Workshop Optimization Methods

Jorge Eduardo Durán Vásquez*
Métodos de optimización, Facultad de Minas
Universidad Nacional de Colombia – Sede Medellín

Abstract – In this document the lector can find the application of different optimization techniques to solve some specials mathematics problems, also we show 2 demonstrations and some graphs to show the converge of the solutions, finally, we put the Matlab code that was used to find the optimal in each case.

Kay words – Constraints, Convexity, Linear Programming, Nonlinear optimization, Matlab, Quadratic programming, Unimodal.

I. INTRODUCTION

In this paper we going to use many optimization methods to solve a different kind of objective functions, with some of this kind of elements, for example with linear/nonlinear equality or inequality constraints, of some parameters in the cost function. Depending how is defined the objective functions cause a big difference in the election of the method that we need to use to solve it, so that is an important thing to consider in each of the exercises, also we going to bring some explanation of the methods and then we are going to use use the Matlab to solve it. In another hand, the reader can find 2 demonstrations, one of them is about the Jacobian and another one is about the subgradients of the functions.

Here we try to do things in the best possible manner as recommended Tom van den Boom and Bart de Shutter the writers of the book that we take as a reference to solve some of these problems.

Theoretical framework

Important concepts that we would have in mind are the definition of the convex function, sugradient, and the Kuhn Tucker conditions, so here we define 2 of them, the subgradient definition the reader could finding it on the point 9.

Convex function:

A function f is a convex if

(a) The domain dom(f) is a convex set, that means that for each pair x,y ϵ C and for all λ ϵ [0,1] the next property holds:

$$(1-\lambda)x + \lambda y \in C$$

(b) The next inequality holds for all x,y ϵ dom(f) and 0 ≤ λ ≤ 1:

$$f\big((1-\lambda)x + \lambda y\big) \le (1-\lambda)f(x) + \lambda f(y)$$

Kuhn-Tucker conditions:

For an inequality/equality constrained optimization problem necessary conditions for a minimum or maximum of the function f in x, satisfying h(x)=0 and g(x)≤0 are given by Kuhn-Tucker conditions. Those say that exist vector λ and u such that:

$$\nabla f(x) + \nabla g(x)u + \nabla h(x)\lambda = 0$$

$$u^T g(x) = 0$$

$$u \ge 0$$

$$h(x) = 0$$

$$g(x) \le 0$$

For convex optimization problems, the Kuhn-Tucker conditions give necessary and sufficient conditions for a minimum or maximum[1].

Optimization Techniques

To solve the following exercises we will use some of these optimization techniques:

- Linear programming (the simplex method)
- Quadratic programming
- Nonlinear optimization without constraints
- Constraints in nonlinear optimization
- Convex optimization techniques

II. DEVELOPMENT

In this section of the article, we going to make a detailed description of the solution of the problems raised in the workshop.

Linear Programming

1. Part I: Resolve in a graphic manner the following problem: (key, graph the cost function as level curves and the constraints as lines in 2D)

$$\begin{aligned} &\min\ f(x) = x - 2y && (1)\\ &x + y \geq 2 && (2)\\ &-x + y \geq 1 && (3)\\ &y \leq 3 && (4)\\ &y, x \geq 0 && (5) \end{aligned}$$

We can see the graph of the cost function f in figure 1.

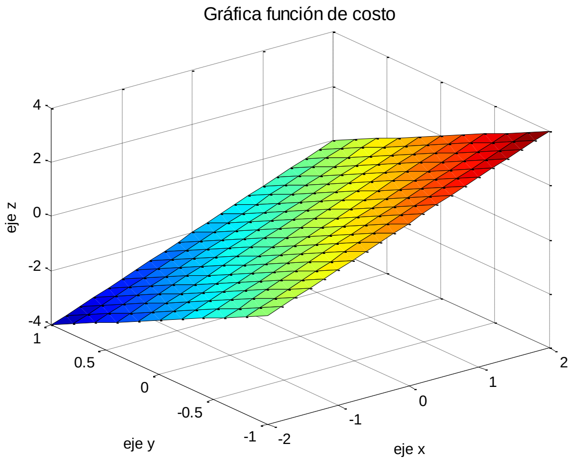

Figure 1. Cost function exercise 1

Now we are going to produce the level curves graph:

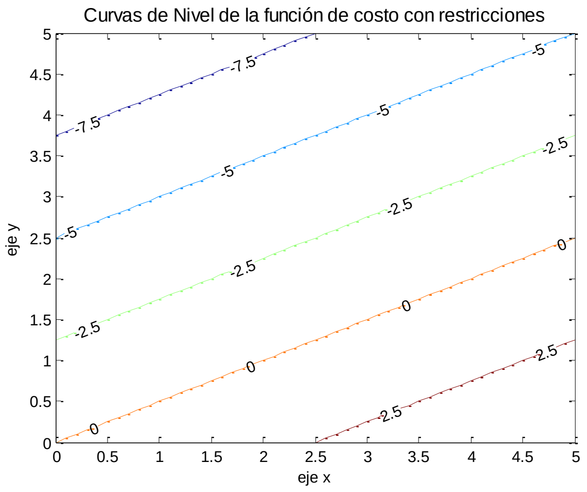

Figure 2. The level curves graph of the cost function

The constraints of the problem are 4 inequalities, we can rewrite two of them, and those are the follows:

a) We can rewrite the inequality equation 2 as:

$$y \geq 2 - x \quad (6)$$

b) Rewriting the inequality 3, we obtain:

$$y \geq 1 + x \quad (7)$$

Now the graphs the cost function as level curves and the constraints as lines in 2D:

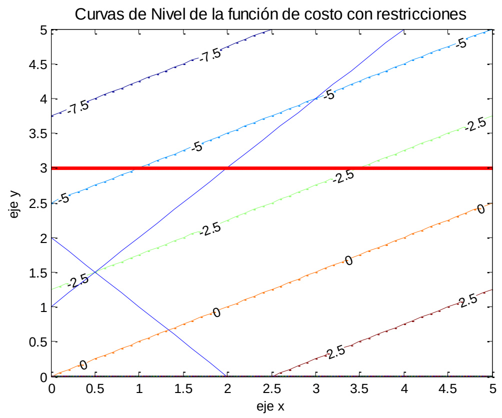

Figure 3. Cost function as level curves and the constraints (1)

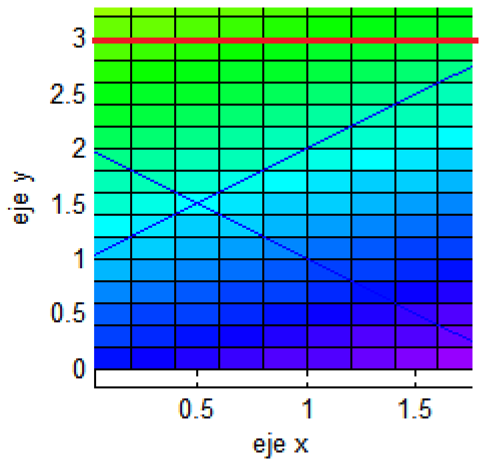

Figure 4. Cost function as level curves and the constraints (2)

Analyzing the graph of the function with the constraints we can see that the minimum point is the point where the restrictions 4 and 5 cross each other, so the minimum of the function is:

$$x = 0\,;\, y = 3;\ Z = -6$$

Part II: The simplex algorithm, showing the steps for each iteration (Matrices B and N).

The first step has reformulated the exercise as a linear programming problem in standard form. We define the objective function:

$$\min f(x) = (x_1 - 2x_2) \qquad (1)$$

And to the constraints 2 and 3 we multiplicity for -1 each side of the inequation

$$-(x+y) \leq -2 \qquad (2)$$
$$-(-x+y) \leq -1 \qquad (3)$$

The constraints are equivalent to:

$$-x_1 - x_2 + x_3 = -2 \qquad (3)$$
$$x_1 - x_2 + x_4 = -1 \qquad (4)$$
$$x_2 + x_5 = 3 \qquad (5)$$
$$y, x \geq 0 \qquad (6)$$

We have:

$$c = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, b = \begin{bmatrix} -2 \\ -1 \\ 3 \\ 0 \\ 0 \end{bmatrix}$$

$$A = \begin{bmatrix} -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We suppose the first choice of B and N, how B is a nonsingular (it has inverse matrix) *m x m* submatrix of A:

$$B = \begin{bmatrix} -1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The determinant of the matrix B is -1, for that reason we can say that it is a nonsingular matrix, now the remaining part of A is:

$$N = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Now we find x_B, x_N, c_B, and c_N as:

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, x_N = \begin{bmatrix} x_6 \\ x_7 \end{bmatrix}$$

$$x_B = \begin{bmatrix} 0 \\ 0 \\ -2 \\ -1 \\ 3 \end{bmatrix}, x_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, c_B = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, c_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The corresponding values of z_0 and p are:

$$z_0 = 0 \text{ and } p = [-2 \quad 1]$$

Since p^T is not greater than 0, the optimum is not found yet.

Selection of the column on N: How -2 is the largest negative component of p we select the first column of N.

Selection of the column of B: we have

$$y = B^{-1} * N_{.,1} = \begin{bmatrix} 0 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

How the only positive component of the vector y is Y_5, we select the 5 column of B.

Now we interchange the third column of B and the second column of N, which leads to:

$$B = \begin{bmatrix} -1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$N = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Now we have:

$$x_B = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_6 \end{bmatrix}, x_N = \begin{bmatrix} x_5 \\ x_7 \end{bmatrix}$$

$$x_B = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 2 \\ 3 \end{bmatrix}, x_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, c_B = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, c_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The corresponding values of z_0 and p are:

$$z_0 = -6 \text{ and } p = [2 \quad 1]$$

How all the components of p^T is greater than 0, the optimum is found, the optimal solution is (X1,X2)=(0 ; 3) and the corresponding profit is -6.

2. Dos estudiantes A y B trabajan en un laboratorio x y y horas semanales respectivamente. De acuerdo con las reglas, A puede trabajar como máximo 8 horas más que B y B puede trabajar como máximo 6 horas más A. Juntos pueden trabajar un máximo de 40 horas por semana. Encuentre el ingreso máximo combinado que pueden lograr si el estudiante A recibe 15.000 \$/hora y el estudiante B 17.000\$/hora.

$$\begin{array}{lr} \max\ f(x) = 15000\,x + 17000\,y & (1) \\ x - y \le 8 & (2) \\ y - x \le 6 & (3) \\ x + y \le 40 & (4) \\ y, x \ge 0 & (5) \end{array}$$

The first step has reformulated the exercise as a linear programming problem in standard form. We define the objective function as negative profit:

$$min\ f(x,y) = -(15000\,x + 17000\,y)$$

The constraints:

$$\begin{array}{llr} x - y \le 8 & \text{(Green)} & (2) \\ -x + y \le 6 & \text{(Red)} & (3) \\ x + y \le 40 & \text{(Blue)} & (4) \\ y, x \ge 0 & \text{(Black)} & (5) \end{array}$$

Of the last inequalities we obtain the following matrix:

$$A = \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$b = \begin{bmatrix} 8 \\ 6 \\ 40 \end{bmatrix}$$

And the coefficient vector is:

$$c = [-15000 \quad -17000]$$

The feasible solution of the problem is the yellow zone of the following picture:

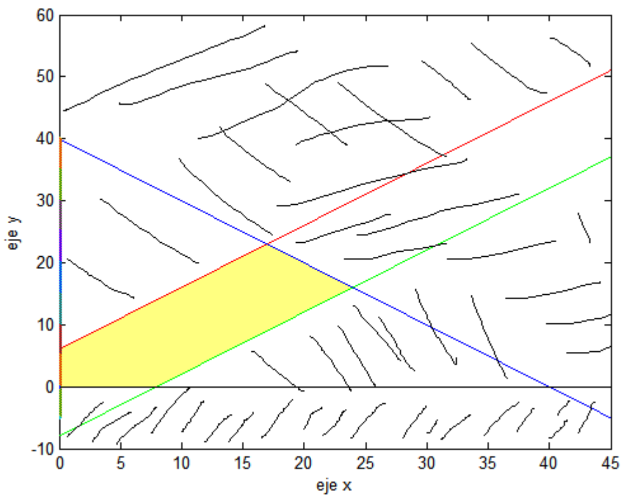

Figure 5. A feasible solution of the problem 2

After complete the matrix A with the slack variables we obtain:

$$A = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We divide the A matrix in B and N to have a “basic solution” of the problem:

$$B = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} N = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The first basic solution is:

$$x_B = B^{-1} * b = \begin{bmatrix} 0 & -0{,}5 & 0{,}5 \\ 0 & 0{,}5 & 0{,}5 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 8 \\ 6 \\ 40 \end{bmatrix} = \begin{bmatrix} 17 \\ 23 \\ 14 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x_N = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The corresponding “basic cost”:

$$z_0 = c_B^T * x_B = \begin{bmatrix} -15000 & -17000 & 0 \end{bmatrix} * \begin{bmatrix} 17 \\ 23 \\ 14 \end{bmatrix}$$
$$= -646.000$$

Now:

$$z = z_0 + p^T x_N$$

$$p^T = c_N^T - c_B^T * B^{-1} * N$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -15000 \\ -17000 \\ 0 \end{bmatrix} * \begin{bmatrix} 0 & -0{,}5 & 0{,}5 \\ 0 & 0{,}5 & 0{,}5 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$p^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -100 \\ -16000 \end{bmatrix} = \begin{bmatrix} 100 \\ 16000 \end{bmatrix}$$

How p > 0we obtain an optimal solution, that means that the values of x and y are:

$$(x, y) = (17\,, 23)$$

Using the sequence of MATLAB commands (linprog) that are in the book of the class we confirm that the solution is:

$$(x, y) = (17\,, 23)$$

And the corresponding profit is $ 646.000.

Quadratic Programming

3. Resolve the next optimization quadratic problem- type 2

The constraints of this problem are a linear equality constraint, defined by:

$$H = \begin{bmatrix} 1 & -4 & 2 & 1 \\ -4 & 16 & -8 & -4 \\ 2 & -8 & 4 & 2 \\ 1 & -4 & 2 & 1 \end{bmatrix}; c = \begin{bmatrix} -1 \\ 0 \\ 7 \\ 4 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}; b = 4$$

In the first iteration, we check the Kuhn-Tucker conditions with the λ, x and u zeros vectors.

$$Ax = b \quad (1)$$

$$H * x + A^T \lambda - \mathrm{u} = -\mathrm{c} \;\; (2)$$

$$x, u \geq 0 \;\; (3)$$

$$x^T u = 0 \quad (4)$$

$$\lambda = 0, \qquad \mathrm{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \mathrm{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The conditions 3 and 4 are satisfied because the zeros vectors satisfy the positive condition and the product, but it isn't a minimum of the function, now we use the quadprog Matlab function to solve this problem and then we check the Kuhn-Tucker conditions again.

$$\lambda = 0{,}8, \qquad \mathrm{x} = \begin{bmatrix} 3{,}24 \\ 0{,}76 \\ 0 \\ 0 \end{bmatrix} \text{ and } \mathrm{u} = \begin{bmatrix} 0 \\ 0 \\ 8{,}2 \\ 5 \end{bmatrix}$$

$$Ax = b \quad (1)$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 3{,}24 \\ 0{,}76 \\ 0 \\ 0 \end{bmatrix} = 4$$

$$H * x + A^T \lambda - \mathrm{u} = -\mathrm{c} \;\; (2)$$

$$\begin{bmatrix} 1 & -4 & 2 & 1 \\ -4 & 16 & -8 & -4 \\ 2 & -8 & 4 & 2 \\ 1 & -4 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 3{,}24 \\ 0{,}76 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} * 0{,}8 - \begin{bmatrix} 0 \\ 0 \\ 8{,}2 \\ 5 \end{bmatrix} = - \begin{bmatrix} -1 \\ 0 \\ 7 \\ 4 \end{bmatrix}$$

$$x, u \geq 0 \;\; (3)$$

$$x^T u = 0 \quad (4)$$

$$\begin{bmatrix} 3{,}24 & 0{,}76 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 8{,}2 \\ 5 \end{bmatrix} = 0$$

How we demonstrate all Kuhn-Tucker conditions are satisfied, so the x value is a minimum of the function because, for a convex optimization problem, the Kuhn-tucker conditions give necessary and sufficient conditions for a minimum.

4. Prove that the gradient of C^Tx is C and the Jacobian of (Ax-b)=A^T

At the beginning we prove the first sentence:

$$f(x) = C^T * x$$

Here we suppose that C and X are column vectors.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \frac{\partial f}{\partial x_4} \\ \vdots \end{bmatrix}$$

Now we apply the gradient to the function, how we are working with a linear function of X we can take out the constant (c^T) to the function and apply the partial derivative to variable X.

$$\nabla f(x) = C^T \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \end{bmatrix} = C$$

When we multiplicate the row vector (1 x n) with a column vector (n x 1) of ones, we obtain a column vector (n x 1) with the same entries as the row vector C^T.

Now we prove the second sentence:

Jacobean of (Ax-b) is A^T

$$h_1(x) = A_{11} * x_1 + A_{12} * x_2 + A_{13} * x_3 \dots - b_1$$

$$h_2(x) = A_{21} * x_1 + A_{22} * x_2 + A_{23} * x_3 \dots - b_2$$

$$h_n(x) = A_{n1} * x_1 + A_{n2} * x_2 + A_{n3} * x_3 \dots - b_n$$

$$\vdots$$

$$\nabla h(x) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial x_n} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix}$$

We apply the partial derivative of each h(n) functions, we have:

$$\nabla h(x) = \begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & & \ddots & A_{n2} \\ \vdots & & & \vdots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix} = A^T$$

5. Resolve the following optimization problem:

$$\min f(x) = -8x_1 - 16x_2 + x_1^2 + 4\,x_2^2$$

$$Sujeto\ a: (x_1 + x_2) \leq 5, x_1 \leq 3, x_1 \geq 0, x_2 \geq 0$$

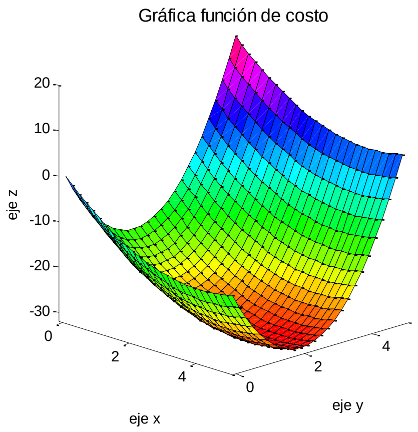

Figure 6. Cost function point 5

Now we put the function in the standard way, for that reason here calculate the Hessian matrix:

$$H = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$

$$c^T = [-8 \quad -16]$$

$$\min_x \frac{1}{2} x^T H x + c^T x$$

$$\min_x \frac{1}{2} x^T \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix} x + [-8 \quad -16] x$$

With the constraints:

$$Ax \leq b$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

In the next plot, we can see the 3D function and its level curves:

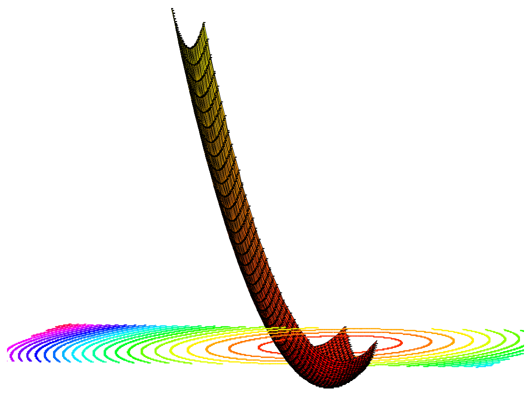

Figure 7. 3D function and its level curves

In the next plot, we put the linear constraints, and we indicate the solution point using the algorithm, here we use the Matlab function quadprog, and the parameters of the solution of this QP problem are (look the point 5 of the Matlab code in the annexed):

$$\lambda = \begin{bmatrix}0\\2\end{bmatrix}, \qquad x = \begin{bmatrix}3\\2\end{bmatrix} \text{ and } u = \begin{bmatrix}0\\0\end{bmatrix}$$

$$f(x_1, x_2) = -31$$

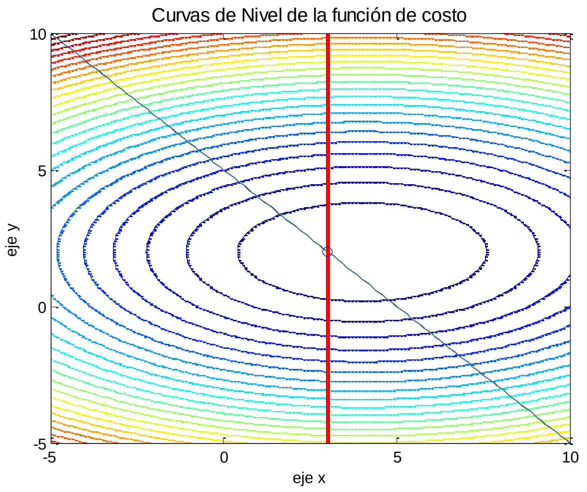

Figure 8. Level curves cost function point 5

Now we use the quadprog Matlab function to solve this problem and then we check the Kuhn-Tucker conditions

$$Ax = b \quad (1)$$

$$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix} * \begin{bmatrix}3\\2\end{bmatrix} = \begin{bmatrix}5\\3\end{bmatrix}$$

$$H * x + A^T\lambda - \mathrm{u} = -\mathrm{c} \ \ (2)$$

$$\begin{bmatrix}2 & 0\\0 & 8\end{bmatrix} * \begin{bmatrix}3\\2\end{bmatrix} + \begin{bmatrix}1 & 1\\1 & 0\end{bmatrix} * \begin{bmatrix}0\\2\end{bmatrix} - \begin{bmatrix}0\\0\end{bmatrix} = -\begin{bmatrix}-8\\-16\end{bmatrix}$$

$$\begin{bmatrix}6\\16\end{bmatrix} + \begin{bmatrix}2\\0\end{bmatrix} - \begin{bmatrix}0\\0\end{bmatrix} = -\begin{bmatrix}-8\\-16\end{bmatrix}$$

$$x, u \geq 0 \quad (3)$$

$$x^T u = 0 \quad (4)$$

$$[3 \quad 2] * \begin{bmatrix}0\\0\\8{,}2\\5\end{bmatrix} = 0$$

How we demonstrate all Kuhn-Tucker conditions are satisfied, so the x value is a minimum of the function because, for a convex optimization problem, the Kuhn-tucker conditions give necessary and sufficient conditions for a minimum.

Optimization without constraints

6. Do 3 iterations to find the minimum of

$$f(x_1, x_2) = (x_1 + 4)^2 + (x_1 + 4x_2)^2$$

Using the following methods (you have to write the Matlab code and plot the iterations):

a. Newton's algorithm

In the next graph we can see the level curves of the function and the solution found in each iteration (almost 6 iterations) of the algorithm, the first solution is on the (0,0) point and the solution is moving to the center of the first level curve indicating that the optimum value is near to that point.

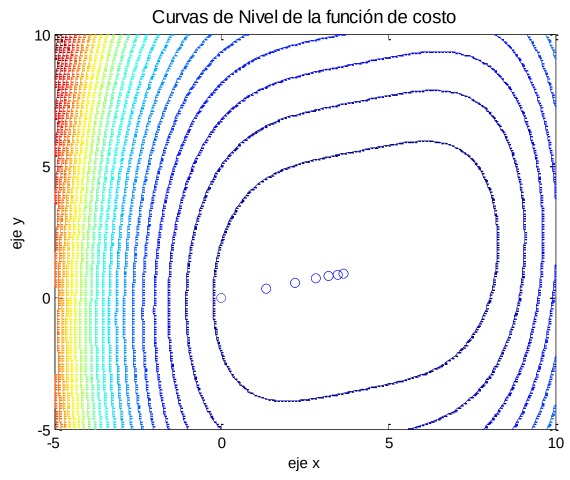

Figure 9. Level curves and Newton's algorithm solution

In the next table we computed all of the x values and the corresponding f value, there we can see the evolution of X$_1$ and X$_2$, and how the f value decrease:

Table 1 Newton algorithm solution

Iteración	X1	X2	f(x1,x2)
1	0,00	0,00	256,000
2	1,33	0,33	50,570
3	2,22	0,56	9,989
4	2,81	0,70	1,973
5	3,21	0,80	0,390
6	3,47	0,87	0,077
7	3,65	0,91	0,015

The Matlab code is in the annex of this paper.

b. Levenberg-Marquardt method

In the next graph we can see the level curves of the function and the solution found in each iteration (almost 6 iterations) of the algorithm, the first point solution is on the (0,0) and the solution is moving to the center of the first level curve indicating that the optimum value is near to that point.

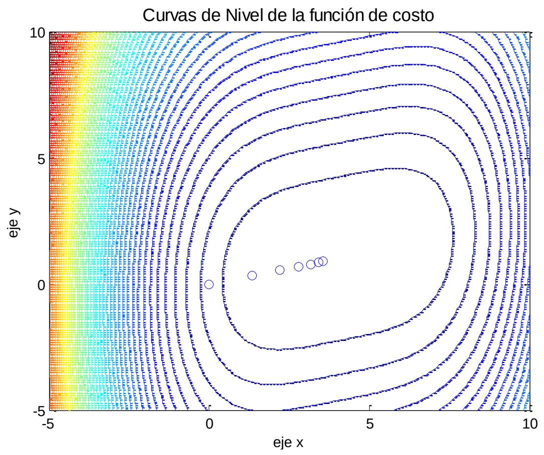

Figure 10. Level curves and Levenberg-Marquardt solutions

In the next table we computed all of the x values and the corresponding f value, there we can see the evolution of X_1 and X_2, and how the f value decrease, in comparison with Newton algorithm the evolutions of the X values here is slower, so we need more iterations to arrive at the solution point and it takes more time:

Table 2 Levenberg-Marquardt solutions

Iteración	X1	X2	f(x1,x2)
1	0	0	256,000
2	1,3253	0,3203	51,180
3	2,2049	0,5435	10,390
4	2,7857	0,6913	2,175
5	3,1653	0,788	0,486
6	3,4094	0,8502	0,122
7	3,5633	0,8895	0,036

The Matlab code is in the annex of this paper.

c. Broyden-Fletcher-Goldfarb-Shanno (BFGS)

In the next graph we can see the level curves of the function and the solution found in each iteration (almost 40 iterations) of the algorithm, the first point solution is on the (0,0) and the solution of the algorithm starts to follow one of the graphed level curves, then it is moving to the center of the first level curve indicating that the optimum value is near to that point. Here we need more iterations to have a good solution point, so the method takes more time to converge one good solution that arrives at the desired accuracy.

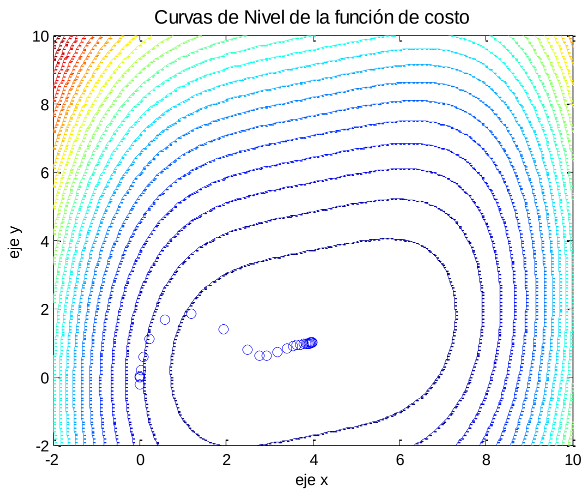

Figure 11 Level curves and BFGS solutions

In the next table we computed all of the x values and the corresponding function f value, there we can see the evolution of X_1 and X_2 in this algorithm, and also how the f value decrease, in comparison with the Newton algorithm and the Levenberg-Marquardt method, there is evidence that the evolutions of the X values here are slower, so we need more iterations to arrive to the solution point and it takes more time to stop:

Table 3 BFGS solutions

Iteración	X1	X2	f(x1,x2)
1	0	0	256
2	256	0	4032823552
3	0,001	0,0082	255,738949
4	0,002	-0,2375	256,384843
5	0,0066	-0,0312	254,321702
6	0,0223	0,182	250,831174
7	0,0832	0,5832	240,417426
8	0,2367	1,1023	217,978399
9	0,5867	1,6562	172,186784
10	1,1859	1,8542	101,542577
11	1,9296	1,395	31,7004255
12	2,4921	0,8058	5,70462575
13	2,7689	0,6201	2,38020164
14	2,9443	0,6172	1,46820412
15	3,1929	0,7046	0,56462954

d. The Davidson-Fletcher-Powell (DFP) method

In the next graph we can see the level curves of the function and the solution found in each iteration (almost 40 iterations) of the algorithm, the first point solution is on the (0,0) and the solution of the algorithm starts to follow one of the graphed level curves, but in all of 40 iterations the function value doesn't start to converge a one good solution with a good accuracy. Here we need more iterations to have a good solution point, so the method takes more time to converge to a good solution that arrives at the desired accuracy.

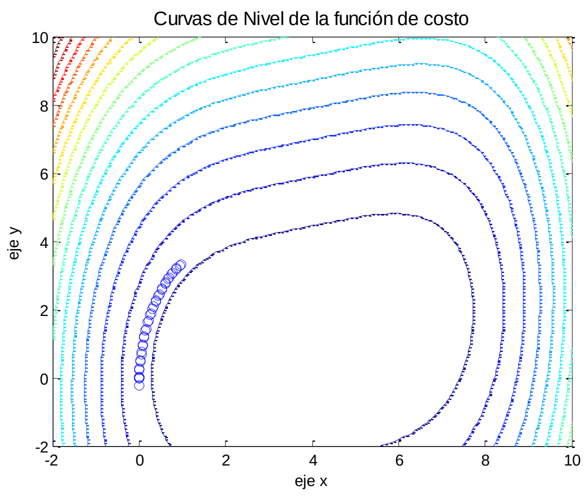

Figure 12 Level curves and DFP solution

In the next table we computed all of the x values and the corresponding function f value, there we can see the evolution of X_1 and X_2 in this algorithm, and also how the f value decrease, in comparison with the Newton algorithm, the Levenberg-Marquardt method or Broyden-Fletcher-Goldfarb-Shannon method is evident that the evolutions of the X values here are slower, so we need more iterations to arrive to the solution point and it takes more time to stop:

Table 4 DFP solution

Iteration	X1	X2	f(x1,x2)
1	0	0	256
2	256	0	4032823552
3	0,001	0,0082	255,738949
4	0,002	-0,2375	256,384843
5	0,0022	-0,0324	255,450562
6	0,0033	0,0077	255,163978
7	0,0097	0,213	254,235092
8	0,012	0,2532	253,949284
9	0,0246	0,4576	253,025745
10	0,0281	0,4976	252,740674
11	0,0467	0,7	251,822705
12	0,0514	0,7396	251,538287

Comparison:

These methods to solve unconstrained nonlinear optimization problems are so useful because allow solve that kind of problems so fast, here exist one big difference between the Newton (Newton algorithm and the Levenberg-Marquardt method) and Quasi-Newton methods (Broyden-Fletcher-Goldfarb-Shannon method and The Davidson-Fletcher-Powell (DFP) method) it is that the first need the evaluation of the Hessian of the objective function, the others only require the evaluation of the objective function and the gradient.

In this solution we can see that the evaluation of the Hessian matrix isn't a problem, so the big difference between the methods are the number of the iterations that the algorithm needs to converge to one good solution near to the desired accuracy. Probably in some exercises that have a greater difficulty in the Hessian matrix the Quasi-Newton methods could be useful.

Optimization with constraints

7. Resolve the following optimization problem:

$$\min f(x_1,x_2,x_3) = x_1^2 + 8 * x_2^2 + 3 * x_1 x_3$$

Subject to:

$$x_1 - x_2 + x_3 = 1$$
$$x_1 + x_2 = 2$$

Resolve for the elimination constraints method.

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 0 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The matrix A has 2x3 dimensions, here we need to eliminate the equality constraints so we define the parameter vector $\bar{x}$, an 1x3 matrix $\bar{A}$ and a vector x_0 such that $Ax_0=b$ and $A\bar{A}^T = 0$. For that we compute a singular value decomposition of A as follow:

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1{,}7321 & 0 \\ 0 & 1{,}4142 \end{bmatrix}$$

$$V^T = \begin{bmatrix} 0{,}5774 & 0{,}7071 & -0{,}4082 \\ -0{,}5774 & 0{,}7071 & 0{,}4082 \\ 0{,}5774 & 0 & 0{,}8165 \end{bmatrix}$$

For the matrix V, we search the submatrix V_1 and V_2 that satisfy $V_1^T V_2 = 0$, we use Matlab to do that and we find:

$$V_1 = \begin{bmatrix} 0{,}5774 & 0{,}7071 \\ -0{,}5774 & 0{,}7071 \\ 0{,}57740 & 0 \end{bmatrix}$$

$$\bar{A}^T = V_2^T = [-0{,}4082 \quad 0{,}4082 \quad 0{,}8165]$$

Now we find a vector X_0 that solve the equality constraints as $Ax_0=b$:

$$x_0 = V_1\Sigma^{-1}V_1^T V_2 = \begin{bmatrix} 1{,}333 \\ 0{,}667 \\ 0{,}333 \end{bmatrix}$$

So the new x value is:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1{,}333 \\ 0{,}667 \\ 0{,}333 \end{bmatrix} + [-0{,}4082 \quad 0{,}4082 \quad 0{,}8165] * \bar{x}$$

The new unconstrained optimization problem is:

$$\min f(\bar{x}) = (1{,}333 - 0{,}4082\bar{x})^2 + 8 * (0{,}667 + 0{,}408\bar{x})^2 + 3 * (1{,}333 - 0{,}4082\bar{x}) * (0{,}33 + 0{,}8165\bar{x})$$

To solving this unconstrained optimization problem we are going derive and equalize to zero, in that way we find the $\bar{x}$ minimum for the problem.

Table 5 Solution in the nonlinear optimization

Solution	$\bar{x}$	$f(\bar{x})$
1	-6,1	-12,083

Using the value of the $\bar{x}$ we find the x_1, x_2, x_3

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1{,}333 \\ 0{,}667 \\ 0{,}333 \end{bmatrix} + [-0{,}4082 \quad 0{,}4082 \quad 0{,}8165] * \bar{x}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3{,}833 \\ -1{,}8336 \\ -4{,}647 \end{bmatrix}$$

The Matlab code is annexed to this delivery.

8. Using Matlab, Python or the tool that you prefer. Resolve using sequential quadratic programming (SQP)(In Matlab function fmincon with the option: SQP)

Beginning in the point $x_0=[0,0]^T$, show the evolution of:

1. Search direction d_k (Step 2 of the algorithm of the book)

Table 6 Search direction

Search dir		
Iteration	Com X1	Com X2
0	0	0
1	4,77	2,75
2	0,3648	-0,5669
3	-0,0713	-0,0889
4	0,0141	0,0471
5	0,0002	0,0007
6	0	0

In the next plot we show how the vectors indicate the next search direction, all the vectors are pointing to the position of the next iteration solution that the algorithm finds. Note that we make a zoom on the axis to see better the behavior of the search direction.

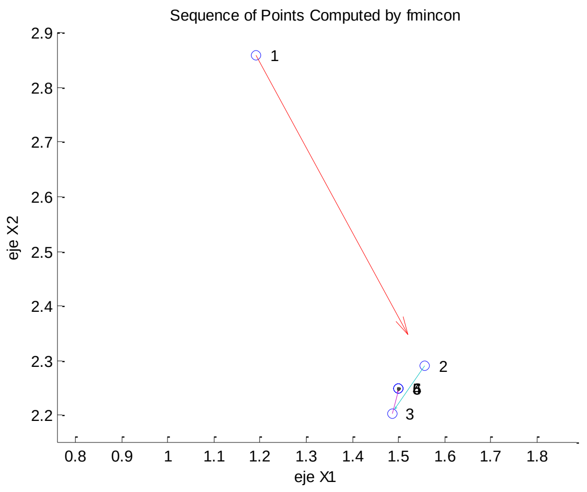

Figure 13 Search direction behavior

2. Step Length s_k (Step 3 of the algorithm of the book)

Table 7 Step length

Iter	Step length	Norm of step
0		
1	4,9	3,097
2	1	0,6741
3	1	0,114
4	1	0,0492
5	1	0,0007
6	1	0

3. Variable of the optimization $(x_k)_1,(x_k)_2$ in the iteration k

Table 8 X values

X Value		
Iter	X1	X2
0	0	0
1	1,19223124	2,85812916
2	1,55700484	2,29120431
3	1,48571154	2,20225604
4	1,49984864	2,24934609
5	1,50000002	2,25000003
6	1,50000001	2,25000003

In the next picture we can see the solutions points of the algorithm, and in the next table, we can see the f(x1,x2) value.

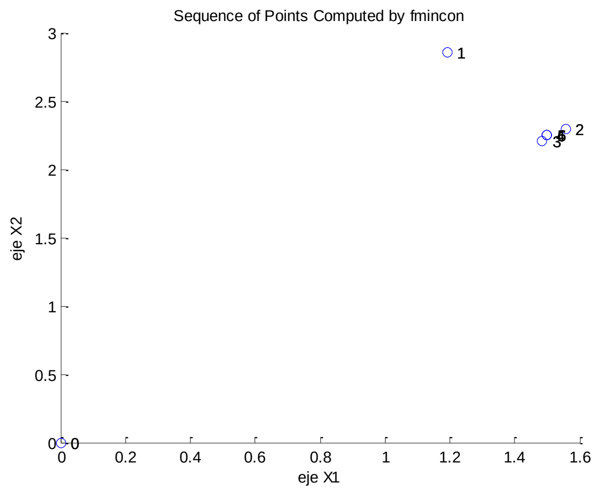

Figure 14 Sequence of points computed by fmincon

Table 9. F values

f (x1,x2)	
Iter	F value
0	9,0625
1	1,85526039
2	0,56504223
3	0,62504436
4	0,62490053
5	0,62499999
6	0,625

Convex Optimization

9. Show that the functions g (x) are sub-gradients of the corresponding functions f (x):

Definition of sub-gradient: The ∇f is called a subgradient of f if:

$$f(x) \geq f(y) + \left(\nabla f(y)\right)^T (x-y) \; for\ all\ x, y \in dom(f)$$

Now, keeping in mind the sub-gradient definition, we suppose linearity of the functions f(x) and g(x), also the operator ∇ as a gradient of the functions $f_1(x)$ and $f_2(x)$.

- The first point:

$$f(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x), \qquad x \in R^n,$$
$$f_1(x)\ and\ f_2(x)\ are\ derivable\ and\ convex.$$

$$g(x) = \alpha_1 \nabla f_1(x) + \alpha_2 \nabla f_2(x)$$

Here we demonstrate that the g(x) is a sub-gradient of function f (x):

$$g(x) = \alpha_1 \nabla f_1(x) + \alpha_2 \nabla f_2(x)$$

$$g(x) = \nabla[\alpha_1 f_1(x)] + \nabla[\alpha_2 f_2(x)]$$

$$g(x) = \nabla[\alpha_1 f_1(x) + \alpha_2 f_2(x)]$$

And, on another hand:

$$\nabla f(x) = \nabla[\alpha_1 f_1(x) + \alpha_2 f_2(x)]$$

$$\therefore \nabla f(x) = g(x)$$

- The second point:

$f(x) = f_1(Ax+b),$
$x \in R^n, A \in R^{mxn}, b \in R^m \;\; f_1(x)\ is\ derivable\ and\ convex.$

$$g(x) = A^T \nabla[f_1(Ax+b)]$$

To demonstrate this chain rule, we can:

$$\nabla f(x) = \nabla f_1(Ax + b)$$

$$u = Ax + b \text{ and } f(u)$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} * \frac{\partial u}{\partial x}$$

How we demonstrate in the point 4 the partial derivative of u is A^T and the partial derivative of "f" with respect to u is the gradient of "f", so:

$$\nabla f(x) = A^T \nabla f_1(u)$$

$$\nabla f(x) = A^T \nabla f_1(Ax + b)$$

For that reason:

$$\therefore \nabla f(x) = g(x)$$

10. Do two iterations of the ellipsoid algorithm to solve:

$$\min f(x_1, x_2) = 4(x_1 - 10)^2 + (x_2 - 4)^2$$

Subject to:

$$x_1 - x_2 \leq 10$$
$$x_1 - x_2 \geq 3$$
$$x_1 \geq 0$$

Draw the viable region and the algorithm steps taken from $x_0=[0,0]^T$

In the annex point_8, the reader can see the algorithm to resolve this problem, how the code was made to make the iterations automatically, we increase the number of iteration to see the solution into the feasible region. In the next table we can see how the x1 and x2 values are variating to arrive one optimal solution; in the final of the table, we can see a good solution of the problem but also the algorithm find one optimal solution in the 13 iterations.

Table 10 X solutions and f value Ellipsoid algorithm

Iteration	X1	X2	f(x1,x2)
1	0,000	0,000	416,000
2	4,083	-4,083	205,395
3	9,360	-0,289	20,036
4	11,197	4,472	5,955
5	7,132	5,387	34,818
6	9,022	1,397	10,606
7	11,134	4,027	5,143
8	8,855	5,038	6,325
9	10,228	2,883	1,457
10	10,235	6,360	5,791
11	9,435	3,825	1,306
12	10,816	4,173	2,692
13	9,889	4,046	0,052
14	10,482	2,931	2,074
15	10,436	5,249	2,320
16	10,055	3,505	0,257
17	9,994	4,797	0,635
18	9,879	3,898	0,069

The figure X.X is the graphic of the level curves of the problem; also we plot both of the constraints, the blue point indicate the points x1 and x2 that the algorithm find in the process, there we can see that blue points are in the center of the feasible zone, that indicates that the last ellipsoid contain the last point found.

$$x_1 - x_2 \leq 10 \text{ (Red line)}$$
$$x_1 - x_2 \geq 3 \text{ (Blue line)}$$

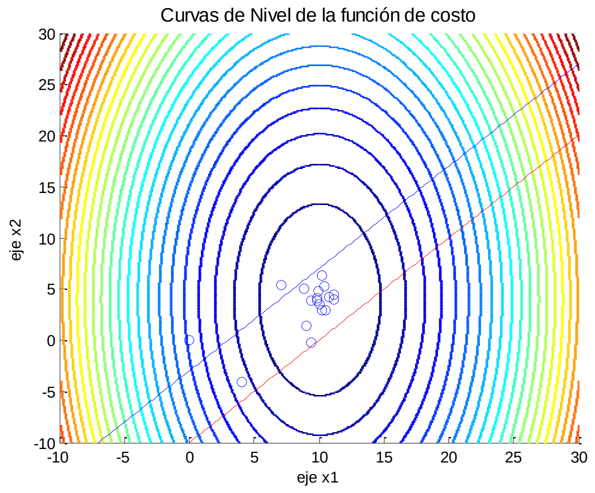

Figure 15 Level curves of the cost function

The Matlab code is annexed to this delivery.

11. Use the inner point algorithm to solve:

$$\min f(x_1, x_2) = -x_1 x_2$$

Subject to:

$$1 - x_1^2 - x_2^2 \geq 0$$

Graph the viable region and the algorithm steps starting from [0.1,0.1] and from [-0.1,-0.1].

The next 3D graph represents the cost function of this problem.

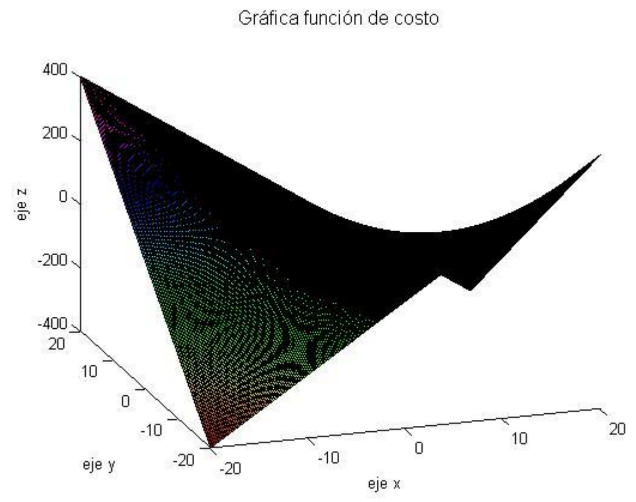

Figure 16 3D Cost function

After using the inner point algorithm, we can find the optimal solution starting in two different points, now we are going to show both process solutions; in a both of the cases, we use 51 iterations to find the solution, 10 iterations for the Newton algorithm for each of the 5 t value.

Starting from (-0.1,-0.1)

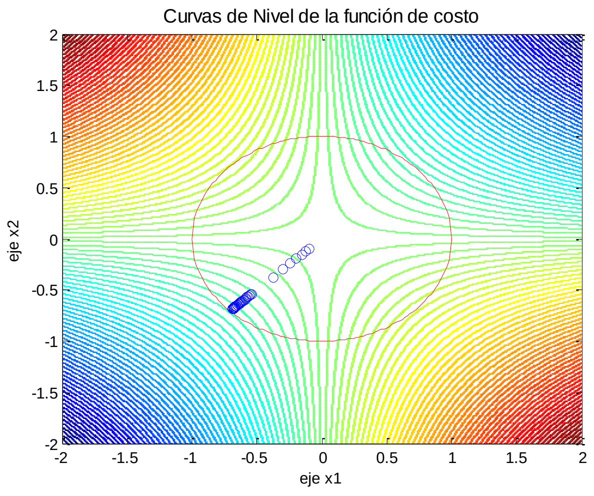

Figure 17 Level curves of cost function (-0.1,- 0.1)

Both of the solutions arrive at the same level curve value, that is -0,47, and the x values are showing on the tables.

Table 11 Solutions x values

Iteration	X1	X2	f(x1,x2)
1	-0,100	-0,100	-0,010
2	-0,125	-0,125	-0,016
3	-0,156	-0,156	-0,024
4	-0,194	-0,194	-0,038
5	-0,241	-0,241	-0,058
46	-0,683	-0,683	-0,466
47	-0,684	-0,684	-0,467
48	-0,684	-0,684	-0,468
49	-0,685	-0,685	-0,469
50	-0,685	-0,685	-0,470
51	-0,686	-0,686	-0,471

Starting from (0.1, 0.1)

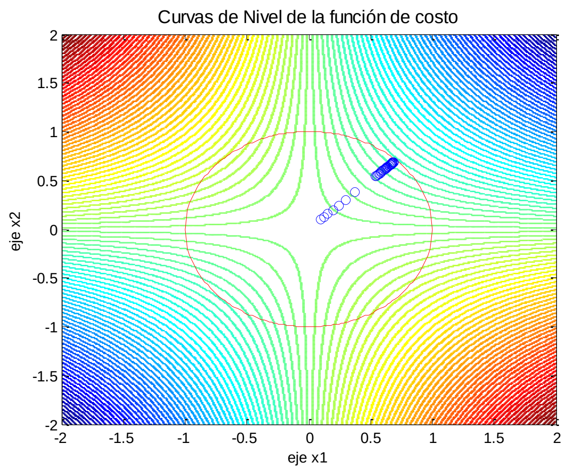

Figure 18 Level curves of cost function (0.1, 0.1)

Table 12 Optimal solution values

Iteration	X1	X2	f(x1,x2)
1	0,100	0,100	-0,010
2	0,125	0,125	-0,016
3	0,156	0,156	-0,024
4	0,194	0,194	-0,038
5	0,241	0,241	-0,058
46	0,683	0,683	-0,466
47	0,684	0,684	-0,467
48	0,684	0,684	-0,468
49	0,685	0,685	-0,469
50	0,685	0,685	-0,470
51	0,686	0,686	-0,471

III. CONCLUSIONS

- The computational tools are very useful to find a minimum or a maximum of one cost function whit or without linear/nonlinear equality or inequality constraints, almost all of the algorithms have to make repetitive iterations, and many repetitive calculations to find the optimum solution of one problem, so Matlab is a very good tool design and solved these kind of problems.
- The nonlinear optimization without constraints methods is very useful to solve many kinds of problems, because in many cases (as we see in this workshop) we try to eliminate the constraints and put them on the cost function to find the answer using a Newton or Quasi-Newton methods.
- How we can see in the development of this workshop exist many forms to solve the cost functions, is a very important task of the engineering or the designer of the optimizations problem take in mind the type of constraints, parameters and cost functions that are on the problem, so in many cases we try to do things in the best possible manner to find a good solution that arrives at the desired accuracy.
- Exist many approximations of the cost function or of the constraints to make less complex the solutions of the exercises, but how we saw in the nonlinear optimization without constraints, probably have less complexity could increase the number of iterations that the algorithm has to arrive at the desired accuracy, and finally it is more time.
- Here we were working with simply cost function with few variables so in many cases we can graph the solutions and the level curves, but is good to understand the way its works of the methods and have it in mind to find the solutions in a problems with many variables, because probably those are harder and would be delayed to solve.

ANNEX

The Matlab codes are:

Linear Programming

- Punto_1.m
- Punto_2.m

Quadratic Programming

- Punto_3.m
- Punto_5.m

Optimization without constraints

- Punto_6.m

Optimization with constraints

- Punto_7.m
- Punto_8.m

Convex Optimization

- Punto_10.m
- Punto_11.m

REFERENCES

[1] Boom T.V., De Shutter B., Optimization in Systems and Control. Delft center of systems and control, Delft University of Technology, July 2017. Netherlands.
[2] Matlab documentation

*Jorge Eduardo Durán Vásquez, C.C.: 1037634392
Ing. Electricista e Ing. Administrador
Facultad de minas UNAL
jeduranvas@gmail.com