points(1,:));
        ymi2=min(points(2,:));
```

```
            yma2=max(points(2,:));
            zmi2=min(points(3,:));
            zma2=max(points(3,:));

            xmi=min(xmi,xmi2);
            xma=max(xma,xma2);
            ymi=min(ymi,ymi2);
            yma=max(yma,yma2);
            zmi=min(zmi,zmi2);
            zma=max(zma,zma2);
        end

        axis([(xmi-0.1) (xma+0.1) (ymi-0.1) (yma+0.1) (zmi-0.1) (zma+0.1)]);

        if nargin==7
            plot3(points(1,:)',points(2,:)',points(3,:)','k-');hold on;
        elseif nargin==8
            plot3(points(1,:)',points(2,:)',points(3,:)',linetyp);hold on;
        elseif nargin==9
            plot3(points(1,:)',points(2,:)',points(3,:)',linetyp);hold on;
            if length(start_end)==2
                plot3(points(1,1)',points(2,1)',points(3,1)',start_end);hold on;
                plot3(points(1,end)',points(2,end)',points(3,end)',start_end);hold on;
            elseif length(start_end)==4
                plot3(points(1,1)',points(2,1)',points(3,1)',start_end(1:2));hold on;
                plot3(points(1,end)',points(2,end)',points(3,end)',start_end(3:4));hold on;
            end
        end

        for i=1:m
            [ind1,ind2]=find(eq==i);
            lind2=length(ind2);
            if lind2>0
                xm=mean(X(:,ind2),2);
                Xdiff=X(:,ind2);
                for j=1:lind2
                    Xdiff(:,j)=Xdiff(:,j)-xm;
                    Xdiff(:,j)=Xdiff(:,j)/norm(Xdiff(:,j));
                end
                costhe=zeros(lind2,1);

                for j=1:lind2
                    costhe(j)=Xdiff(:,1)'*Xdiff(:,j);
                end

                [cc,ind]=min(abs(costhe));
                ref2=Xdiff(:,ind(1))-(Xdiff(:,ind(1))'*Xdiff(:,1))*Xdiff(:,1);
                ref2=ref2'/norm(ref2);

                for j=1:lind2
                    if ref2*Xdiff(:,j)<0
                        costhe(j)=-2-costhe(j);
                    end
                end
                [sooo,ind3]=sort(costhe);

set(patch(X(1,ind2(ind3))',X(2,ind2(ind3))',X(3,ind2(ind3))',c),'FaceAlpha',transp);
            end
        end
    else
        error('Your region must be in 2D or 3D!');
    end

end
```

MATLAB code File 3: (***magnify*** function from another file)

```
function magnify(f1)
% Created by RC
%magnify(f1)
%
%  Figure creates a magnification box when under the mouse
%  position when a button is pressed.  Press '+'/'-' while
%  button pressed to increase/decrease magnification. Press
%  '>'/'<' while button pressed to increase/decrease box size.
%  Hold 'Ctrl' while clicking to leave magnification on figure.
%
%  Example:
%     plot(1:100,randn(1,100),(1:300)/3,rand(1,300)), grid on,
%     magnify;

% Rick Hindman - 7/29/04

if (nargin == 0), f1 = gcf; end;
set(f1, ...
   'WindowButtonDownFcn',  @ButtonDownCallback, ...
   'WindowButtonUpFcn', @ButtonUpCallback, ...
   'WindowButtonMotionFcn', @ButtonMotionCallback, ...
   'KeyPressFcn', @KeyPressCallback);
return;

function ButtonDownCallback(src,eventdata)
   f1 = src;
   a1 = get(f1,'CurrentAxes');
   a2 = copyobj(a1,f1);

   set(f1, ...
      'UserData',[f1,a1,a2], ...
      'Pointer','fullcrosshair', ...
      'CurrentAxes',a2);
   set(a2, ...
      'UserData',[2,0.2], ...  %magnification, frame size
      'Color',get(a1,'Color'), ...
      'Box','on');
   xlabel(''); ylabel(''); zlabel(''); title('');
   set(get(a2,'Children'), ...
      'LineWidth', 2);
   set(a1, ...
      'Color',get(a1,'Color')*0.95);
   set(f1, ...
      'CurrentAxes',a1);
   ButtonMotionCallback(src);
return;

function ButtonUpCallback(src,eventdata)
   H = get(src,'UserData');
   f1 = H(1); a1 = H(2); a2 = H(3);
   set(a1, ...
      'Color',get(a2,'Color'));
   set(f1, ...
      'UserData',[], ...
      'Pointer','arrow', ...
      'CurrentAxes',a1);
   if ~strcmp(get(f1,'SelectionType'),'alt'),
      delete(a2);
   end;
return;
```

```
function ButtonMotionCallback(src,eventdata)
   H = get(src,'UserData');
   if ~isempty(H)
      f1 = H(1); a1 = H(2); a2 = H(3);
      a2_param = get(a2,'UserData');
      f_pos = get(f1,'Position');
      a1_pos = get(a1,'Position');

      [f_cp, a1_cp] = pointer2d(f1,a1);

      set(a2,'Position',[(f_cp./f_pos(3:4)) 0 0]+a2_param(2)*a1_pos(3)*[-1 -1 2 2]);
      a2_pos = get(a2,'Position');

   set(a2,'XLim',a1_cp(1)+(1/a2_param(1))*(a2_pos(3)/a1_pos(3))*diff(get(a1,'XLim'))*[-0.5
0.5]);
   set(a2,'YLim',a1_cp(2)+(1/a2_param(1))*(a2_pos(4)/a1_pos(4))*diff(get(a1,'YLim'))*[-0.5
0.5]);
   end;
return;

function KeyPressCallback(src,eventdata)
   H = get(gcf,'UserData');
   if ~isempty(H)
      f1 = H(1); a1 = H(2); a2 = H(3);
      a2_param = get(a2,'UserData');
      if (strcmp(get(f1,'CurrentCharacter'),'+') | strcmp(get(f1,'CurrentCharacter'),'='))
         a2_param(1) = a2_param(1)*1.2;
      elseif (strcmp(get(f1,'CurrentCharacter'),'-') |
strcmp(get(f1,'CurrentCharacter'),'_'))
         a2_param(1) = a2_param(1)/1.2;
      elseif (strcmp(get(f1,'CurrentCharacter'),'<') |
strcmp(get(f1,'CurrentCharacter'),','))
         a2_param(2) = a2_param(2)/1.2;
      elseif (strcmp(get(f1,'CurrentCharacter'),'>') |
strcmp(get(f1,'CurrentCharacter'),'.'))
         a2_param(2) = a2_param(2)*1.2;
      end;
      set(a2,'UserData',a2_param);
   ButtonMotionCallback(src);
   end;
return;

% Included for completeness (usually in own file)
function [fig_pointer_pos, axes_pointer_val] = pointer2d(fig_hndl,axes_hndl)
%
%pointer2d(fig_hndl,axes_hndl)
%
% Returns the coordinates of the pointer (in pixels)
% in the desired figure (fig_hndl) and the coordinates
%       in the desired axis (axes coordinates)
%
% Example:
%  figure(1),
%  hold on,
%  for i = 1:1000,
%     [figp,axp]=pointer2d;
%     plot(axp(1),axp(2),'.','EraseMode','none');
%     drawnow;
%  end;
%  hold off

% Rick Hindman - 4/18/01

if (nargin == 0), fig_hndl = gcf; axes_hndl = gca; end;
```

```
if (nargin == 1), axes_hndl = get(fig_hndl,'CurrentAxes'); end;

set(fig_hndl,'Units','pixels');

pointer_pos = get(0,'PointerLocation'); %pixels {0,0} lower left
fig_pos = get(fig_hndl,'Position'); %pixels {l,b,w,h}

fig_pointer_pos = pointer_pos - fig_pos([1,2]);
set(fig_hndl,'CurrentPoint',fig_pointer_pos);

if (isempty(axes_hndl)),
axes_pointer_val = [];
elseif (nargout == 2),
axes_pointer_line = get(axes_hndl,'CurrentPoint');
axes_pointer_val = sum(axes_pointer_line)/2;
end;
```

2.7 Matlab Implementation

To solve this part of the problem, it is needed to implement the following code using the *linprog* function on Matlab:

```
% Cost function to minimize
f = [0.30 0.90];

% Constraints matrix
A = [-1     -1    ;
    -1.3   -1.4  ;
     1.3    1.4  ;
     0.21  -0.3  ;
    -0.03   0.01 ;] ;

b = [-400 -600 1000 0 0];

[val, fval] = linprog(f, A, b, [], [], lb, [])
```

2.8 Solution

Matlab provides the solution for the problem as the following:

```
val =
263.1579
184.2105

fval =
244.7368
```

Relating to the problem, it means that the best food balance of products to buy each day is **263.16 grams** of corn and **184.21 grams** of fodder per portion. Doing this, all the rules will be respected, and the farmer will pay **244.74** £ per day for all the cattle which is the less possible.

2.9 Sensitivity Analysis

Graphical Method and Sensitivity Analysis:

Sensitivity 1:

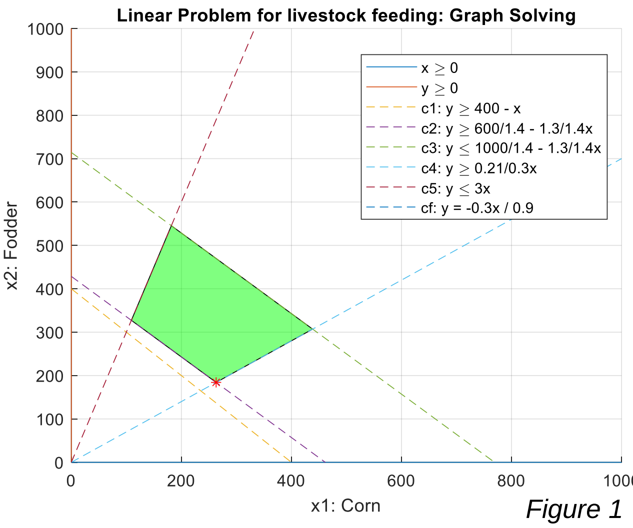


Figure 1

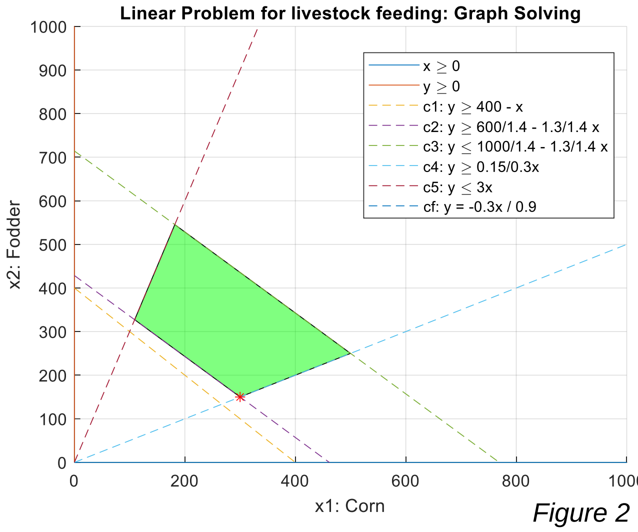


Figure 2

As mentioned before, the **binding constraints** are visually the second constraint and the fourth one. In the sensitivity analysis, the objective is to alter some parameters and observe the consequences on the optimal solution. Of course, any line changes would affect the feasible region, but only changes of **sensitive parameters** would affect the optimal solution.

To start, let's try for instance to alter the fourth constraint (blue line). The current constraint is the line: $y \geq \frac{0.21}{0.3} x$

To illustrate this sensitivity, the coefficient 0.21 becomes 0.15. What does it mean for the problem? This change symbolizes that the protein apport for corn is no longer 0.09 g but 0.15 g per gram of corn:

Protein amount constraint:

$$0.15\, x_1 + 0.6\, x_2 \geq 0.3(x_1 + x_2)$$
$$=> -0.15\, x_1 + 0.3\, x_2 \geq 0$$
$$=> \mathbf{0.15\, x_1 - 0.3\, x_2 \leq 0}$$

With the fourth constraint as $y \geq \frac{0.15}{0.3} x$, the graph becomes the one on *Figure 2*.

Besides, the optimal solution changes into:

val =
300.0000
150.0000

fval =
225.0000

That means due to these changes, if it was real, from now on, the farmer should buy **300 grams** of corn and **150 grams** of fodder daily for each portion of his 200 cows. The new minimum price would be **225** £ per day.

Consequently, it can be concluded that increasing the protein apport for corn would make the farmer gain money. Indeed, he would gain 244.74 – 225 = **19,74** £ **per day**. It is understandable since then the farmer should buy less seeds to reach the required protein threshold in the portions.

Sensitivity 2:

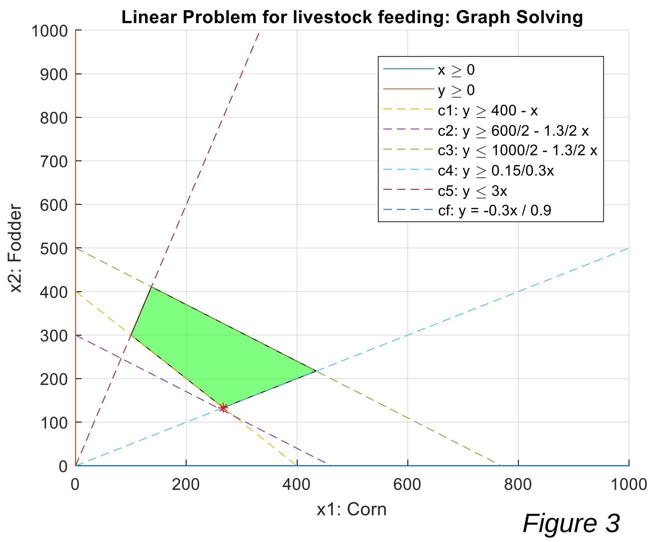


Figure 3

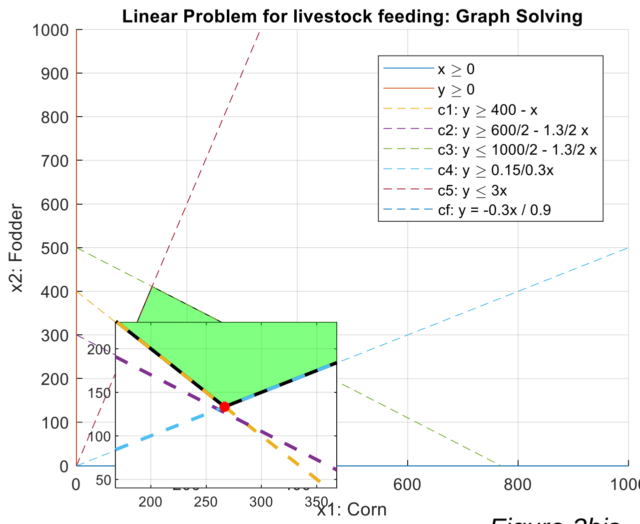


Figure 3bis

Now the fourth constraint has just been altered, what would happen if the second constraint (purple line) was changed in turn? Graphically, it is predicable that if the purple line goes down, it may be the orange line which would become a binding constraint instead of the purple one.
Let's check this altering the purple line. The current second constraint is the line: $y \geq \frac{600}{1.4} - \frac{1.3}{1.4}x$

To do the changes, the coefficient 1.4 becomes 2. What does it mean for the problem? It symbolizes that the calories apport for fodder is no longer 1.4 calories per gram of fodder but 2. These changes also affect the third constraint $y \geq \frac{1000}{1.4} - \frac{1.3}{1.4}x$:

Calories lower constraint:

$$1.3\ x_1\ + \color{red}{2}\ x_2\ \geq 600$$
$$=> -\mathbf{1.3}\ \boldsymbol{x_1}\ - \color{red}{\mathbf{2}}\ \boldsymbol{x_2}\ \leq \mathbf{-600}$$

Calories upper constraint:

$$\mathbf{1.3}\ \boldsymbol{x_1}\ + \color{red}{\mathbf{2}}\ \boldsymbol{x_2}\ \leq \mathbf{1000}$$

With the second constraint as $y \geq \frac{600}{2} - \frac{1.3}{2}x$ and the third one as $y \geq \frac{1000}{2} - \frac{1.3}{2}x$. The graph becomes the one on *Figure 3.*

Then, the optimal solution changes into:

val =
266.6667
133.3333
fval =
200.0000

It's indeed the orange constraint which prevails now over the purple one. If these changes were applied, the farmer should buy **266.67 grams** of corn and **133.33 grams** of fodder per day for each portion of his 200 cows. The new minimum price would be **200** £ per day.
Consequently, it can be concluded that increasing the calories apport for fodder would make the farmer gain money once again (244.74 – 200 = **44,74 £ per day**). It is understandable since then the farmer should buy less seeds to reach the required calories threshold in the portions.

Sensitivity 3:

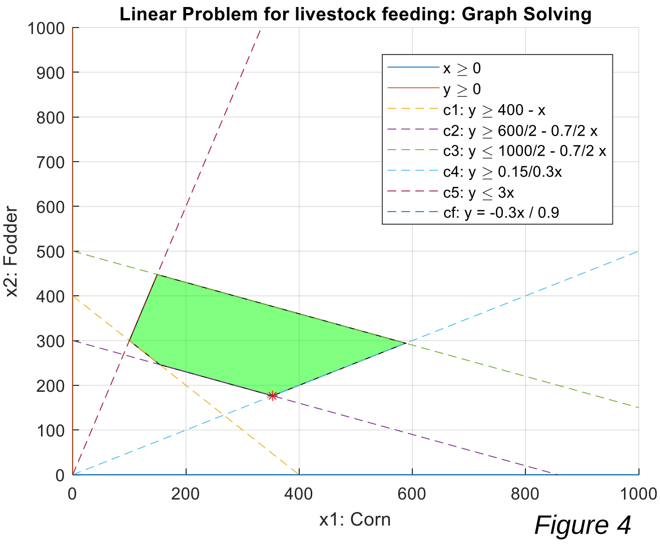


Figure 4

Finally, let's make a last alteration to make sure that the purple line returns to be a bounding constraint. For this, some parameters of the second constraint $y \geq \frac{600}{2} - \frac{1.3}{2}x$ change.

The coefficient 1.3 becomes 0.7. What does it mean for the problem? This symbolizes that the calories apport for corn is no longer 1.3 calories per gram of corn but 0.7. These changes also affect the third constraint $y \geq \frac{1000}{2} - \frac{1.3}{2}x$:

Calories lower constraint:

$$0.7\,x_1 + 2\,x_2 \geq 600$$
$$=> -\mathbf{0.7}\,\boldsymbol{x_1} - \mathbf{2}\,\boldsymbol{x_2} \leq \mathbf{-600}$$

Calories upper constraint:

$$\mathbf{0.7}\,\boldsymbol{x_1} + \mathbf{2}\,\boldsymbol{x_2} \leq \mathbf{1000}$$

Then, the second constraint becomes: $y \geq \frac{600}{2} - \frac{0.7}{2}x$ and the third one as $y \geq \frac{1000}{2} - \frac{0.7}{2}x$.The graph becomes the one on *Figure 4.*

Then, the optimal solution changes into:

```
val =
  352.9412
  176.4706

fval =
  264.7059
```

That means now, due to these changes, now the farmer must buy **352.94 grams** of corn and **176.47 grams** of fodder daily for each portion of his 200 cows. The new minimum price is **264.71** £ per day. Consequently, it can be concluded that decreasing the calories apport for corn would make the farmer lose money compared to the previous case. It is comprehensive since then the farmer must buy more seeds to reach the required calories threshold in the portions.

This sensitivity analysis proves that the data of the problem must be very accurate, due to the sensitivity of the problem. Indeed, a little change in a sensitive parameter and the optimal solution can vary a lot. Clearly, the farmer can't afford not to have the exact calories and intake data. Data must be estimated with special care to minimize as much as possible his total daily price.
All the codes used for the analysis are presented on the next page.

MATLAB code File 1: (***magnify*** function and ***plotregion*** function opened in other files)

```
%% Sensitivity Analysis 1

% Cost function to minimize
f = [0.30 0.90];

% New constraint matrix
Abis = [-1     -1    ;
        -1.3   -1.4  ;
         1.3    1.4  ;
         0.15  -0.3  ;
        -0.03   0.01 ;] ;

bbis = [-400 -600 1000 0 0];

figure(2)
lb = [0 0]; % lower bounds
ub = [1000 1000]; % upper bound (for the plot)
[valbis, fvalbis] = linprog(f, Abis, bbis, [], [], lb, []);
plotregion(-Abis,-bbis,lb,[],'g',0.5) % other file
xlabel('x1: Corn'), ylabel('x2: Fodder')
axis([lb(1) ub(1) lb(2) ub(2)]), grid
title('Linear Problem for livestock feeding: Graph Solving')
hold on

% Constraints lines:
x1 = lb(1):1000;
x2 = lb(2):1000;
x2_1 = 400 - x1;
x2_2 = 600/1.4 - 1.3/1.4*x1 ;
x2_3 = 1000/1.4 - 1.3/1.4*x1;
x2_4 = 0.15/0.3*x1;
x2_5 = 3*x1;

% Optimal cost function line:
x2_cf = -(0.3.*x1) / 0.9;
obj = plot(x1, zeros(size(x1)), zeros(size(x2)), x2, x1, x2_1, x1, x2_2, x1, x2_3, x1, x2_4,
x1, x2_5, x1, x2_cf);
obj(3).LineStyle = '--';
obj(4).LineStyle = '--';
obj(5).LineStyle = '--';
obj(6).LineStyle = '--';
obj(7).LineStyle = '--';
obj(8).LineStyle = '--';
% plot optimal solution:
plot(valbis(1), valbis(2), 'r*');
hold off % releases graph
% set axes and figure legend:
legend(obj, {'x \geq 0', 'y \geq 0', ...
    'c1: y \geq 400 - x', ...
    'c2: y \geq 600/1.4 - 1.3/1.4 x', ...
    'c3: y \leq 1000/1.4 - 1.3/1.4 x', ...
    'c4: y \geq 0.15/0.3x', ...
    'c5: y \leq 3x', ...
    'cf: y = -0.3x / 0.9'}, ...
    'Location', 'Best');

[valbis, fvalbis] = linprog(f, Abis, bbis, [], [], lb, []);
disp('val = ');
disp(valbis);
disp('fval = ');
disp(fvalbis);
```

```matlab
%% Sensitivity Analysis 2

% Cost function to minimize
f = [0.30 0.90];

% New constraint matrix
Abis2 = [-1     -1    ;
        -1.3   -2  ;
         1.3    2  ;
         0.15  -0.3  ;
        -0.03   0.01 ;] ;

bbis2 = [-400 -600 1000 0 0];

figure(3)
lb = [0 0]; % lower bounds
ub = [1000 1000]; % upper bound (for the plot)
[valbis2, fvalbis2] = linprog(f, Abis2, bbis2, [], [], lb, []);
plotregion(-Abis2,-bbis2,lb,[],'g',0.5) % other file
xlabel('x1: Corn'), ylabel('x2: Fodder')
axis([lb(1) ub(1) lb(2) ub(2)]), grid
title('Linear Problem for livestock feeding: Graph Solving')
hold on

% Constraints lines:
x1 = lb(1):1000;
x2 = lb(2):1000;
x2_1 = 400 - x1;
x2_2 = 600/2 - 1.3/2*x1 ;
x2_3 = 1000/2 - 1.3/2*x1;
x2_4 = 0.15/0.3*x1;
x2_5 = 3*x1;

% Optimal cost function line:
x2_cf = -(0.3.*x1) / 0.9;
obj = plot(x1, zeros(size(x1)), zeros(size(x2)), x2, x1, x2_1, x1, x2_2, x1, x2_3, x1, x2_4,
x1, x2_5, x1, x2_cf);
obj(3).LineStyle = '--';
obj(4).LineStyle = '--';
obj(5).LineStyle = '--';
obj(6).LineStyle = '--';
obj(7).LineStyle = '--';
obj(8).LineStyle = '--';
% plot optimal solution:
plot(valbis2(1), valbis2(2), 'r*');
hold off % releases graph
% set axes and figure legend:
legend(obj, {'x \geq 0', 'y \geq 0', ...
    'c1: y \geq 400 - x', ...
    'c2: y \geq 600/2 - 1.3/2 x', ...
    'c3: y \leq 1000/2 - 1.3/2 x', ...
    'c4: y \geq 0.15/0.3x', ...
    'c5: y \leq 3x', ...
    'cf: y = -0.3x / 0.9'}, ...
    'Location', 'Best');

[valbis2, fvalbis2] = linprog(f, Abis2, bbis2, [], [], lb, []);
disp('val = ');
disp(valbis2);
disp('fval = ');
disp(fvalbis2);
```

```matlab
%% Sensitivity Analysis 3

% Cost function to minimize
f = [0.30 0.90];

% New constraint matrix
Abis3 = [-1      -1     ;
        -0.7    -2   ;
         0.7     2   ;
         0.15   -0.3   ;
        -0.03    0.01 ;] ;

bbis3 = [-400 -600 1000 0 0];

figure(4)
lb = [0 0]; % lower bounds
ub = [1000 1000]; % upper bound (for the plot)
[valbis3, fvalbis3] = linprog(f, Abis3, bbis3, [], [], lb, []);
plotregion(-Abis3,-bbis3,lb,[],'g',0.5) % other file
xlabel('x1: Corn'), ylabel('x2: Fodder')
axis([lb(1) ub(1) lb(2) ub(2)]), grid
title('Linear Problem for livestock feeding: Graph Solving')
hold on

% Constraints lines:
x1 = lb(1):1000;
x2 = lb(2):1000;
x2_1 = 400 - x1;
x2_2 = 600/2 - 0.7/2*x1 ;
x2_3 = 1000/2 - 0.7/2*x1;
x2_4 = 0.15/0.3*x1;
x2_5 = 3*x1;

% Optimal cost function line:
x2_cf = -(0.3.*x1) / 0.9;
obj = plot(x1, zeros(size(x1)), zeros(size(x2)), x2, x1, x2_1, x1, x2_2, x1, x2_3, x1, x2_4,
x1, x2_5, x1, x2_cf);
obj(3).LineStyle = '--';
obj(4).LineStyle = '--';
obj(5).LineStyle = '--';
obj(6).LineStyle = '--';
obj(7).LineStyle = '--';
obj(8).LineStyle = '--';
% plot optimal solution:
plot(valbis3(1), valbis3(2), 'r*');
hold off % releases graph
% set axes and figure legend:
legend(obj, {'x \geq 0', 'y \geq 0', ...
    'c1: y \geq 400 - x', ...
    'c2: y \geq 600/2 - 0.7/2 x', ...
    'c3: y \leq 1000/2 - 0.7/2 x', ...
    'c4: y \geq 0.15/0.3x', ...
    'c5: y \leq 3x', ...
    'cf: y = -0.3x / 0.9'}, ...
    'Location', 'Best');

[valbis3, fvalbis3] = linprog(f, Abis3, bbis3, [], [], lb, []);
disp('val = ');
disp(valbis3);
disp('fval = ');
disp(fvalbis3);
```

3 Mixed-integer Programming Problem

3.1 Problem Description

Recently, new products have been introduced on the market. From now on, farmers have the possibility to buy concentrate seeds to add in their portions. These seeds also contain proteins and fibers, and are sold as a part of cubes, called *Extra seed cube* and *Super seed cube*. One cube weighs 2 grams, and it is sold at a unit. The new concentrate seeds could reveal themselves a boon for him because they are cheaper than the corn and the fodder. A relative told him that it would be convenient to try these new products but keeping at least 75 grams of corn and 75 grams of fodder per portion, to avoid any unpleasant surprise of the product at the beginning. The farmer agrees that it could be the good deal. Besides, he decides to introduce at least 2 cubes per portion.

From now on, the farmer needs to find a new balance between each of his products per portion, keeping all the previous rules concerning the protein, fiber, and calories apports. The objective is to find the appropriate repartition of corn, fodder, and cubes per portion to minimize the total price.

Product	**Quantity of *protein* (g)**	**Quantity of *fiber* (g)**	**Cost (£)**	**Calories (cal)**
***Extra seed cube* (1 cube)**	1	0.2	0.00075	2
***Super seed cube* (1 cube)**	1.2	0.18	0.001	2

Each data is for 1 cube.

3.2 Decision variables

Let's again x_1 be the quantity of corn in gram per daily portion, and x_2 the one of fodder also in gram per daily portion. But this time, let's include c_1 being the number of *Extra seed cubes* per daily portion, and c_2 the number of *Super seed cubes* per daily portion.

These variables $\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{c_1}, \boldsymbol{c_2}$ are the **decision variables.** Indeed, the objective is to find the appropriate repartition of corn, fodder, and cubes per portion to minimize the total price.

3.3 Objective function

As the goal is to minimize the total daily price of food, let's again Z be this total daily food price for all the cattle (200 cows).

To write it, the same sum is kept, but adding the prices for each cube, considering the needed quantity per daily portion, and this for all the cattle.

Due to this new arrangement, the cost function Z becomes:

$$Z = 200\,(0.0015x_1 + 0.0045\,x_2 + 0.00075\,c_1 + 0.001\,c_2)$$

$$\text{then}\quad \boldsymbol{Z = 0.30x_1 + 0.90\,x_2 + 0.15\,c_1 + 0.2\,c_2}$$

3.4 Problem constraints

Then, constraints associated to the new problem can be written mathematically with the new decisions variables:

1) *Portion weight constraint:*

$$x_1 + x_2 + 2c_1 + 2c_2 \geq 400$$

$$=> \mathbf{-x_1 - x_2 - 2c_1 - 2c_2 \leq -400}$$

2) *Calories lower constraint:*

$$1.3\,x_1 + 1.4\,x_2 + 2c_1 + 2c_2 \geq 600$$

$$=> \boldsymbol{-1.3\ x_1 - 1.4\,x_2 - 2\,c_1 - 2\,c_2 \leq -600}$$

3) *Calories upper constraint:*

$$\boldsymbol{1.3\,x_1 + 1.4\,x_2 + 2\,c_1 + 2\,c_2 \leq 1000}$$

4) *Protein amount constraint:*

$$0.09\,x_1 + 0.6\,x_2 + c_1 + 1.2\,c_2 \geq 0.3(x_1 + x_2 + 2c_1 + 2\,c_2)$$

$$=> -0.21\,x_1 + 0.3\,x_2 + 0.4c_1 + 0.6\,c_2 \geq 0$$

$$=> \boldsymbol{0.21\,x_1 - 0.3\,x_2 - 0.4\,c_1 - 0.6\,c_2 \leq 0}$$

5) *Fiber amount constraint:*

$$-0.02\,x_1 + 0.06\,x_2 + 0.2\,c_1 + 0.18\,c_2 \leq 0.05(x_1 + x_2 + 2c_1 + 2\,c_2)$$

$$=> \boldsymbol{-0.03\,x_1 + 0.02\,x_2 + 0.1\,c_1 + 0.08\,c_2 \leq 0}$$

6) *Number of cubes constraint:*

$$c_1 + c_2 \geq 2 => \mathbf{-c_1 - c_2 \leq 2}$$

7) *Boundary constraints:*

$$\boldsymbol{x_1 \geq 75\,;\, x_2 \geq 75\,;\, c_1 \geq 1\,;\, c_2 \geq 1}$$

8) *Integer variables:*

$$\boldsymbol{x_1\,;\, x_2\,;\, c_1\,;\, c_2}$$

3.5 Mathematical formulation

Consequently, the linear optimization problem is the following:

$$\textbf{min} \quad Z = 0.30x_1 + 0.90\,x_2 + 0.15\,c_1 + 0.2\,c_2$$

$$\begin{cases} \textit{s.t.} \quad -x_1 - x_2 - 2c_1 - 2\,c_2 \leq -400 \\ -1.3\ x_1 - 1.4\,x_2 - 2\,c_1 - 2\,c_2 \leq -600 \\ 1.3\,x_1 + 1.4\,x_2 + 2\,c_1 + 2\,c_2 \leq 1000 \\ 0.21\,x_1 - 0.3\,x_2 - 0.4\,c_1 - 0.6\,c_2 \leq 0 \\ -0.03\,x_1 + 0.01\,x_2 + 0.1\,c_1 + 0.08\,c_2 \leq 0 \\ -c_1 - c_2 \leq 2 \\ x_1 \geq 75\,;\, x_2 \geq 75\,;\, c_1 \geq 1\,;\, c_2 \geq 1 \\ c_1 \text{ and } c_2 \text{ are integers} \end{cases}$$

3.6 Matlab Implementation

To solve this part of the problem, it is needed to implement the following code using the *intlinprog* function on Matlab:

```
% Cost function to minimize
f1 = [0.30 0.90 0.15 0.2];

% Constraint matrix
A1 = [-1     -1    -2    -2;
     -1.3   -1.4  -2    -2;
      1.3    1.4   2     2;
      0.21  -0.3  -0.4  -0.6;
     -0.03   0.01  0.1   0.08;
      0      0    -1    -1] ;

b1 = [-400 -600 1000 0 0 2];

%% Solving by intlinprog
lb1 = [75, 75, 1, 1];
ic = (3:4);
[val1, fval1] = intlinprog(f1, ic, A1, b1, [], [], lb1, []);
disp('val = ');
disp(val1);
disp('fval = ');
disp(fval1);
```

3.7 Solution

Matlab provides the solution for the problem as the following:

```
val =
  263.8462
   75.0000
   54.0000
   22.0000

fval =
  159.1538
```

With the new data, the best food balance of products to buy each day, is **263.16 grams** of corn, **184.21 grams** of fodder, **54** Extra seed cubes and **22** Super seed cubes per portion. Doing this, the farmer will pay **159.15** £ per day for all the cattle.
Introducing these both new products, the farmer **gains** 244.74 – 159.15 = **85.59** £ **each day.**

3.8 Sensitivity Analysis

Sensitivity 1:

Compared to the previous part, there are four decisions variables. This means the 2D graph won't be represented and used to the sensitivity analysis. Consequently, the results by themselves will be studied and compared.

Let's observe what happens to the results applying the same changes as in the first sensitivity in the previous part. The fourth constraint is changed without altering the parameters of the new decision variables c_1 and c_2 . The protein apport for corn is no longer 0.09 grams but 0.15 grams per gram of corn.

Protein amount constraint:

$$\mathbf{0.15\,x_1 - 0.3\,x_2 - 0.4\,c_1 - 0.6\,c_2 \leq 0}$$

Then, the optimal solution changes into:

val =
267.0000
75.0000
67.0000
7.0000

fval =
159.0500

As the optimal solution changed, it can be concluded that the fourth constraint remained **binding constraints** and its coefficients **sensitive parameters**. It's logical since new decisions variables added keep the previous state. Some data have been introduced without totally overwhelm the previous one.

If these changes were real, the farmer should buy **267 grams** of corn, **75 grams** of fodder, **67** Extra seed cubes and **7** Super seed cubes, daily for each portion of his 200 cows. The new minimum price would be **159.05** £ per day.
Consequently, it can be concluded that increasing the protein apport for corn would once more make the farmer gains money, such as 159.15 - 159.05 = **0.10**£ **each day**. It is understandable since then the farmer should buy less seeds to reach the required proteins threshold in the portions.

Sensitivity 2:

Any similar changes concerning the sensitive parameters linked to x_1 and x_2 as in the previous part would also affect this problem. Now, let's see what happen if you change some parameters associated with both new decision variables c_1 and c_2 .

Let's suppose the number of calories contained in each cube goes from 2 to 3. It is a small difference but let's check if our optimal solution varies in this case, and if these parameters are sensitive.

The second and the third constraint are then affected:

Calories lower constraint:

$$-\mathbf{1.3}\ \boldsymbol{x_1} - \mathbf{1.4}\,\boldsymbol{x_2} - \mathbf{3}\,\mathbf{c_1} - \mathbf{3}\,\mathbf{c_2} \leq -\mathbf{600}$$

Calories upper constraint:

$$\mathbf{1.3}\,\boldsymbol{x_1} + \mathbf{1.4}\,\boldsymbol{x_2} + \mathbf{3}\,\mathbf{c_1} + \mathbf{3}\,\mathbf{c_2} \leq \mathbf{1000}$$

Then, the optimal solution becomes:

```
val =
  217.0000
   75.0000
    4.0000
   67.0000

fval =
  146.6000
```

Because the optimal solution varied, the second and the third constraint are also **binding constraints** and their coefficients **sensitive parameters**.

In this case, the farmer should buy **217 grams** of corn, **75 grams** of fodder, **4** Extra seed cubes and **67** Super seed cubes, daily for each portion of his 200 cows. The new minimum price would be **146.60** £ per day.

Consequently, it can be concluded that increasing the calories apport for both cubes would again make the farmer gain money equal to 159.15 – 146.60 = **12.55** £ **each day**. It is comprehensive since then the farmer should buy less seeds to reach the required calories threshold in the portions.

Sensitivity 3:

Finally, what would happen if the farmer hasn't decided to introduce at least 2 cubes per portion. Mathematically, this means removing the sixth constraint. Further, what if there weren't the constraints about the minimal number of cubes (at least 1 of each per portion), introduced to force the trying of the new product.

Then, the sixth constraint is removed, and the boundary constraints become:

Boundary constraints:

$$\boldsymbol{x_1} \geq \mathbf{75} \;;\; \boldsymbol{x_2} \geq \mathbf{75} \;;\; \boldsymbol{c_1} \geq \mathbf{0} \;;\; \boldsymbol{c_2} \geq \mathbf{0}$$

In this case, the optimal solution becomes:

```
val =
  263.8462
   75.0000
   54.0000
   22.0000

fval =
  159.1538
```

The result is the same as the first optimal solution for this part. Consequently, even without forcing to introduce cubes, it would have been a better deal to try the new product.

All the codes used for the analysis are presented on the next page.

MATLAB code File 1:

```
%% Sensitivity Analysis Integer 1
% Cost function to minimize
f1 = [0.30 0.90 0.15 0.2];
% Constraint matrix
A1bis = [-1     -1    -2    -2;
        -1.3   -1.4  -2    -2;
         1.3    1.4   2     2;
         0.15  -0.3  -0.4  -0.6;
        -0.03   0.01  0.1   0.08;
         0      0     -1    -1] ;
b1bis = [-400 -600 1000 0 0 2];

% Solving by intlinprog
lb1 = [75, 75, 1, 1];
ic = (3:4);
[val1bis, fval1bis] = intlinprog(f1, ic, A1bis, b1bis, [], [], lb1, []);
disp('val = ');
disp(val1bis);
disp('fval = ');
disp(fval1bis);

%% Sensitivity Analysis Integer 2
% Cost function to minimize
f1 = [0.30 0.90 0.15 0.2];
% Constraint matrix
A1bis1 = [-1     -1    -2    -2;
         -1.3   -1.4  -3    -3;
          1.3    1.4   3     3;
          0.21  -0.3  -0.4  -0.6;
         -0.03   0.01  0.1   0.08;
          0      0     -1    -1] ;
b1bis1 = [-400 -600 1000 0 0 2];

% Solving by intlinprog
lb1 = [75, 75, 1, 1];
ic = (3:4);
[val1bis1, fval1bis1] = intlinprog(f1, ic, A1bis1, b1bis1, [], [], lb1, []);
disp('val = ');
disp(val1bis1);
disp('fval = ');
disp(fval1bis1);

%% Sensitivity Analysis Integer 3
% Cost function to minimize
f1 = [0.30 0.90 0.15 0.2];
% Constraint matrix
A1bis2 = [-1     -1    -2    -2;
        -1.3   -1.4  -2    -2;
         1.3    1.4   2     2;
         0.21  -0.3  -0.4  -0.6;
        -0.03   0.01  0.1   0.08];
b1bis2 = [-400 -600 1000 0 0];

% Solving by intlinprog
lb1bis2 = [75, 75, 0, 0];
ic = (3:4);
[val1bis2, fval1bis2] = intlinprog(f1, ic, A1bis2, b1bis2, [], [], lb1bis2, []);
disp('val = ');
disp(val1bis2);
disp('fval = ');
disp(fval1bis2);
```

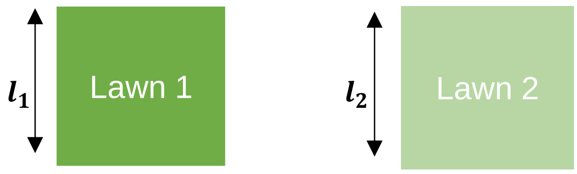
4 Non-Linear Programming Problem

4.1 Problem Description

After finding out that the introduction of new products could decrease the total daily price, the farmer decides to go further introducing once again two new testing products. These products are somehow special. Indeed, these are pieces of artificial lawns sold per square meter. These lawns are fitted to be set in the barn. Each lawn contains grass providing protein and fiber apports for cattle. One square meter of lawn weights 3 grams in terms of the weight of food eaten by the animal. Lawns can be an intermediate intake between other seed contributions of the day. To prevail the testing of the new lawns, the farmer limits the number of seed cubes by 50 each. However, he estimates that 25m^2 of lawn maximum per animal is enough. Of course, he also keeps all the previous rules concerning the protein, fiber, and calories apports.To simplify the problem, let's suppose if the animals eat all the surface eatable of their lawn in a day.
The objective is to find the appropriate repartition of corn, fodder, cubes, and the length of the lawn squares per portion to minimize the total price.

Product	**Quantity of *protein* (g)**	**Quantity of *fiber* (g)**	**Cost (£)**	**Calories (cal)**
Lawn type 1 (1 m^2)	0.4	0.23	0.0025	6
Lawn type 2 (1 m^2)	0.5	0.21	0.0035	7

Each data is for 1 m^2 of lawn.



4.2 Decision variables

Let's again x_1 be the quantity of corn in gram per daily portion, and x_2 the one of fodder also in gram per daily portion, c_1 the number of *Extra seed cubes* per daily portion, and c_2 the number of *Super seed cubes* per daily portion. Let's add l_1 the length of the lawn type 1 square size in meter per daily portion (per animal each day), and l_2 the length of the lawn type 2 square size in meter per daily portion.
These variables $\boldsymbol{x_1}$, $\boldsymbol{x_2}, \boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{l_1}, \boldsymbol{l_2}$ are the **decision variables.** Indeed, the objective is to find the appropriate repartition of corn, fodder, cubes, and the lawn square sizes per portion to minimize the total price.

4.3 Objective function

As the goal is to minimize the total daily price of food, let's again Z be this total daily food price for all the cattle (200 cows).
To write it, the same sum is kept, but adding the prices for each piece of lawn, considering the square's lengths in meter per daily portion, and this for all the cattle.

Due to this new arrangement, the cost function Z becomes:

$$Z = 200\left(0.0015x_1 + 0.0045\,x_2 + 0.00075\,c_1 + 0.001\,c_2 + 0.0025\,{l_1}^2 + 0.0035\,{l_2}^2\right)$$

then $$\boldsymbol{Z = 0.30x_1 + 0.90\,x_2 + 0.15\,c_1 + 0.2\,c_2 + 0.5\,{l_1}^2 + 0.7\,{l_2}^2}$$

4.4 Problem constraints

Then, constraints associated to the new problem can be written mathematically with the new decisions variables:

1) *Portion weight constraint:*

$$x_1 + x_2 + 2c_1 + 2c_2 + 3l_1^2 + 3l_2^2 \geq 400$$

$$\Rightarrow \mathbf{-x_1 - x_2 - 2c_1 - 2c_2 - 3l_1^2 - 3l_2^2 \leq -400}$$

2) *Calories lower constraint:*

$$1.3x_1 + 1.4x_2 + 2c_1 + 2c_2 + 6l_1^2 + 7l_2^2 \geq 600$$

$$\Rightarrow \mathbf{-1.3x_1 - 1.4x_2 - 2c_1 - 2c_2 - 6l_1^2 - 7l_2^2 \leq -600}$$

3) *Calories upper constraint:*

$$\mathbf{1.3x_1 + 1.4x_2 + 2c_1 + 2c_2 + 6l_1^2 + 7l_2^2 \leq 1000}$$

4) *Protein amount constraint:*

$$0.09x_1 + 0.6x_2 + c_1 + 1.2c_2 + 0.4l_1^2 + 0.5l_2^2 \geq 0.3(x_1 + x_2 + 2c_1 + 2c_2 + 3l_1^2 + 3l_2^2)$$

$$\Rightarrow -0.21x_1 + 0.3x_2 + 0.4c_1 + 0.6c_2 - 0.5l_1^2 - 0.4l_2^2 \geq 0$$

$$\Rightarrow \mathbf{0.21x_1 - 0.3x_2 - 0.4c_1 - 0.6c_2 + 0.5l_1^2 + 0.4l_2^2 \leq 0}$$

5) *Fiber amount constraint:*

$$-0.02x_1 + 0.06x_2 + 0.2c_1 + 0.18c_2 + 0.23l_1^2 + 0.21l_2^2 \leq 0.05(x_1 + x_2 + 2c_1 + 2c_2 + 3l_1^2 + 3l_2^2)$$

$$\Rightarrow \mathbf{-0.03x_1 + 0.02x_2 + 0.1c_1 + 0.08c_2 + 0.08l_1^2 + 0.06l_2^2 \leq 0}$$

6) *Lower boundary constraints:*

$$\mathbf{x_1 \geq 75\,;\ x_2 \geq 75\,;\ c_1 \geq 0\,;\ c_2 \geq 0\,;\ l_1^2 \geq 1\,;\ l_2^2 \geq 1}$$

7) *Upper boundary constraints:*

$$\mathbf{x_1 \leq 400\,;\ x_2 \leq 400\,;\ c_1 \leq 50\,;\ c_2 \leq 50\,;\ l_1^2 \leq 5\,;\ l_2^2 \leq 5}$$

8) *Integer variables:*

$$\mathbf{x_1\,;\ x_2\,;\ c_1\,;\ c_2\,;\ l_1\,;\ l_2}$$

Note: upper bounds for x_1 and x_2 are introduced to enable convergence towards an optimal solution using the Genetic Algorithm later.

4.5 Mathematical formulation

Consequently, the linear optimization problem is the following:

$$\min \quad Z = 0.30x_1 + 0.90x_2 + 0.15c_1 + 0.2c_2 + 0.5l_1^2 + 0.7l_2^2$$

$$\text{s.t.} \begin{cases} -x_1 - x_2 - 2c_1 - 2c_2 - 3l_1^2 - 3l_2^2 \leq -400 \\ -1.3x_1 - 1.4x_2 - 2c_1 - 2c_2 - 6l_1^2 - 7l_2^2 \leq -600 \\ 1.3x_1 + 1.4x_2 + 2c_1 + 2c_2 + 6l_1^2 + 7l_2^2 \leq 1000 \\ 0.21x_1 - 0.3x_2 - 0.4c_1 - 0.6c_2 + 0.5l_1^2 + 0.4l_2^2 \leq 0 \\ -0.03x_1 + 0.01x_2 + 0.1c_1 + 0.08c_2 + 0.08l_1^2 + 0.06l_2^2 \leq 0 \\ x_1 \geq 75\,;\ x_2 \geq 75\,;\ c_1 \geq 0\,;\ c_2 \geq 0\,;\ l_1^2 \geq 1\,;\ l_2^2 \geq 1 \end{cases}$$

$$x_1 \leq 400\,;\ x_2 \leq 400\,;\ c_1 \leq 50\,;\ c_2 \leq 50\,;\ l_1^2 \leq 5\,;\ l_2^2 \leq 5$$

$$c_1, c_2, l_1, \text{and } l_2 \text{ are integers}$$

4.6 Matlab Implementation

In this case, there are integer constraints in a nonlinear optimization problem. This is called a mixed-integer nonlinear optimization problem. As it is difficult to solve and the method for this type of optimization problems hasn't been explored during the lectures, the result will be found by approximation.
The *fmincon* function can't solve nonlinear programming with integer decision variables. Hence, the global optimization method, genetic algorithm (GA), becomes more suitable to solve the problem. However, at each run, the GA doesn't provide the same result, due to its randomness.

To solve this part of the problem, it is needed to implement the following code using the *genetic algorithm* function on Matlab:

```
% Cost function to minimize
fun = @(x)0.3*x(1) + 0.9*x(2) + 0.15*x(3) + 0.2*x(4) + 0.5*x(5)^2 + 0.7*x(6)^2;

% Initial point
x0 = [75 75 1 1 1 1];

% Lower boundaries
lb = [75 75 0 0 1 1];
ub = [400 400 50 50 5 5];

% x(3), x(4), x(5), x(6) are integers
intcon = [3 4 5 6];

nonlcon = @ellipsecons;
% ga_options = optimoptions('ga','PlotFcn', @gaplotbestf);
ga_options = optimoptions('ga');

% Running the Genetic Algorithm 40 times
S = zeros(40,6);
F = zeros(1,40);
for i = 1:40
    [sol, fval] = ga(fun,6,[],[],[],[],lb,ub,nonlcon,intcon,ga_options);
    S(i,1:6) = [sol];
    F(1,i) = fval;
end

% Constraints
function [c, ceq] = ellipsecons(x)
c(1) = -x(1)-x(2)-2*x(3)-2*x(4)-3*x(5)^2-3*x(6)^2+400;
c(2) = -1.3*x(1)-1.4*x(2)-2*x(3)-2*x(4)-6*x(5)^2-7*x(6)^2+600;
c(3) = 1.3*x(1)+1.4*x(2)+2*x(3)+2*x(4)+6*x(5)^2+7*x(6)^2-1000;
c(4) = 0.21*x(1)-0.3*x(2)-0.4*x(3)-0.6*x(4)+0.5*x(5)^2+0.4*x(6)^2;
c(5) = -0.03*x(1)+0.01*x(2)+0.1*x(3)+0.08*x(4)+0.08*x(5)^2+0.06*x(6)^2;
ceq = [];
end
```

4.7 Solution

Therefore, let's run the code 40 times, to find after analysis a near-optimal solution.

Results of the 40 iterations:

Index	x_1	x_2	c_1	c_2	l_1	l_2	Cost function value
1	230.364563670625	75.3816829035274	4	50	2	3	155.852883714362
2	245.628306583643	75.0867089336248	15	47	2	2	157.716530015355
3	252.313558929177	75.0432695157317	17	50	1	2	159.083010242912
4	228.880078233663	80.1127547391540	4	47	2	3	159.065502735338
5	239.861446522830	75.1354371706293	7	50	1	3	157.430327410415
6	252.742652087239	75.3116983693379	18	48	1	2	159.203324158576
7	225.469435315439	88.4919762943199	12	36	2	3	164.583609259520
8	196.784393866422	104.413886987010	1	30	2	4	172.357816448235
9	253.310802298508	75.0009879059630	25	41	1	2	158.744129804919
10	230.412733755699	75.7081353552996	5	49	2	3	156.111141946479
11	228.098402404060	80.3365007659416	9	43	2	3	158.982371410565
12	238.605320766036	75.0215241270270	12	46	1	3	156.900967944135
13	229.247682503219	75.0276484637741	5	50	2	3	155.349188368363
14	245.401482219278	75.0232231643432	15	47	2	2	157.591345513692
15	242.575003764489	76.2386704616272	11	48	3	1	157.837304544811
16	229.395963920949	75.0257796008667	6	49	2	3	155.341990817065
17	245.408744883679	75.0055265727272	17	45	2	2	157.477597380558
18	264.614274642615	75.0003502545163	37	32	1	1	160.034597621849
19	251.572487539953	75.0022298351811	18	49	1	2	158.773753113649
20	229.108727506118	75.1167961992110	5	50	2	3	155.387734831125
21	251.901922502310	76.0903597250560	21	45	1	2	159.501900503243
22	245.374500760229	75.0086882745831	15	47	2	2	157.570169675194
23	193.801008357744	112.898353782474	4	23	2	4	178.148820911550
24	245.537789645158	75.0238466970268	17	45	2	2	157.532798920872
25	245.411702775125	75.1561610726030	17	45	2	2	157.614055797880
26	237.279065566377	75.2500796245533	12	47	1	3	156.908791332011
27	245.731932106507	75.0434897638125	17	45	2	2	157.608720419383
28	236.921695857426	75.0034762462453	13	46	1	3	156.529637378849
29	256.427076847346	76.5154015128501	24	40	2	1	160.091984415769
30	245.127064095127	75.3297557628183	13	49	2	2	157.884899415075
31	261.511819817223	75.0240634770662	26	45	1	1	160.075203074526
32	244.897747656275	78.9142525583388	20	40	2	2	160.292151599387
33	245.655950279921	75.0548838536975	17	45	2	2	157.596180552304
34	218.121001721728	91.2007458498618	0	46	2	3	165.016971781394
35	245.418667345163	75.0083797138093	17	45	2	2	157.483141945977
36	237.109926987556	75.0746906839056	11	48	1	3	156.750199711782
37	229.302284170748	75.0011183935250	6	49	2	3	155.291691805397
38	255.379277063655	75.0071413438206	24	42	2	1	158.820210328535
39	246.837432272260	75.1013723362154	18	43	2	2	157.742464784272
40	264.615595720524	75.0058475736801	37	32	1	1	160.039941532469

The cost function values are quite similar but a bit different due to the randomness of the algorithm.
Of course, all these results are different at each running of 40 times. But here a package of 40 runs is fixed to illustrate the example.

After that, it is necessary to filter solutions, deleting those which are not feasible. For the non-feasible solutions, Matlab replies: *'Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance but constraints are not satisfied.'*:

```
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
and constraint violation is less than options.ConstraintTolerance.
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
and constraint violation is less than options.ConstraintTolerance.
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
and constraint violation is less than options.ConstraintTolerance.
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
but constraints are not satisfied.
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
and constraint violation is less than options.ConstraintTolerance.
Optimization terminated: average change in the penalty fitness value less than options.FunctionTolerance
and constraint violation is less than options.ConstraintTolerance.
```

In this example, the message is displayed for the index values: 4 ; 29 ; 34.
Then, it remains only 37 feasible solutions:

Index	x_1	x_2	c_1	c_2	l_1	l_2	Cost function value
1	230.364563670625	75.3816829035274	4	50	2	3	155.852883714362
2	245.628306583643	75.0867089336248	15	47	2	2	157.716530015355
3	252.313558929177	75.0432695157317	17	50	1	2	159.083010242912
~~4~~	~~228.880078233663~~	~~80.1127547391540~~	~~4~~	~~47~~	~~2~~	~~3~~	~~159.065502735338~~
5	239.861446522830	75.1354371706293	7	50	1	3	157.430327410415
6	252.742652087239	75.3116983693379	18	48	1	2	159.203324158576
7	225.469435315439	88.4919762943199	12	36	2	3	164.583609259520
8	196.784393866422	104.413886987010	1	30	2	4	172.357816448235
9	253.310802298508	75.0009879059630	25	41	1	2	158.744129804919
10	230.412733755699	75.7081353552996	5	49	2	3	156.111141946479
11	228.098402404060	80.3365007659416	9	43	2	3	158.982371410565
12	238.605320766036	75.0215241270270	12	46	1	3	156.900967944135
13	229.247682503219	75.0276484637741	5	50	2	3	155.349188368363
14	245.401482219278	75.0232231643432	15	47	2	2	157.591345513692
15	242.575003764489	76.2386704616272	11	48	3	1	157.837304544811
16	229.395963920949	75.0257796008667	6	49	2	3	155.341990817065
17	245.408744883679	75.0055265727272	17	45	2	2	157.477597380558
18	264.614274642615	75.0003502545163	37	32	1	1	160.034597621849
19	251.572487539953	75.0022298351811	18	49	1	2	158.773753113649
20	229.108727506118	75.1167961992110	5	50	2	3	155.387734831125
21	251.901922502310	76.0903597250560	21	45	1	2	159.501900503243
22	245.374500760229	75.0086882745831	15	47	2	2	157.570169675194
23	193.801008357744	112.898353782474	4	23	2	4	178.148820911550
24	245.537789645158	75.0238466970268	17	45	2	2	157.532798920872
25	245.411702775125	75.1561610726030	17	45	2	2	157.614055797880
26	237.279065566377	75.2500796245533	12	47	1	3	156.908791332011
27	245.731932106507	75.0434897638125	17	45	2	2	157.608720419383
28	236.921695857426	75.0034762462453	13	46	1	3	156.529637378849
~~29~~	~~256.427076847346~~	~~76.5154015128501~~	~~24~~	~~40~~	~~2~~	~~1~~	~~160.091984415769~~
30	245.127064095127	75.3297557628183	13	49	2	2	157.884899415075
31	261.511819817223	75.0240634770662	26	45	1	1	160.075203074526
32	244.897747656275	78.9142525583388	20	40	2	2	160.292151599387
33	245.655950279921	75.0548838536975	17	45	2	2	157.596180552304
~~34~~	~~218.121001721728~~	~~91.2007458498618~~	~~0~~	~~46~~	~~2~~	~~3~~	~~165.016971781394~~
35	245.418667345163	75.0083797138093	17	45	2	2	157.483141945977
36	237.109926987556	75.0746906839056	11	48	1	3	156.750199711782
37	229.302284170748	75.0011183935250	6	49	2	3	155.291691805397
38	255.379277063655	75.0071413438206	24	42	2	1	158.820210328535
39	246.837432272260	75.1013723362154	18	43	2	2	157.742464784272
40	264.615595720524	75.0058475736801	37	32	1	1	160.039941532469

It is interesting to analyze more deeply the results. Below, the statistical table gathers some measures of the results obtained with Excel:

Cost function results analysis:

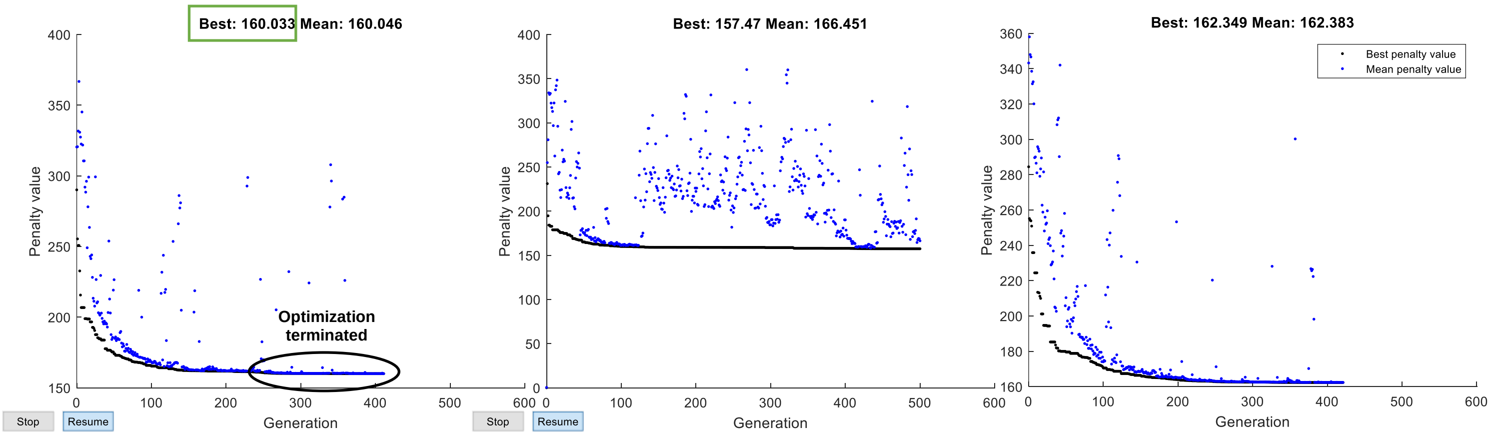
Measures	Corresponding point						
Decision Variables	x_1	x_2	c_1	c_2	l_1	l_2	**Cost function**
Minimum of the cost function	229,30228	75,00112	6	49	2	3	**155,29169**
Maximum of the cost function	193,80101	112,89835	4	23	2	4	**178,14882**
Average of the cost function	255.37928	75.00714	24	42	2	1	**158,81273**

Parameters' analysis:

Decision Variables	x_1	x_2	c_1	c_2	l_1	l_2	Cost function
Average (for each parameter)	**240,77652**	**77,57751**	**14,70270**	**44,35135**	**1,67568**	**2,32432**	**158,81273**
Standard Deviation (for each parameter)	**14,96303**	**7,96731**	**8,24912**	**6,28143**	**0,52989**	**0,78366**	**4,40231**

Tables shows parameters' variations, which increase for x_1 and x_2 as their values are higher. But the standard deviation of the cost function is not too high (4.40). Moreover, the average stays around 160£, but is a little distorted due to the maximum.

The function *Gaplotbestf* plots the best and mean score of the population at every generation regarding at the cost function and the violation of constraints. They provide a visualization of the performance of the solver at run time. It is displayed at each of the 40 runs. Here, three of them are summarized, and shows the algorithm reaches an optimal solution each time. It always revolves around 160 for the best result.



Consequently, the solution for this part is the one minimizing the cost function. Then, the best food balance of products to buy each day, is **229.30 grams** of corn, **75 grams** of fodder, **6** Extra seed cubes, **49** Super seed cubes, **2m²** Lawn type 1 and **3m²** lawns type 2 per portion. Doing this, the farmer will pay **155.29**£ per day for all the cattle. Introducing these both new products, the farmer **gains** 159.15 – 155.29 = **3.86** £ **each day** compared to the previous part**.**

4.8 Sensitivity Analysis

Sensitivity 1:

Reminding from the previous part, the previous binding constraints remain **binding constraints** and its coefficients **sensitive parameters**. New data about the lawn have been introduced without totally overwhelm the previous one.
Now, let's see what happen if you change some parameters associated with both new decision variables l_1 and l_2. Let's suppose there is a new offer about the lawn sell, then a square meter of lawn does no longer weigh 3 grams but 4 for the same price. Accordingly, the number of calories rises as follow:

Product	Quantity of *protein* (g)	Quantity of *fiber* (g)	Cost (£)	Calories (cal)
Lawn 4 grams type 1 (1 m^2)	0.4	0.23	0.0025	**8**
Lawn 4 grams type 2 (1 m^2)	0.5	0.21	0.0035	**9**

Each data is for 1 $m^2$of lawn.

Consequently, the mathematical problem becomes:

$$\textbf{min} \quad Z = 0.30x_1 + 0.90\,x_2 + 0.15\,c_1 + 0.2\,c_2 + 0.5\,l_1{}^2 + 0.7\,l_2{}^2$$

$$\textit{s.t.} \begin{cases} -x_1 - x_2 - 2c_1 - 2\,c_2 - \mathbf{4}\,l_1{}^2 - \mathbf{4}\,l_2{}^2 \le -400 \\ -1.3\ x_1 - 1.4\,x_2 - 2\,c_1 - 2\,c_2 - \mathbf{8}\,l_1{}^2 - \mathbf{9}\,l_2{}^2 \le -600 \\ 1.3\,x_1 + 1.4\,x_2 + 2\,c_1 + 2\,c_2 + \mathbf{8}\,l_1{}^2 + \mathbf{9}\,l_2{}^2 \le 1000 \\ 0.21\,x_1 - 0.3\,x_2 - 0.4\,c_1 - 0.6\,c_2 + 0.5\,l_1{}^2 + 0.4\,l_2{}^2 \le 0 \\ -0.03\,x_1 + 0.01\,x_2 + 0.1\,c_1 + 0.08\,c_2 + 0.08\,l_1{}^2 + 0.06\,l_2{}^2 \le 0 \end{cases}$$

$$x_1 \ge 75\,;\ x_2 \ge 75\,;\ c_1 \ge 0\,;\ c_2 \ge 0\,;\ l_1{}^2 \ge 1\,;\ l_2{}^2 \ge 1$$

$$x_1 \le 400\,;\ x_2 \le 400\,;\ c_1 \le 50\,;\ c_2 \le 50\,;\ l_1{}^2 \le 5\,;\ l_2{}^2 \le 5$$

$$c_1, c_2, l_1, \text{and } l_2 \text{ are integers}$$

Results of the 40 iterations:

Index	x_1	x_2	c_1	c_2	l_1	l_2	Cost function value
1	192.114239078050	75.3697752350742	0	46	3	3	145.467069434982
2	190.718551557885	75.2348097654916	4	44	1	4	146.026894256308
3	190.001035777302	75.0007231493859	0	48	1	4	145.800961567638
4	192.694579930752	76.7046457492677	5	40	3	3	146.392555153566
5	188.289388005516	89.7593296780216	19	20	1	4	155.820213111874
6	245.632915442610	75.5027833195899	36	31	2	1	155.942379620414
7	213.496554115507	76.1602170099778	14	40	3	2	149.993161543632
8	209.198793402784	75.0476237372037	9	46	2	3	149.152499384318
9	212.509493531991	75.0950564507414	16	37	2	3	149.438398865265
10	214.325003082725	75.1391218851331	9	46	3	2	149.772710621437
11	213.289595094485	75.3191896121995	10	45	3	2	149.574149179325
12	191.711083390437	75.0100024903806	5	42	1	4	145.872327258474
13	190.049050484080	75.1115979921097	0	48	1	4	145.915153338123

14	230.072604795666	75.0018665922902	23	41	2	2	152.973461371761
15	232.823480969564	75.0030722865449	29	34	2	2	153.299809348760
16	219.253994125876	75.0511766069507	16	45	1	3	151.522257184018
17	210.379081529905	79.7552518289476	12	39	2	3	152.793451105024
18	152.330436332321	99.6807323951188	1	21	1	5	157.761790055303
19	215.151537265033	75.0385918786039	8	46	3	2	149.780193870253
20	193.710843006341	75.6648925809025	0	50	1	4	147.911656224715
21	255.162504893068	75.3694927759945	38	35	1	1	158.281294966315
22	209.850852320809	81.5884754412519	27	24	2	3	153.534883593369
23	210.778294586393	78.3325447397687	24	28	2	3	151.232778641710
24	211.368227671644	75.1599389635033	6	50	3	2	149.254413368646
25	209.255439561799	75.0354938829895	9	46	2	3	149.158576363230
26	190.146157657398	75.8437071750836	3	45	1	4	146.453183754795
27	216.314196229337	75.1571516088045	7	46	3	2	150.085695316725
28	226.391199460592	75.2594066787396	23	37	3	1	151.700825849043
29	203.190314134839	75.6541533267514	0	50	4	1	147.745832234528
30	180.958146281749	78.6864445960078	6	34	2	4	146.005244020932
31	243.142968246699	75.2625233530765	30	39	2	1	155.679161491778
32	233.028209089387	75.0449511985904	26	36	2	2	153.348918805547
33	212.192722769249	75.1748770167512	10	43	2	3	149.715206145851
34	213.417099692909	75.2136350972148	10	45	3	2	149.517401495366
35	180.726164246859	75.0518446747632	1	41	2	4	143.314509481345
~~36~~	~~255.273437500000~~	~~270.507812500000~~	~~9~~	~~18~~	~~5~~	~~2~~	~~340.289062500000~~
37	210.676870287996	75.2659219339857	12	42	2	3	149.442390826986
38	192.051972454587	75.2700182899561	0	47	1	4	146.458608197336
39	192.527728923198	75.5318460743920	6	40	1	4	146.336980143912
40	191.531570187213	75.0131815191897	9	38	1	4	145.621334423435

Matlab displays that the 36th solution is unfeasible, then it is deleted.
The new statistical table is obtained with Excel:

Cost function results analysis:

Measures	Corresponding point						
Decision Variables	x_1	x_2	c_1	c_2	l_1	l_2	Cost function
Minimum of the cost function	180.72616	75.05184	1	41	2	4	143,31450
Maximum of the cost function	255.16250	75.36949	38	35	1	1	158,28129
Average of the cost function	215.15154	75.03859	8	46	3	2	149,8486752

Parameters' analysis:

Average (for each parameter)	**207,19136**	**76,73246**	**11,87179**	**40,38462**	**1,97436**	**2,84615**	**149,84868**
Standard Deviation (for each parameter)	**20,09147**	**4,61090**	**10,64049**	**7,61099**	**0,84253**	**1,08914**	**3,695899**

Consequently, if these changes were real, the best food balance of products to buy each day, would be **180.73 grams** of corn, **75.05 grams** of fodder, **1** Extra seed cube, **41** Super seed cubes, **2m²** Lawn type 1 and **4m²** lawns type 2 per portion. Doing this, the farmer would pay **143.31**£ per day for all the cattle. Introducing these both new products, the farmer would **gain** 155.29 – 143.31 = **11.98** £ **each day**.

Sensitivity 2:

From now on, let's suppose the amounts of proteins and fiber are change as follow:

Product	Quantity of *protein* (g)	Quantity of *fiber* (g)	Cost (£)	Calories (cal)
Lawn 4 grams type 1 (1 m^2)	**1.9**	**0.32**	0.0025	**8**
Lawn 4 grams type 2 (1 m^2)	**1.8**	**0.28**	0.0035	**9**

Each data is for 1 m^2 of lawn.

Consequently, the mathematical problem becomes:

$$\textbf{min} \quad Z = 0.30x_1 + 0.90\,x_2 + 0.15\,c_1 + 0.2\,c_2 + 0.5\,l_1{}^2 + 0.7\,l_2{}^2$$

$$\textit{s.t.} \begin{cases} -x_1 - x_2 - 2c_1 - 2\,c_2 - 4\,l_1{}^2 - 4\,l_2{}^2 \le -400 \\ -1.3\;x_1 - 1.4\,x_2 - 2\,c_1 - 2\,c_2 - 8\,l_1{}^2 - 9\,l_2{}^2 \le -600 \\ 1.3\,x_1 + 1.4\,x_2 + 2\,c_1 + 2\,c_2 + 8\,l_1{}^2 + 9\,l_2{}^2 \le 1000 \\ 0.21\,x_1 - 0.3\,x_2 - 0.4\,c_1 - 0.6\,c_2 + \mathbf{0.7}\,l_1{}^2 + \mathbf{0.6}\,l_2{}^2 \le 0 \\ -0.03\,x_1 + 0.01\,x_2 + 0.1\,c_1 + 0.08\,c_2 + \mathbf{0.12}\,l_1{}^2 + \mathbf{0.08}\,l_2{}^2 \le 0 \\ x_1 \ge 75\,;\; x_2 \ge 75\,;\; c_1 \ge 0\,;\; c_2 \ge 0\,;\; l_1{}^2 \ge 1\,;\; l_2{}^2 \ge 1 \end{cases}$$

$$x_1 \le 400\,;\; x_2 \le 400\,;\; c_1 \le 50\,;\; c_2 \le 50\,;\; l_1{}^2 \le 5\,;\; l_2{}^2 \le 5$$

$$c_1, c_2, l_1, \text{and } l_2 \text{ are integers}$$

Results of the 40 iterations:

Index	x_1	x_2	c_1	c_2	l_1	l_2	Cost function value
1	212.541544755884	76.1136359115264	7	46	2	3	150.814735747139
2	221.671965612965	75.3448443427800	11	48	1	3	152.361949592392
3	212.003350465638	75.3777170617930	8	45	2	3	149.940950495305
4	207.315641211590	87.0155848186745	11	37	2	3	157.858718700284
5	213.653737718272	75.2074505550530	2	50	2	3	150.382826815029
6	223.091142138208	75.0732619648447	21	37	1	3	151.843278409823
7	226.902143425653	92.1616391030460	36	18	2	2	164.816118220437
8	252.284040823178	75.0465810475220	30	45	1	1	157.927135189723
9	212.048643469459	75.3626121438985	7	46	2	3	149.990943970347
10	222.804317631280	75.3599859710663	22	36	1	3	151.965282663344
11	221.479492401572	75.0629381924489	16	43	1	3	151.800492093676
12	222.067037984887	75.9756886181973	16	42	1	3	152.598231151844
13	244.344647571492	75.1640795566974	33	34	1	2	156.001065872475
14	233.388553050156	75.0733414611324	15	47	2	2	154.032573230066
15	213.170149514917	75.0516572975982	4	49	2	3	150.197536422314
16	226.741996134646	77.3181283061959	8	50	3	1	154.008914315970
17	213.103315792310	75.0183502033203	5	48	2	3	150.097509920681
18	232.444357250974	77.3930700513513	26	35	2	2	155.087070221508
19	256.929917099467	75.0637322218407	49	23	1	1	157.786334129497
20	246.046157459449	75.4083138462084	35	32	2	1	156.031329699422
21	233.622500162062	75.0811020433499	24	38	2	2	153.659741887633

22	245.779418642287	75.1969299654790	23	45	2	1	156.561062561617
~~23~~	~~242.487380251241~~	~~79.1016222582840~~	~~41~~	~~24~~	~~1~~	~~2~~	~~158.187674107828~~
24	230.577624741951	75.9088662317742	14	49	2	2	154.191267031182
25	232.891707548928	76.7788946251299	26	35	2	2	154.668517427295
26	231.255582036198	75.3833971242412	14	49	2	2	153.921732022677
27	228.810986835094	75.4556515678648	11	47	3	1	152.803382461606
28	222.602192297147	75.0760097115721	16	43	1	3	152.149066429559
29	221.554322501226	75.0775378914872	13	46	1	3	151.986080852706
30	232.995278087930	75.0957980011395	22	40	2	2	153.584801627405
31	252.285598768191	75.0226872845321	30	45	1	1	157.906098186536
32	229.050943252977	75.5946797365166	8	50	3	1	153.150494738758
33	241.815302070860	75.4868766248836	30	38	1	2	155.882779583653
34	212.133314405625	75.1892670488715	4	49	2	3	150.010334665672
35	213.158044984953	75.1467598372726	3	50	2	3	150.329497349031
36	222.765702526948	75.3119314088714	18	40	1	3	152.110449026069
37	234.544912469875	75.0842195212096	16	45	2	2	154.139271310051
38	225.054887232927	75.0546676926751	11	46	1	3	152.715667093286
39	233.241048485984	75.0379236587165	24	38	2	2	153.506445838640
40	244.420536699454	75.2184984218633	27	41	2	1	155.972809589513

Matlab displays that the 36th solution is unfeasible, then it is deleted.
The new statistical table is obtained with Excel:

Cost function results analysis:

Measures	Corresponding point						
Decision Variables	x_1	x_2	c_1	c_2	l_1	l_2	**Cost function**
Minimum of the cost function	212.00335	75.37771	8	45	2	3	**149,94095**
Maximum of the cost function	226.90214	92.16164	36	18	2	2	**164,81612**
Average of the cost function	233.62250	75.08110	24	38	2	2	**153,71263**

Parameters' analysis:

Average (for each parameter)	**228,27159**	**76,14857**	**17,84615**	**42,17949**	**1,71795**	**2,23077**	**153,71263**
Standard Deviation (for each parameter)	**12,74941**	**3,27165**	**10,80823**	**7,30130**	**0,60475**	**0,80986**	**3,01755**

Consequently, the best food balance of products to buy each day, is **212.00 grams** of corn, **75.38 grams** of fodder, **8** Extra seed cubes, **45** Super seed cubes, **2m^2** Lawn type 1 and **3m^2** lawns type 2 per portion. Doing this, the farmer will pay **149.94**£ per day for all the cattle.
Introducing these both new products, the farmer **gains** 155.29 – 149.94 = **5.35** £ **each day**.

5 Conclusion

The study of this diet problem highlights the sensitivity of an optimization problem. Indeed, the sensitivity analysis reveals that if you would change slightly some sensitive parameters, the optimal solution could vary a lot. Therefore, the farmer has to be sure of the precision of the data (calories, proteins, …), because the smallest error could change a lot the total price to minimize and distort his plans.

The optimal solution stays the one after introducing the cubes and lawns. As a matter of fact, in this problem, the introduction of new products leads to a more optimum solution, providing a lower price. However, in real-life, the problem must be furnished with more parameters, not only proteins and fibers, and take in consideration as many products as possible available on the market (not only six). Furthermore, some farmers don’t have only cows to feed but other livestock. Consequently, it would be necessary to include the portion for other animals too.

To conclude, with accurate data, linear programming is helpful to support development of dietary guidelines that fulfil all nutritional requirements.