# Practical session 5: A non-differentiable, convex function

JALAL DOUIRI | FERRAN DALMAU
Optimization Techniques, UPF

15 de junio de 2019

## 1. Duality based Methods for Convex Optimization

### 1.1. Min - Max Theorem

Sigui $L : XxY \to \mathbb{R}$ una funció amb dues variables, $x \in X$ i $y \in Y$ llavors sempre tenim que:

$$\max_y \min_x L(x,y) \leq \min_x \max_y L(x,y)$$

Assumint que el minim i el maxim de la funció existeix.

### 1.2. Duality Gap

$$DG = min_x max_y L(x,y) - max_y min_y L(x,y) \geq 0$$

Si el Dulity Gap es igual a 0 (DG = 0), en un punt $(x_0, y_o)$
on $x_o$ fa minim $L(x_0, y_0)$ i $y_0$ fa maxim $L(x_0, y_0)$
tenim un punt de sella.
(DG=0 es un condició nessesaria)

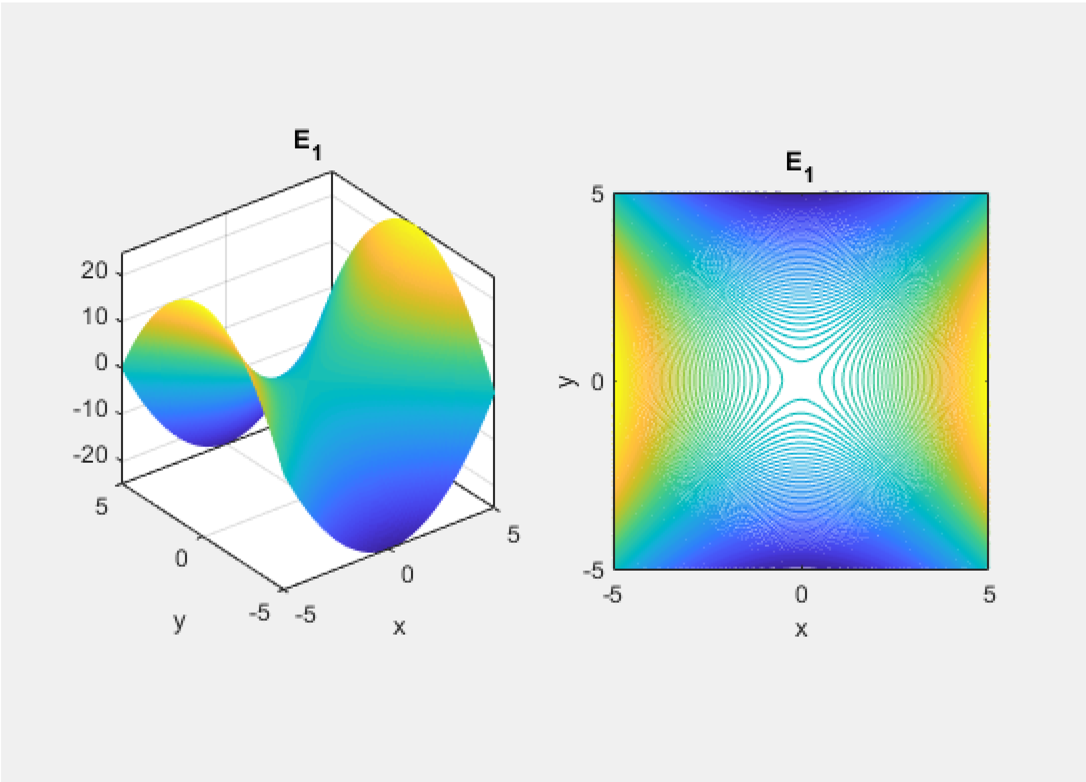
### 1.3. Lagrangian Duality

L(x,$\lambda, \mu) = f(x) - \sum_{i=1}^{k} \lambda_i c_i(x) - \sum_{j=1}^{r} \mu_i d_i(x)$
$d_i$ restriccions de desigualtat
$c_i$ restriccions d'igualtat

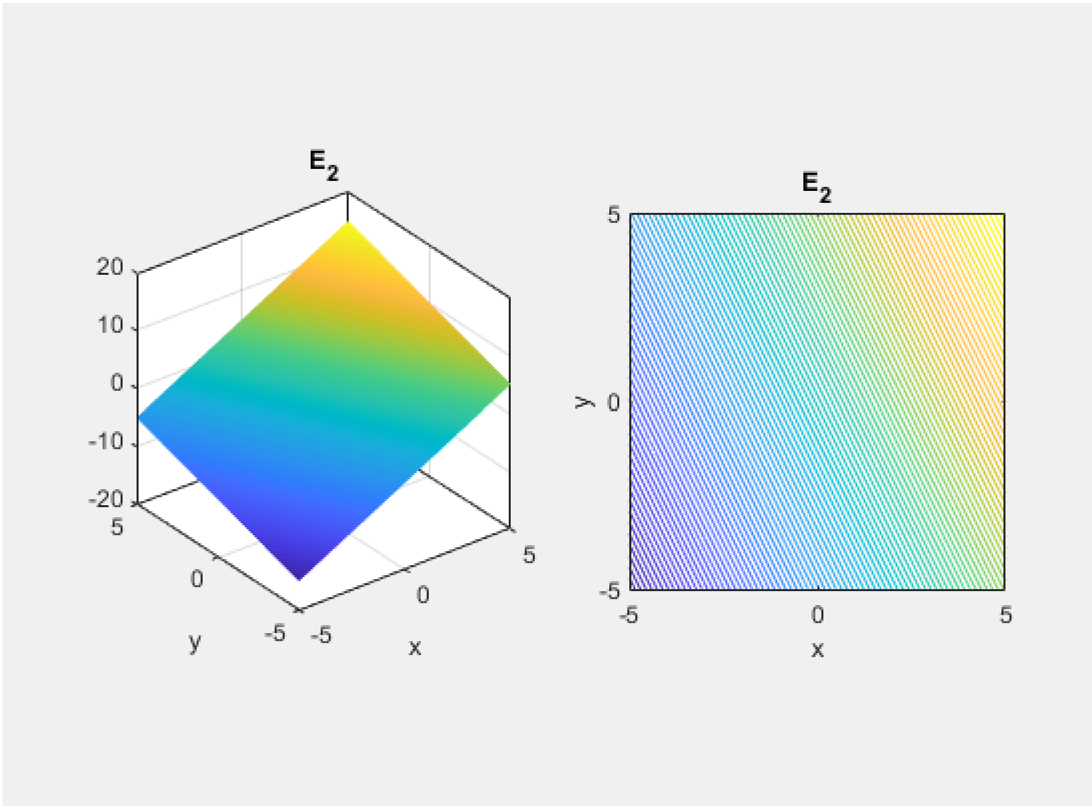
# 2. Minimization of a convex non-differentiable function

## 2.1. Removing the non-differentiability with an auxiliary variable

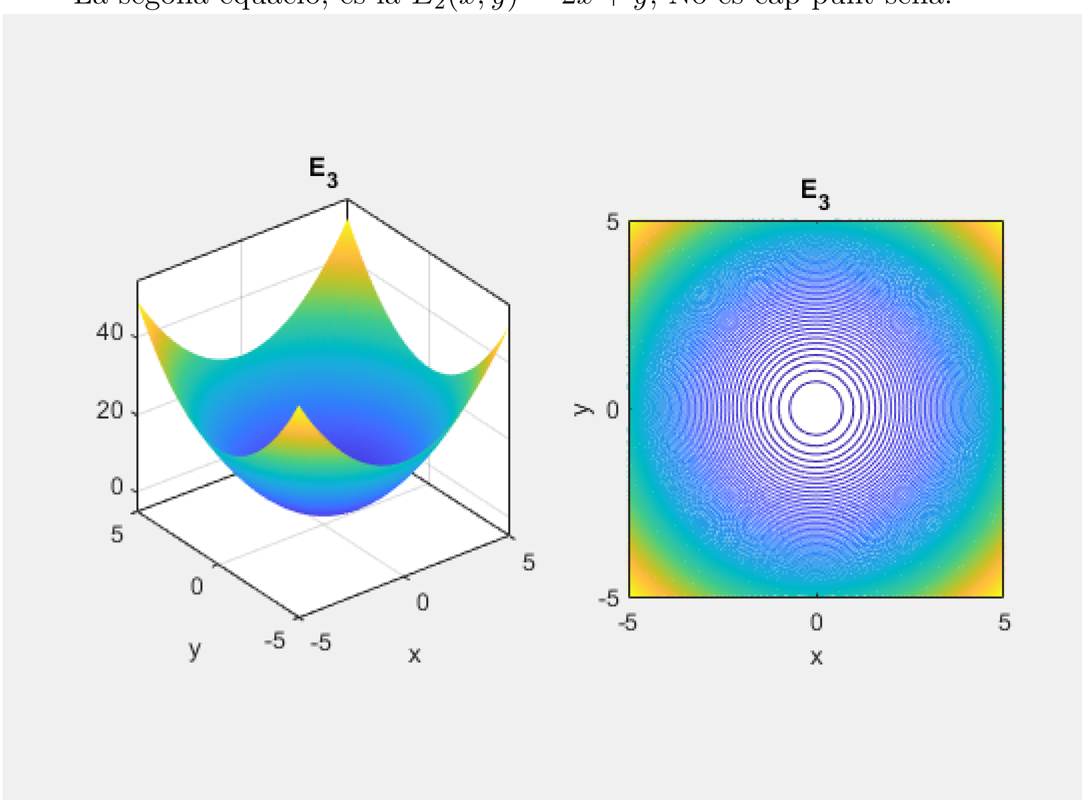
### 2.1.1. Run *toy_saddle_points.m* and determine which of the functions displayed presents a saddle point and which do not.



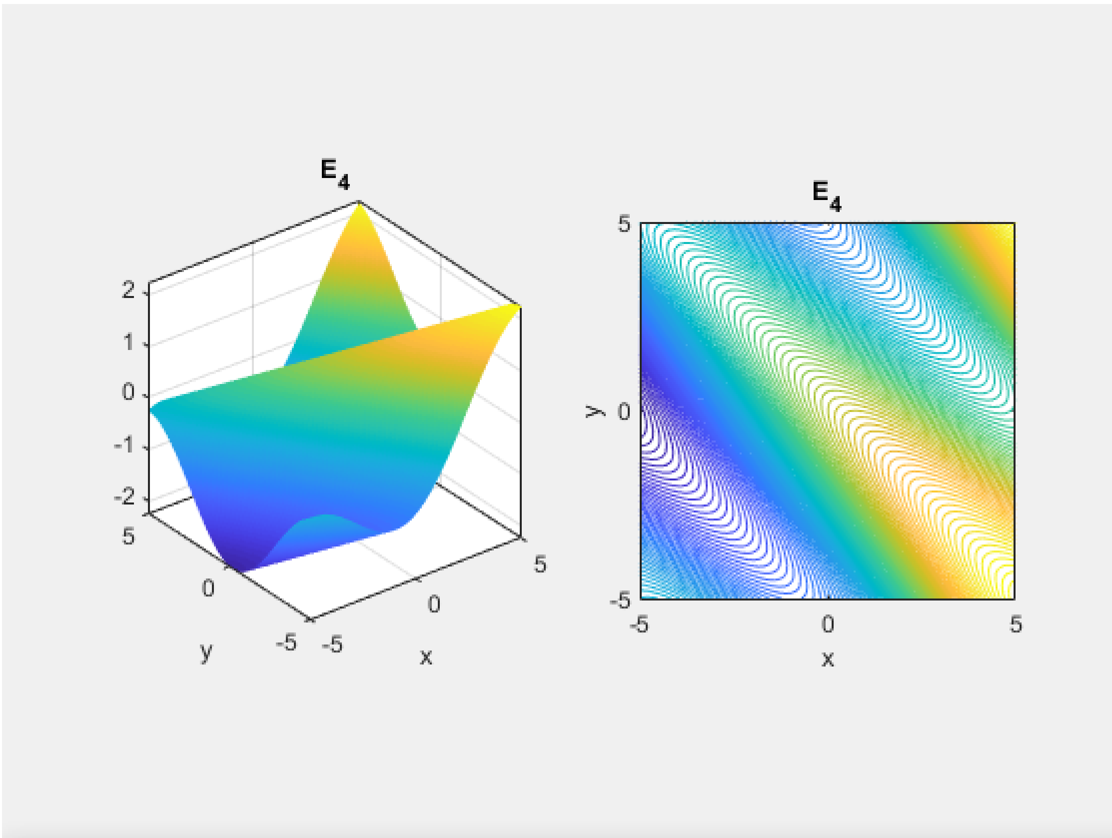

La primera equació que és la $E_1(x, y) = x^2 - y^2$, té un punt de sella en el punt (0,0).




La segona equació, és la $E_2(x, y) = 2x + y$, No és cap punt sella.




La tercera equació, és la $E_3(x, y) = x^2 + y^2$, No és cap punt sella.

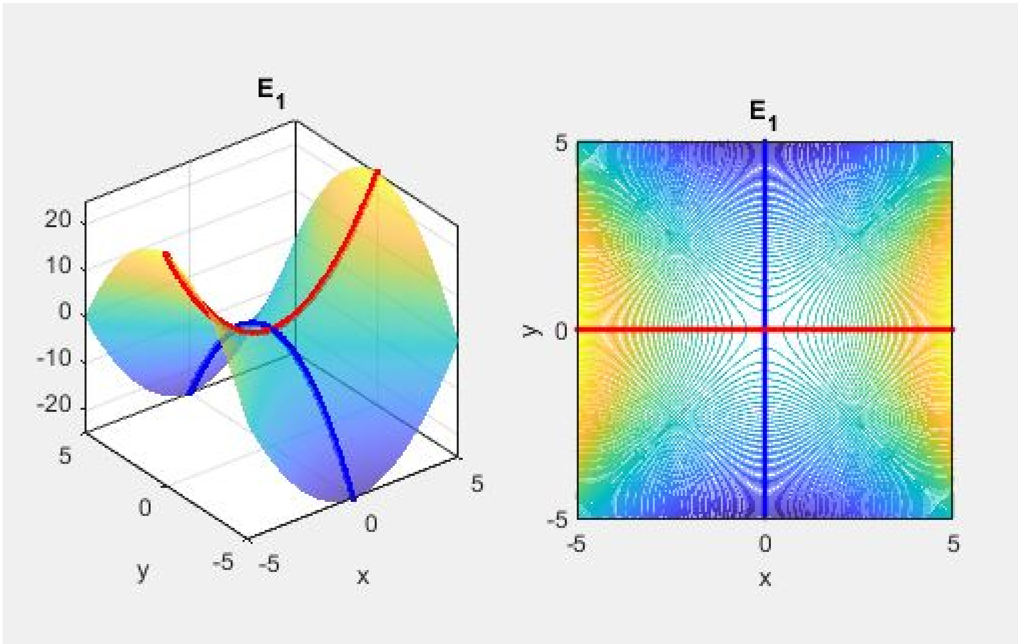


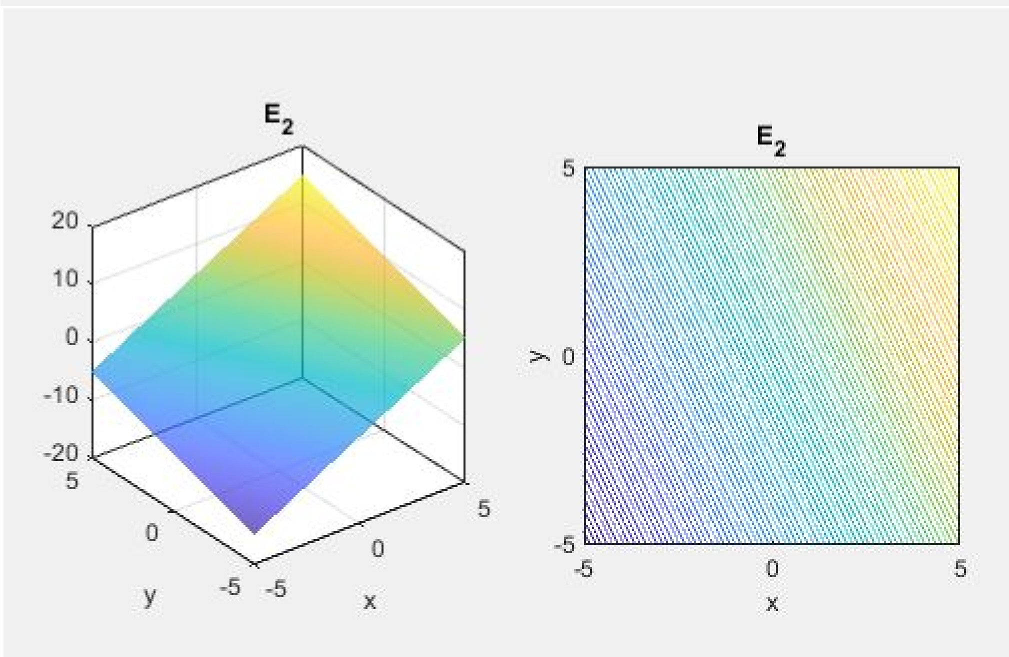

La quarta equació, és la $E_4(x,y) = cos(\frac{3}{5}(x+y))$, No és cap punt sella.

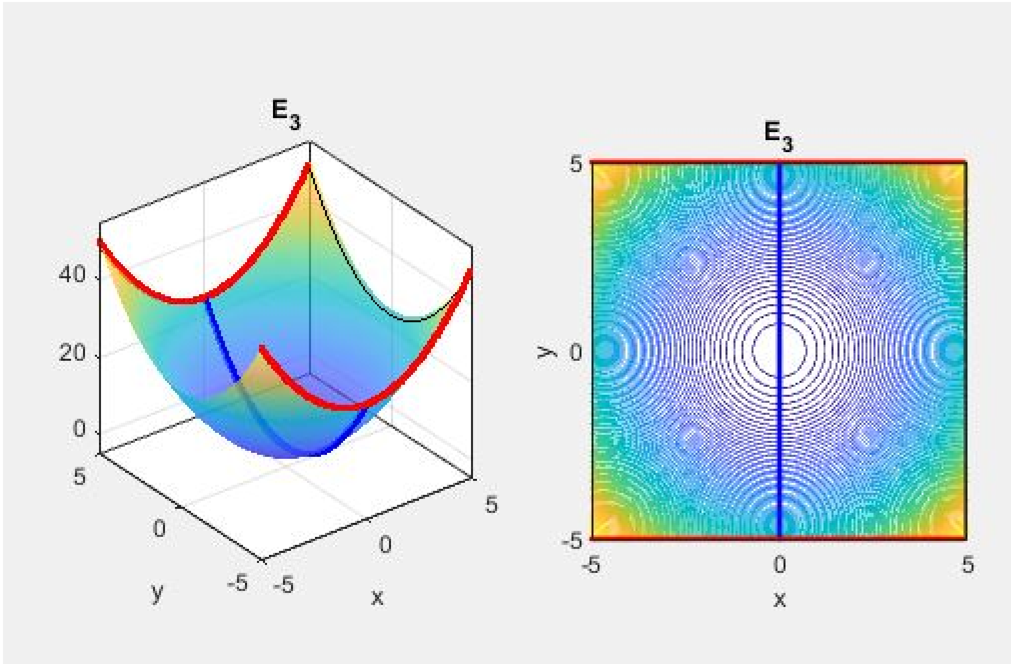
## 2.2. The primal-dual problem: finding a saddle point

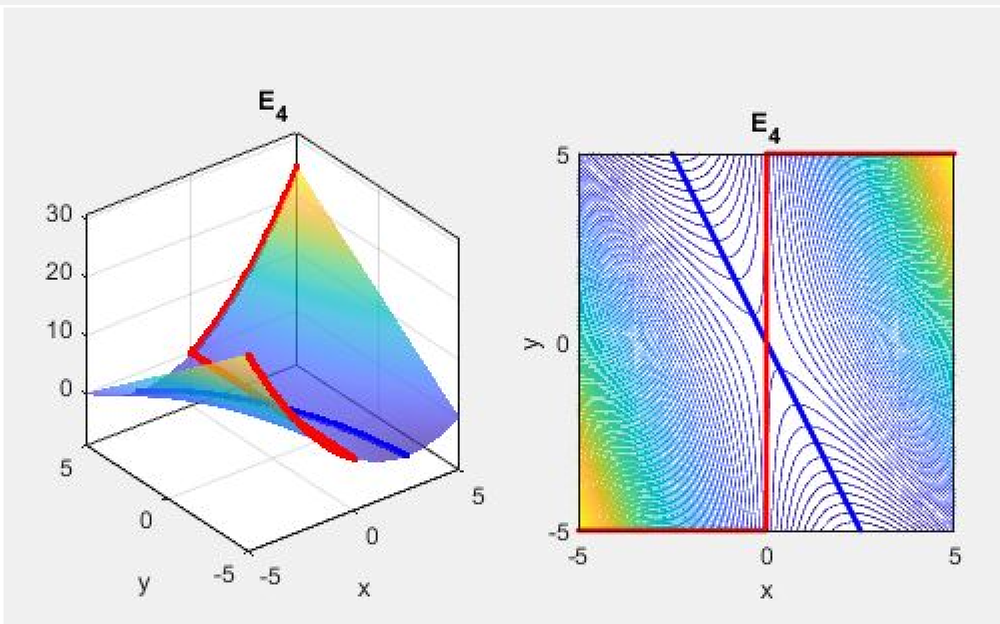
Al executar els archius plot_saddel_point1, plot_saddel_point2, plot_saddel_point3, plot_saddel_point4, plot_saddel_point5, enn les funcións que tenen un saddel point ens troba el punt, el metode utilitzat en aquests fitxers, és el buscar el maxim de la part concava, y el minim de la part convexa, com podem observar en el grafic de la funcó E1, troba perfectament el punt sella, el cual es el (0,0), la linea vermella es el min en funció de X, que seria la convexa y el blau el max en funció de Y.

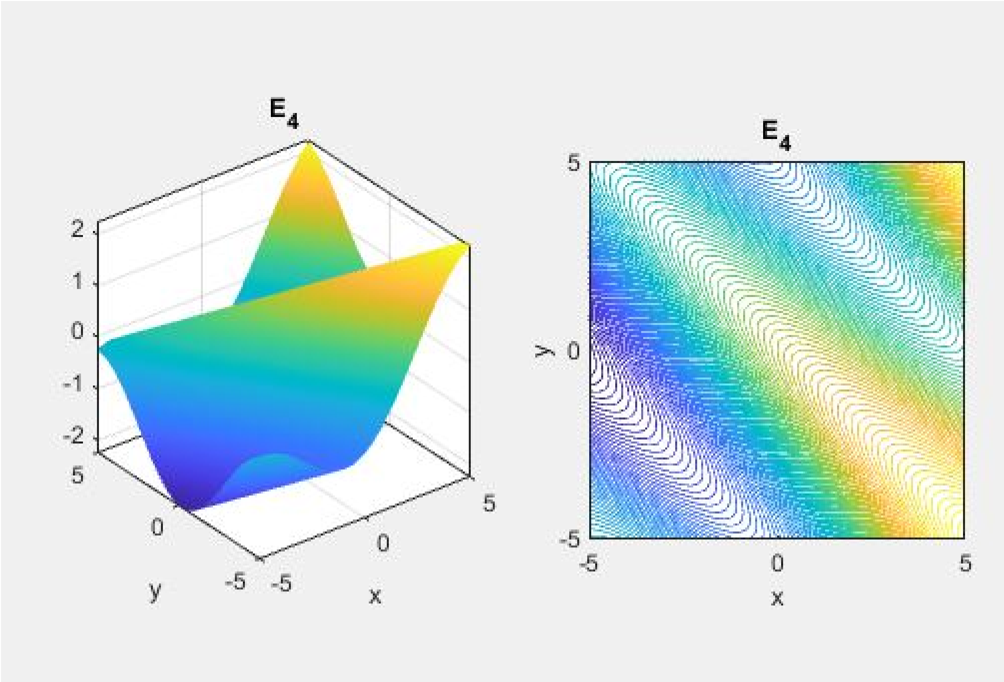
La Figura de la E2, es un punt, la cual consa no te ni convexitat ni concavitat, La E3, en compilarse el minim de y que es la linea blava, es compleix i passa per el minim, pero el max de x dista molt dels punts que passa la Y, la cual cosa deduim que nomes hi ha un minim i cap punt de sella, La E4, es troben el min de y y el max de x, el E5 conte molts minims y maxims la cual cosa es mes complexe de expliar.


E1
20
10
0
-10
-20
5
0
y
-5
-5
0
x
5
E1
5
y 0
-5
-5
0
5
x


E2
20
10
0
-10
-20
5
0
y
-5
-5
0
x
5
E2
5
y 0
-5
-5
0
5
x


E3
40
20
0
5
0
y
-5
-5
0
5
x
E3
5
y 0
-5
-5
0
5
x


E4
30
20
10
0
5
0
y
-5
-5
0
5
x
E4
5
y 0
-5
-5
0
5
x




**2.2.1.** **Complete the MatLab function *toy_primal_dual.m*. Follow the comments provided in the code.**

toy_primal_dual.m

```
x_old  = x;    % this is just for display
y_old = y;   % this too
grad_x = (transpose(A)*y) + (1/lambda)*(x-b);
%update x by gradient descent with step delta
x = x_old - theta*(grad_x);
%update y by gradient ascent with step theta
v = y_old+(delta*A*x);
y = v/max(1,sqrt(sum(v.^2)));
```

## 2.3. The dual problem: constrained maximization

### 2.3.1. Complete the MatLab function *toy_dual.m*. Follow the comments provided in the code.

toy_dual.m

```
v = y_old+(delta*A*x);
y = v/max(1,sqrt(sum(v.^2)));
%compute primal variable as x^*(y)
x_old = x;
x = b-(lambda*transpose(A)*y);
```

Compute cost in toy_dual.m and toy_primal_dual.m

```
Ax = A*x;
E1 = sqrt(sum(Ax.^2));
E2 = sum((x - b).^2);
E  = E1 + (1/(2*lambda))*E2;
P = E;
Atxi = A'*y;
D = b'*Atxi - lambda/2*sum(Atxi.^2);
PD = repmat(E2,[dmg_sz 1])/(2*lambda) + y'*Ax;
dual_gap = P - D;
```

### 2.3.2. Run the MatLab scripts *toy_problem_R1.m* and *toy_problem_R2.m*. Explain the plots that are displayed in each case. Try different parameters as suggested in the comments in the code

Como se puede observar tanto entre las figuras 4 y 5 como entre las figuras 8 y 9, se puede observar la implementacion primal dual tiende a tener mas iteraciones que dual. Esto es debido en gran parte a que en **Dual** el punto optimo de x por una y determinada es determinado de forma directa, y no de forma iterativa con gradient descent.

Figura 1: Primal Dual starting point (3,0)

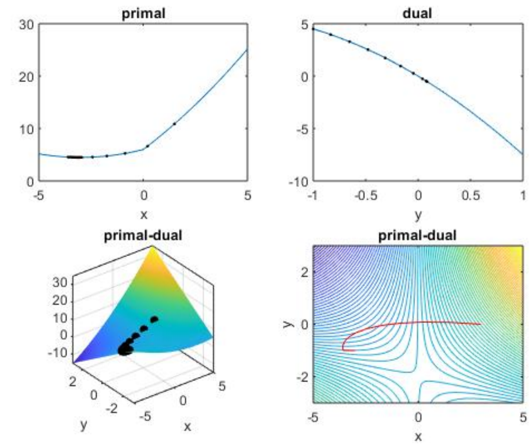



Figura 2: Primal Dual starting point (b,0)

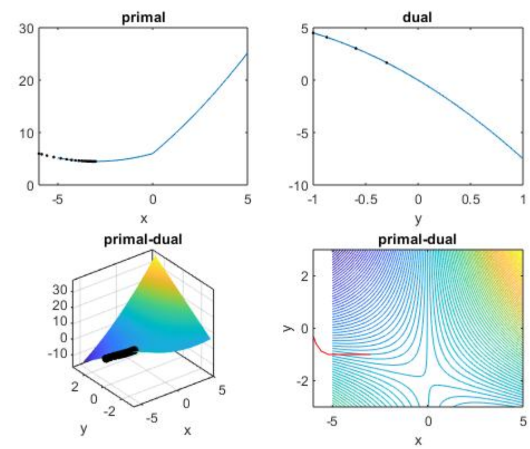



Figura 3: Dual starting point (b,0)

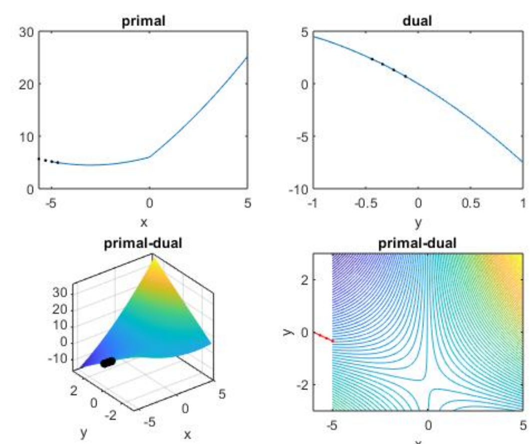


Figura 4: Primal dual iterations starting point (b,0)

```
-> it   31 ::  1000  -> |P( x(k) ) - D( y(k) )| = 4.32632e-05
-> it   32 ::  1000  -> |P( x(k) ) - D( y(k) )| = 3.00439e-05
-> it   33 ::  1000  -> |P( x(k) ) - D( y(k) )| = 2.08638e-05
-> it   34 ::  1000  -> |P( x(k) ) - D( y(k) )| = 1.44887e-05
-> it   35 ::  1000  -> |P( x(k) ) - D( y(k) )| = 1.00616e-05
-> it   36 ::  1000  -> |P( x(k) ) - D( y(k) )| = 6.98724e-06
```

Figura 5: Primal dual iterations starting point (b,0)

```
it    6 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 1.57183
it    7 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 1.15541
it    8 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.801478
it    9 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.501908
it   10 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.249584
it   11 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.0382653
it   12 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0
```

Figura 6: Primal Dual starting point (b,0)

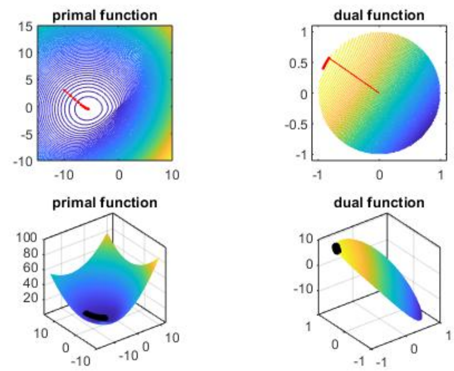



Figura 7: Dual starting point (b,0)

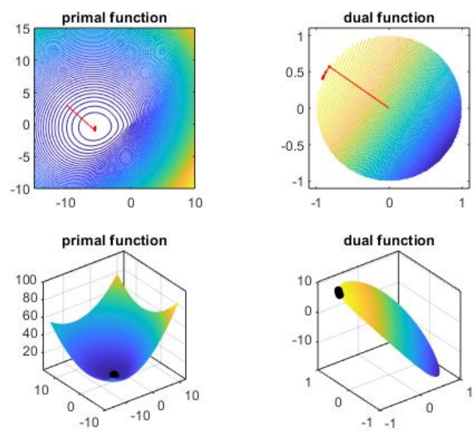



Figura 8: Primal Dual iterations starting point (b,0)

```
-> it   14 ::  1000  -> |P( x(k) ) - D( y(k) )| = 0.00148026
-> it   15 ::  1000  -> |P( x(k) ) - D( y(k) )| = 0.000812803
-> it   16 ::  1000  -> |P( x(k) ) - D( y(k) )| = 0.000450603
-> it   17 ::  1000  -> |P( x(k) ) - D( y(k) )| = 0.000251695
-> it   18 ::  1000  -> |P( x(k) ) - D( y(k) )| = 0.000141303
-> it   19 ::  1000  -> |P( x(k) ) - D( y(k) )| = 7.95406e-05
```

Figura 9: Dual iterations starting point (b,0)

```
-> it    3 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.0150329
-> it    4 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.00399406
-> it    5 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.00103551
-> it    6 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 0.000265029
-> it    7 ::  1000  -> |P( x^*(y(k)) ) - D( y(k) )| = 6.73832e-05
```

## 2.4. The dual gap and the stopping condition

Compute cost in toy_dual.m and toy_primal_dual.m

```
Ax = A*x;
E1 = sqrt(sum(Ax.^2));
E2 = sum((x - b).^2);
E  = E1 + (1/(2*lambda))*E2;
P = E;
Atxi = A'*y;
D = b'*Atxi - lambda/2*sum(Atxi.^2);
PD = repmat(E2,[dmg_sz 1])/(2*lambda) + y'*Ax;
dual_gap = P - D;
```

El criterio usado es la variable dual_gap, que indica la diferencia (gap) entre la funcion dual(**variable D**) y la primal (**variable P**), y cuando esta tiende a cero significa que hemos encontrado el punto que minimiza la funcion primal sometido a las constraints, con lo cual paramos de iterar.