Advanced Optimization and Decision Analyitcs: Assignment 1

Author: Simon Schmetz

The Document contains the solutions to the first Assignment of the Advanced Optimization Class at the Universidad Carlos III de Madrid in the Master of Statistics for Datascience. The first part contains the solutions to the two problems, manually found using matlab, the second part contains the solution to the same problems, but solved via Gurobi in Python.

Part 1: Solving manually Using Matlab

The following section contains the solutions solved using Matlab.

Problem 1

We try to optimize the following function defined $\mathbf{x} \in \mathbb{R}^2$,

$f(\mathbf{x}) \triangleq 3x_1 + 5x_2 - 3x_3^2$

s.t

$g_1(\mathbf{x}) \triangleq 2x_1^2 - 37x_2 + 9x_3 = 18$

$g_1(\mathbf{x}) \triangleq 5x_1 + x_2 + 5x_3^2 = 24$

We begin by defining the functions using the Symbolic Math Toolbox

```
clear; clc;

% Define Variables
syms x1 x2 x3 lambda1 lambda2 real

x = [x1 x2 x3];
lambda = [lambda1 lambda2];
xlambda = [x1 x2 x3 lambda1 lambda2];

% Define functions
syms f(x1, x2, x3)
syms g1(x1, x2, x3)
syms g2(x1, x2, x3)
syms L(x1, x2, x3, lambda1, lambda2)

%% objective
f(x1, x2, x3) = 3*x1 + 5*x2 -3*x3^2; % define f

%% left side of constraints
g1(x1, x2, x3) = 2*x1^2 - 37*x2 +  9*x3;
g2(x1, x2, x3) = 5*x1 + x2 +  5*x3^2;

% right side of constraints
b1 = 18;
```

```
b2 = 24;
b = [b1, b2];

% define Lagrange Function:
L(x1, x2, x3, lambda1, lambda2) = f - lambda1*(g1-b1) - lambda2*(g2-b2)
```

L(x1, x2, x3, lambda1, lambda2) = $3\,x_1 + 5\,x_2 + \lambda_1\left(-2\,{x_1}^2 + 37\,x_2 - 9\,x_3 + 18\right) - \lambda_2\left(5\,{x_3}^2 + 5\,x_1 + x_2 - 24\right) - 3\,{x_3}^2$

a) Analysis of the regularity of the feasible points

We than check that the the gradient of the constraint 1 can not be expressed as a linear combination of the the gradient of constraint two, to verify that the problem is well defined, that the constraints are indipendent.

```
gradg1 = gradient(g1)
```

gradg1(x1, x2, x3) =

$$\begin{pmatrix} 4\,x_1 \\ -37 \\ 9 \end{pmatrix}$$

```
gradg2 = gradient(g2)
```

gradg2(x1, x2, x3) =

$$\begin{pmatrix} 5 \\ 1 \\ 10\,x_3 \end{pmatrix}$$

```
syms c1
xc = [x c1]
```

xc = $\begin{pmatrix} x_1 & x_2 & x_3 & c_1 \end{pmatrix}$

```
%assume(xc, 'real')
xnr = solve(gradg2 == c1 * gradg1, g1 == b(1), g2 == b(2), xc)
```

```
xnr = struct with fields:
    x1: [0×1 sym]
    x2: [0×1 sym]
    x3: [0×1 sym]
    c1: [0×1 sym]
```

As no solution exists, the constraints are independent and the feasible points are regular.

a) First order Necessary Conditions for local optimum

Define Gradients

We begin bei defining the gradient of the Lagrangian function L with respect to x

```
gradxL = simplify(gradient(L, x))
```

gradxL(x1, x2, x3, lambda1, lambda2) =

$$\begin{pmatrix} 3-4\,\lambda_1\,x_1-5\,\lambda_2 \\ 37\,\lambda_1-\lambda_2+5 \\ -9\,\lambda_1-6\,x_3-10\,\lambda_2\,x_3 \end{pmatrix}$$

The Gradient with respect to Lambda

```
gradlambdaL = simplify(gradient(L, lambda))
```

gradlambdaL(x1, x2, x3, lambda1, lambda2) =

$$\begin{pmatrix} -2\,{x_1}^2+37\,x_2-9\,x_3+18 \\ -5\,{x_3}^2-5\,x_1-x_2+24 \end{pmatrix}$$

And the gradient of L with respect to both x and lambda.

```
gradL = simplify(gradient(L, xlambda))
```

gradL(x1, x2, x3, lambda1, lambda2) =

$$\begin{pmatrix} 3-4\,\lambda_1\,x_1-5\,\lambda_2 \\ 37\,\lambda_1-\lambda_2+5 \\ -9\,\lambda_1-6\,x_3-10\,\lambda_2\,x_3 \\ -2\,{x_1}^2+37\,x_2-9\,x_3+18 \\ -5\,{x_3}^2-5\,x_1-x_2+24 \end{pmatrix}$$

Identify Optima

With the Gradients defined, we solve for the points that fulfill the first oder NC of the gradient of L and yield a total of 4 candidate points.

```
stat_points = solve(gradL == 0, xlambda);
pe = [stat_points.x1 stat_points.x2 stat_points.x3 stat_points.lambda1
stat_points.lambda2];
pe = double(pe)
```

```
pe = 4×5
   -9.6690    5.4567    3.6576   -0.1504   -0.5630
  -10.1460    4.1641   -3.7568   -0.1523   -0.6365
    4.6563    0.7002    0.0607   -0.1080    1.0025
  -97.1627  509.8128   -0.0101    0.1080    8.9970
```

We then store the x and lambda values into seperate vectors.

```
xA = double([stat_points.x1(1) stat_points.x2(1) stat_points.x3(1)]);
lambdaA = double([stat_points.lambda1(1) stat_points.lambda2(1)]);

xB = double([stat_points.x1(2) stat_points.x2(2) stat_points.x3(2)]);
```

```
lambdaB = double([stat_points.lambda1(2) stat_points.lambda2(2)]);

xC = double([stat_points.x1(3) stat_points.x2(3) stat_points.x3(3)]);
lambdaC = double([stat_points.lambda1(3) stat_points.lambda2(3)]);

xD = double([stat_points.x1(4) stat_points.x2(4) stat_points.x3(4)]);
lambdaD = double([stat_points.lambda1(4) stat_points.lambda2(4)]);
```

c) Second order conditions for local optimum

We then want to identify wether the canidate points are maxima, minnima or saddle points. We beginn by calculating the Hessian of the Lagrangian. We find the Hessian to be indipendent of x values, meaning that if the Hessian for a given canidate point fulfills the second order conditions for the given lambdas, it does so for all values of x, e.g. globally.

```
HessxL = hessian(L, x)
```

```
HessxL(x1, x2, x3, lambda1, lambda2) =
```

$$\begin{pmatrix} -4\,\lambda_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -10\,\lambda_2 - 6 \end{pmatrix}$$

```
syms d1 d2 d3 real
d = [d1 d2 d3].';
gradg1 = gradient(g1, x);
```

We thus calculate the Hessian for all Canidate Points.

Point A

For point A, we find the Hessian to be indefinite and thus a potential canidate for a local minimum.

```
HA = double(subs(HessxL, lambda, lambdaA));
double(eig(HA))
```

```
ans = 3×1
   -0.3700
         0
    0.6014
```

We continue by evaluating the tangential gradients

```
assume(d,'clear');
subs(gradg1, x, xA).' * d
```

```
ans(x1, x2, x3) =
```

$$9\,d_3 - 37\,d_2 - \frac{2721593429150539\,d_1}{70368744177664}$$

```
subs(gradg2, x, xA).' * d
```

```
ans(x1, x2, x3) =
```

$$5\,d_1 + d_2 + \frac{41180391572731175\,d_3}{1125899906842624}$$

where we find the term gradients to always be positive except for d = 0, indicating a local minimum with NC-1

```
assume(9*d3 - 37*d2 - (sym("2721593429150539")*d1)/sym("70368744177664") ==
0 &  5*d1 + d2 + (sym("41180391572731175")*d3)/sym("1125899906842624") ==0)
simplify(d.' * HA * d)
```

```
ans =
```

$$\frac{1581670238076957565936625444852850693611100480865843\,d_3^2}{30558320183165709706359994090657607176452145414144}$$

Solving the system of equations shows that the zero vector is the solution, concluding the SC2 Condition for a local minimum

```
solve(d3==0,d)
```

```
ans = struct with fields:
    d1: 0
    d2: 0
    d3: 0
```

Point B

We continue with point B, which we find to be positive semi-definite

```
HB = double(subs(HessxL, lambda, lambdaB));
double(eig(HB))
```

```
ans = 3×1
         0
    0.3650
    0.6094
```

We continue by evaluating the tangential gradients

```
assume(d,'clear');
subs(gradg1, x, xB).' * d
```

```
ans(x1, x2, x3) =
```

$$9\,d_3 - 37\,d_2 - \frac{713961626381871\,d_1}{17592186044416}$$

```
subs(gradg2, x, xB).' * d
```

```
ans(x1, x2, x3) =
```

$$5\,d_1 + d_2 - \frac{10574315830673315\,d_3}{281474976710656}$$

where we find the term gradients to always be positive except for d = 0, indicating a global minimum with NC-1

```
assume(9*d3 - 37*d2 - (sym("713961626381871")*d1)/sym("17592186044416") ==
0 &  5*d1 + d2 - (sym("10574315830673315")*d3)/sym("281474976710656") ==0)
simplify(d.' * HB * d)
```

```
ans =
```

$$\frac{8347532532619763996303782656276357703098931430296 77\, d_3^2}{14883321343858284730736981737144195845442651553792}$$

Again, solving the system of equations shows that the zero vector is the solution, and thus together with the initial observation of the Hessians independence of x, fullfilling the SC-2 of a global minimum

```
solve(d3==0,d)
```

```
ans = struct with fields:
    d1: 0
    d2: 0
    d3: 0
```

Point C

Like before, we evaluate the Hessian via the Eigenvalues and find it to be indefinite for Point C.

```
HC = double(subs(HessxL, lambda, lambdaC));
double(eig(HC))
```

```
ans = 3×1
  -16.0246
         0
    0.4322
```

```
assume(d,'clear');
subs(gradg1, x, xC).' * d
```

```
ans(x1, x2, x3) =
```

$$\frac{5242499709344185\, d_1}{281474976710656} - 37\, d_2 + 9\, d_3$$

```
subs(gradg2, x, xC).' * d
```

```
ans(x1, x2, x3) =
```

$$5\, d_1 + d_2 + \frac{43724728865280405\, d_3}{72057594037927936}$$

In the tangent gradient analysis, we find the gradients to always be negative, indicating a NC1 Local maximum.

```
assume((sym("5242499709344185")*d1)/sym("281474976710656") - 37*d2 + 9*d3
== 0 &  5*d1 + d2 + (sym("43724728865280405")*d3)/sym("72057594037927936")
==0)
simplify(d.' * HC * d)
```

```
ans =
```

$$-\frac{3450455160490933651913005521481885811470234017074871 83\, d_3^2}{21546136066498305225299114678351732534529874219499520}$$

With the only solution to the system the zero vector, the point thus fulfills the SC-2 condition for a Local Maximum.

```
solve(d3==0,d)
```

```
ans = struct with fields:
    d1: 0
    d2: 0
    d3: 0
```

Point D

For point D, we find the Hessian to be Negative Semidefinite, indicating a potential global maximum.

```
HD = double(subs(HessxL, lambda, lambdaD));
double(eig(HD))
```

```
ans = 3×1
  -95.9704
   -0.4321
         0
```

```
assume(d,'clear');
subs(gradg1, x, xD).' * d
```

```
ans(x1, x2, x3) =
```

$$9\,d_3 - 37\,d_2 - \frac{6837214120977123\,d_1}{17592186044416}$$

```
subs(gradg2, x, xD).' * d
```

```
ans(x1, x2, x3) =
```

$$5\,d_1 + d_2 - \frac{29199930463551635\,d_3}{288230376151711744}$$

This is verified by the tangent gradient analysis with both the NC-1 of the following term always being negative fulfilled.

```
assume(9*d3 - 37*d2 - (sym("6837214120977123")*d1)/sym("17592186044416") ==
0 &  5*d1 + d2 - (sym("29199930463551635")*d3)/sym("288230376151711744")
==0)
simplify(d.' * HD * d)
```

```
ans =
```

$$-\frac{29783754178546394447174157737635255729296570587557729878 55\,d_3^2}{31034215142363974018918116680222741123184445232972300288}$$

As well as the only solution to the system the zero vector. Point D thus indeed is a global maximum.

```
solve(d3==0,d)
```

```
ans = struct with fields:
```

```
d1: 0
d2: 0
d3: 0
```

We can also get the function values for the corresponding x coordinates as a quick sanity check to find that Point B indeed has the minimum function value out of all points, while Point D has the overall largest value.

```
fA = double(subs(f, x, xA));
fB = double(subs(f, x, xB));
fC = double(subs(f, x, xC));
fD = double(subs(f, x, xD));

fvals = [fA; fB; fC; fD]
```

```
fvals = 4×1
10^3 ×
   -0.0419
   -0.0520
    0.0175
    2.2576
```

Problem 2

We try to optimize the following function defined $\mathbf{x} \in \mathbb{R}^2$,

$$f(\mathbf{x}) \triangleq x_1^2 + x_2^2 - x_3^2$$

s.t

$$g_1(\mathbf{x}) \triangleq 8x_1^2 + 24x_2 - 15x_3 \leq 129$$

$$g_1(\mathbf{x}) \triangleq x_1^2 + 2x_2^2 + 4x_3^2 \geq 15$$

with $g_1(\mathbf{x})$ being reformulated to put the problem into standard form as

$$g_1(\mathbf{x}) \triangleq -x_1^2 - 2x_2^2 - 4x_3^2 \leq -15$$

We then begin by defining the functions using the Symbolic Math Toolbox

```
clear; clc; % clear all variables

% Define Variables
syms x1 x2 x3 mu1 mu2 real

x = [x1 x2 x3];
mu = [mu1 mu2];

xmu = [x mu];
```

```
% Define functions
syms f(x1, x2, x3)
syms h1(x1, x2, x3)
syms h2(x1, x2, x3)
syms L(x1, x2, x3, mu1, mu2)

%% objective
f(x1, x2, x3) = x1^2 + x2^2 - x3^2; % define f

%% left side of constraints
h1(x1, x2, x3) = 8*x1^2 + 24*x2 - 15*x3;
h2(x1, x2, x3) = -x1^2 + -2*x2^2 -4*x3^2;

% right side of constraints
b1 = 129;
b2 = -15;
b = [b1, b2];

% define Lagrange Function:
L(x1, x2, x3, mu1, mu2) = f - mu1*(h1-b1) - mu2*(h2-b2)
```

L(x1, x2, x3, mu1, mu2) = $\mu_2 \left(x_1^2 + 2\,x_2^2 + 4\,x_3^2 - 15\right) - \mu_1 \left(8\,x_1^2 + 24\,x_2 - 15\,x_3 - 129\right) + x_1^2 + x_2^2 - x_3^2$

a) Find candidate Points for a local maxima

As per assignment definition we can assume all points to be regular, we proceed to finding canidate points for Local Maxima under NC criterion. We start by calculating the Gradient of the Lagrangian for x and the Hessian

```
gradxL = simplify(gradient(L, x))
```

gradxL(x1, x2, x3, mu1, mu2) =

$$\begin{pmatrix} 2\,x_1\,(\mu_2 - 8\,\mu_1 + 1) \\ 2\,x_2 - 24\,\mu_1 + 4\,\mu_2\,x_2 \\ 15\,\mu_1 - 2\,x_3 + 8\,\mu_2\,x_3 \end{pmatrix}$$

We then set the ktt criterion by assuming both must to be non-negative and solve with complementary slackness. We get a total of 8 candidate points out of this approach.

```
assume(mu1 >= 0 & mu2 >= 0)
kkt = solve(gradxL == 0, mu1*(h1-b1) == 0, mu2*(h2-b2) == 0, xmu);
```

The candidate points are stored seperately like in Problem 1.

```
double([kkt.x1, kkt.x2, kkt.x3, kkt.mu1, kkt.mu2].')
```

```
ans = 5×8
   -3.5424    3.5424         0         0         0         0   -3.6583    3.6583
    0.9200    0.9200         0    8.8205         0         0    1.5000    1.5000
   -0.4356   -0.4356         0    5.5128   -1.9365    1.9365    0.9375    0.9375
    0.3383    0.3383         0    0.7350         0         0    0.1250    0.1250
    1.7061    1.7061         0         0    0.2500    0.2500         0         0
```

```
xA = double([kkt.x1(1), kkt.x2(1), kkt.x3(1)]);
muA = double([kkt.mu1(1), kkt.mu2(1)]);

xB = double([kkt.x1(2), kkt.x2(2), kkt.x3(2)]);
muB = double([kkt.mu1(2), kkt.mu2(2)]);

xC = double([kkt.x1(3), kkt.x2(3), kkt.x3(3)]);
muC = double([kkt.mu1(3), kkt.mu2(3)]);

xD = double([kkt.x1(4), kkt.x2(4), kkt.x3(4)]);
muD = double([kkt.mu1(4), kkt.mu2(4)]);

xE = double([kkt.x1(5), kkt.x2(5), kkt.x3(5)]);
muE = double([kkt.mu1(5), kkt.mu2(5)]);

xF = double([kkt.x1(6), kkt.x2(6), kkt.x3(6)]);
muF = double([kkt.mu1(6), kkt.mu2(6)]);

xG = double([kkt.x1(7), kkt.x2(7), kkt.x3(7)]);
muG = double([kkt.mu1(7), kkt.mu2(7)]);

xH = double([kkt.x1(8), kkt.x2(8), kkt.x3(8)]);
muH = double([kkt.mu1(8), kkt.mu2(8)]);
```

We then have to check the feasibility of the candidate points by introducing the x values of the candidate points into the constraints and check if they are fullfilled. Assuming very small values equal to zero, we yield only Point C unfeasable with a non-negative value (e.g. not fullfilling h2)

```
double(subs([h1 - b1, h2 - b2], x, xA))
```

```
ans = 1×2
10^-15 ×
    0.5331    0.0042
```

```
double(subs([h1 - b1, h2 - b2], x, xB))
```

```
ans = 1×2
10^-15 ×
    0.5331    0.0042
```

```
double(subs([h1 - b1, h2 - b2], x, xC))
```

```
ans = 1×2
  -129    15
```

```
double(subs([h1 - b1, h2 - b2], x, xD))
```

```
ans = 1×2
         0 -262.1677
```

```
double(subs([h1 - b1, h2 - b2], x, xE))
```

```
ans = 1×2
  -99.9526         0
```

```
double(subs([h1 - b1, h2 - b2], x, xF))
```

```
ans = 1×2
 -158.0474         0
```

```
double(subs([h1 - b1, h2 - b2], x, xG))
```

```
ans = 1×2
         0   -6.3984
```

```
double(subs([h1 - b1, h2 - b2], x, xH))
```

```
ans = 1×2
         0   -6.3984
```

b) Evaluate 2nd order Conditions to Evaluate for local or global maximum

With all points except Point C identified as feasible, we continue with evaluating which candidate points correspond to a local or global maximum. First, we set up the Hessian and notice its indipendence of x, which will give us later insights into the globality of maxima.

```
H = hessian(L, x)
```

```
H(x1, x2, x3, mu1, mu2) =
```

$$\begin{pmatrix} 2\mu_2 - 16\mu_1 + 2 & 0 & 0 \\ 0 & 4\mu_2 + 2 & 0 \\ 0 & 0 & 8\mu_2 - 2 \end{pmatrix}$$

We further define the gradients of the Constraints.

```
gradh1 = simplify(gradient(h1, x))
```

```
gradh1(x1, x2, x3) =
```

$$\begin{pmatrix} 16x_1 \\ 24 \\ -15 \end{pmatrix}$$

```
gradh2 = simplify(gradient(h2, x))
```

```
gradh2(x1, x2, x3) =
```

$$\begin{pmatrix} -2x_1 \\ -4x_2 \\ -8x_3 \end{pmatrix}$$

```
% variables for tangent direction analysis
syms d1 d2 d3 real
d = [d1 d2 d3].';
```

Point A

The Hessian of point A is positive semi-definite implying a potential mininmum. As we are only interested in Maximum, we do not pursue the candidate further.

```
HA = subs(H, mu, muA);
double(eig(HA))
```

```
ans = 3×1
         0
    8.8242
   11.6485
```

Point B

Like for Point A, the Hessian of point B is positive semi-definite implying a potential mininmum. As we are only interested in Maximum, we again do not pursue the candidate further.

```
HB = subs(H, mu, muB);
double(eig(HB))
```

```
ans = 3×1
         0
    8.8242
   11.6485
```

Point C

As point C is not feasible (shown in part a above), we do not investigate.

Point D

The Hessian of Point D is indefinite, giving a potential local optimum.

```
HD = subs(H, mu, muD);
double(eig(HD))
```

```
ans = 3×1
   -9.7607
   -2.0000
    2.0000
```

We thus apply a tangent direction analysis

```
assume(d,'clear');
subs(gradh1, x, xE).' * d
```

```
ans(x1, x2, x3) = 24 d_2 − 15 d_3
```

```
subs(gradh2, x, xE).' * d
```

```
ans(x1, x2, x3) = 4 √15 d_3
```

and find the gradient to be negative in all directions, implying a local maximum.

```
assume(24*d2 - 15*d3 == 0 &  4*sqrt(sym(15))*d3 ==0)
simplify(d.' * HD * d)
```

```
ans(x1, x2, x3, mu1, mu2) =
```

$$-\frac{1142\,d_1^2}{117}$$

With the only solution to the system the zero vector, the point thus fulfills the SC-2 condition for a Local Maximum.

```
solve(d3==0,d)
```

```
ans = struct with fields:
    d1: 0
    d2: 0
    d3: 0
```

Point E

The Hessian of candidate point E is positive semi-definite implies a potential mininmum. As we are only interested in maxima, we again do not pursue the candidate further.

```
HE = subs(H, mu, muE);
double(eig(HE))
```

```
ans = 3×1
         0
    2.5000
    3.0000
```

Point F

The Hessian of candidate point F is positive semi-definite implies a potential mininmum. As we are only interested in maxima, we again do not pursue the candidate further.

```
HF = subs(H, mu, muF);
double(eig(HF))
```

```
ans = 3×1
         0
    2.5000
    3.0000
```

Point G

```
HG = subs(H, mu, muG);
double(eig(HG))
```

```
ans = 3×1
    -2
     0
     2
```

We thus apply a tangent direction analysis

```
assume(d,'clear');
subs(gradh1, x, xG).' * d
```

ans(x1, x2, x3) = $24\,d_2 - 15\,d_3 - \sqrt{2}\ \sqrt{1713}\ d_1$

```
subs(gradh2, x, xG).' * d
```

ans(x1, x2, x3) =

$$\frac{\sqrt{2}\ \sqrt{1713}\ d_1}{8} - \frac{15\,d_3}{2} - 6\,d_2$$

and find the gradient to be positive in all directions, implying a local minimum. As we are only searching for maxima, candidate point G can thus be discarded.

```
assume(24*d2 - 15*d3 - sqrt(sym(2))*sqrt(sym(1713))*d1 == 0 &
(sqrt(sym(2))*sqrt(sym(1713))*d1)/8 - (15*d3)/2 - 6*d2 ==0)
simplify(d.' * HG * d)
```

ans(x1, x2, x3, mu1, mu2) =

$$\frac{561\,{d_3}^2}{32}$$

Point H

```
HH = subs(H, mu, muH);
double(eig(HH))
```

ans = 3×1

```
    -2
     0
     2
```

We thus apply a tangent direction analysis

```
assume(d,'clear');
subs(gradh1, x, xH).' * d
```

ans(x1, x2, x3) = $24\,d_2 - 15\,d_3 + \sqrt{2}\ \sqrt{1713}\ d_1$

```
subs(gradh2, x, xH).' * d
```

ans(x1, x2, x3) =

$$-6\,d_2 - \frac{15\,d_3}{2} - \frac{\sqrt{2}\ \sqrt{1713}\ d_1}{8}$$

and find the gradient to be positive in all directions, implying a local minimum. As we are only searching for maxima, candidate point H like candidate point G previously, can thus be discarded.

```
assume(24*d2 - 15*d3 + sqrt(sym(2))*sqrt(sym(1713))*d1 == 0 &  - 6*d2 -
(15*d3)/2 - (sqrt(sym(2))*sqrt(sym(1713))*d1)/8 ==0)
simplify(d.' * HH * d)
```

```
ans(x1, x2, x3, mu1, mu2) =
```

$$\frac{561\, d_3^{\,2}}{32}$$

Conclusion

From all 8 candidate points, we thus conclude only Point D to fullfill the NC2 and SC2 Local Maxima Criteria. Point D's function value can be computed as shown below and can be later used to be compared with the gurbi python result.

```
fD = double(subs(f, x, xD));
```

```
fvals = 8×1
   13.2050
   13.2050
         0
   47.4103
   -3.7500
   -3.7500
   14.7539
   14.7539
```

Advanced Optimization and Decision Analytics: Assignment 1 - Solving via Gurobi in Python

In the following section, Problems 1 and 2 are solved using Gurobi in Python.

Problem 1

In problem 1, we are serching for optima of the objective function

$$f(\mathbf{x}) = x_1^2 + x_2^2 - x_3^2,$$

s.t.

$$8x_1^2 + 24x_2 - 15x_3 \leq 129$$

$$x_1^2 + 2x_2^2 + 4x_3^2 \geq 15.$$

We set up the Variables and Constraints of the Problem first.

In [10]:
```
from gurobipy import Model, GRB

model1 = Model('prob1')

# Define the variables
x1 = model1.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x1")
x2 = model1.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x2")
x3 = model1.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x3")

#add constraints
model1.addConstr(2 * x1 ** 2 - 37 * x2 + 9 * x3 == 18, "c1")
model1.addConstr(5 * x1 + x2 + 5 * x3 ** 2 == 24, "c2")
```

Out[10]:
```
<gurobi.QConstr Not Yet Added>
```

Then we first define the objective function with the goal to maximize, run the solver and print the results. Gurobi, looking for the global optimum finds the same point as found in the matlab analysis as candidate point D.

In [11]:
```
# define objective for maximization
model1.setObjective(3*x1 + 5*x2 - 3*x3**2, GRB.MAXIMIZE)

# solve
model1.setParam('OutputFlag', 0)
model1.optimize()

# print results
```

```
for v in model1.getVars():
    print(f"{v.varName}: {v.x}")
print(f"Optimal objective value: {model1.objVal}")
```

```
x1: -97.16265952910427
x2: 509.8127734171107
x3: -0.010244122590746783
Optimal objective value: 2257.5755736720976
```

Likewise, the global minimum is found corresponding to candidate Point B from the Matlab analysis.

In [12]:
```
# define objective for maximization
model1.setObjective(3*x1 + 5*x2 - 3*x3**2, GRB.MINIMIZE)

# solve
model1.optimize()

# print results
for v in model1.getVars():
    print(f"{v.varName}: {v.x}")
print(f"Optimal objective value: {model1.objVal}")
```

```
x1: -10.14600494474171
x2: 4.164109941389492
x3: -3.756751649559144
Optimal objective value: -51.95701399667372
```

Problem 2

In problem 2, our aim is to maximize the following function.

$$f(\mathbf{x}) = x_1^2 + x_2^2 - x_3^2,$$

s.t.

$$8x_1^2 + 24x_2 - 15x_3 \leq 129$$

$$x_1^2 + 2x_2^2 + 4x_3^2 \geq 15.$$

where as in the previous section, we reformulate the second constraint as

$$-x_1^2 - 2x_2^2 - 4x_3^2 \leq -15$$

to make sure that the optimization is in standard form for maximization. We then define the Variables, Objective function and Constraints in Gurobi and run the solver. When looking at the results the solver outputs, we find the variables and function value to be extremely large and the solver to be giving a warning to the Problem being unbounded.

In [ ]:
```
from gurobipy import Model, GRB

model2 = Model('prob2')
```

```python
# Define the variables
x1 = model2.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x1")
x2 = model2.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x2")
x3 = model2.addVar(lb = -GRB.INFINITY, ub= GRB.INFINITY, name="x3")

# define objective
model2.setObjective(x1**2 + x2**2 - x3**2, GRB.MAXIMIZE)

#add constraints
model2.addConstr(8 * x1**2 + 24 * x2 - 15 * x3 <= 129, "c2")
model2.addConstr(- x1**2 - 2*x2**2 - 4*x3**2 <= -15, "c1")

# solve
model2.optimize()

# print results
for v in model2.getVars():
    print(f"{v.varName}: {v.x}")
print(f"Optimal objective value: {model2.objVal}")
```

```
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (mac64[x86] - Darwi
n 24.3.0 24D81)

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
Thread count: 6 physical cores, 12 logical processors, using up to 1
2 threads

Optimize a model with 0 rows, 3 columns and 0 nonzeros
Model fingerprint: 0x2382531e
Model has 3 quadratic objective terms
Model has 2 quadratic constraints
Coefficient statistics:

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
Thread count: 6 physical cores, 12 logical processors, using up to 1
2 threads

Optimize a model with 0 rows, 3 columns and 0 nonzeros
Model fingerprint: 0x2382531e
Model has 3 quadratic objective terms
Model has 2 quadratic constraints
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  QMatrix range    [1e+00, 8e+00]
  QLMatrix range   [2e+01, 2e+01]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
  QRHS range       [2e+01, 1e+02]

Continuous model is non-convex -- solving as a MIP

Presolve time: 0.00s
Presolved: 10 rows, 8 columns, 24 nonzeros
```

```
Presolved model has 3 bilinear constraint(s)
Warning: Model contains variables with very large bounds participati
ng
         in product terms.
         Presolve was not able to compute smaller bounds for these v
ariables.
         Consider bounding these variables or reformulating the mode
l.

Variable types: 8 continuous, 0 integer (0 binary)
Found heuristic solution: objective 47.4102564

Root relaxation: unbounded, 7 iterations, 0.00 seconds (0.00 work un
its)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/N
ode Time

     0     0  postponed    0        47.41026          -      -     -
0s
     0     0  postponed    0        47.41026          -      -     -
0s
     0     2  postponed    0        47.41026          -      -     -
0s
*    3     6               2    625000.00000          -      -   4.7
0s
*    8     8               3    750000.00000          -      -   3.1
0s
*    9     8               3    1125000.0000          -      -   2.8
0s
*   13    12               4    1250000.0000          -      -   2.8
0s
*   19    12               5    2250000.0000          -      -   1.9
0s
*   21    12               6    4250000.0000          -      -   1.7
0s
*   26    12               6    8125000.0000          -      -   1.4
0s
*   40    14               7    8250000.0000          -      -   0.9
0s
*   42    14               8    1.625000e+07          -      -   0.9
0s
*   44    14               9    3.225000e+07          -      -   0.8
0s
*   49    14               9    6.412500e+07          -      -   0.8
0s
*   67    26              10    6.425000e+07          -      -   0.6
0s
*   69    26              11    1.282500e+08          -      -   0.5
0s
*   74    26              11    2.561250e+08          -      -   0.5
0s
*  106    40              12    2.562500e+08          -      -   0.4
0s
```

```
*  108    40              13    5.122500e+08          -      -   0.4
0s
*  110    40              14    1.024250e+09          -      -   0.4
0s
*  112    40              15    2.048250e+09          -      -   0.4
0s
*  114    40              16    4.096250e+09          -      -   0.4
0s
*  149    40              19    4.098000e+09          -      -   0.4
0s
*  165    40              19    8.193000e+09          -      -   0.3
0s
*  193    38              24    8.224000e+09          -      -   0.3
0s
*  195    38              25    1.641600e+10          -      -   0.3
0s
*  230    38              22    3.277000e+10          -      -   0.3
0s
*  232    38              23    6.553800e+10          -      -   0.3
0s
*  234    38              24    1.310740e+11          -      -   0.3
0s
*  236    38              25    2.621460e+11          -      -   0.3
0s
*  278    44              29    2.621760e+11          -      -   0.2
0s
*  280    44              30    5.243200e+11          -      -   0.2
0s
*  282    44              31    1.048608e+12          -      -   0.2
0s
*  284    44              32    2.097184e+12          -      -   0.2
0s
*  359    44              33    2.097216e+12          -      -   0.3
0s
*  361    44              33    4.194336e+12          -      -   0.3
0s
*  363    44              34    8.388640e+12          -      -   0.3
0s
*  365    44              35    1.677725e+13          -      -   0.2
0s
*  367    44              36    3.355446e+13          -      -   0.2
0s
*  369    44              37    6.710890e+13          -      -   0.2
0s
H  371    44                    4.551109e+14          -      -   0.2
0s
*  547    94              44    5.113881e+14          -      -   0.2
0s
*  592    94              42    5.368710e+14          -      -   0.2
0s
*  749    66              45    5.368719e+14          -      -   0.3
0s
H 1017    46                    6.710886e+19          -      -   0.2
0s

Explored 2249 nodes (235 simplex iterations) in 0.16 seconds (0.01 w
```

```
ork units)
Thread count was 12 (of 12 available processors)

Solution count 10: 6.71089e+19 5.36872e+14 5.36871e+14 ... 2.56125e+
08

Optimal solution found (tolerance 1.00e-04)
Best objective 6.710886188749e+19, best bound 6.710886188749e+19, ga
p 0.0000%
x1: -32000.0
x2: -8191999871.0
x3: 0.0
Optimal objective value: 6.710886188748802e+19
```

As Gurboi searches only for a single global optima, we thus due to the unbounded nature of the problem and unlike in matlab where multiple candiate points for local as well as global optima are identified. However, in case of the problem being unbounded and the objective function going to infinity in certain directions like in the objective function of Problem 2, the approach taken in MATLAB is unable to find the maximum value as the conditions used to identify optima do not work as they are limited to fininding saddle points or maxima/minima with gradients equal to zero. Significantly different results in both approaches are thus not unexpected.

The Optimization Problem should thus be reconsidered as in its current unbounded form, the optimal x values to maximize would be $x_1 = 0$, $x_2 = -\infty$, $x_3 = 0$ yielding $f(x) = \infty$. Overall this is a good example on why black-box solvers need to be used with caution and a good understanding of their limitations.