Obligatorio 1: Optimalidad y métodos de descenso

Lucía Bouza - 4.289.797-0

Ejercicio 1 - Convexidad

PARTE A :

probar que $g(x) = \sum_i w_i . f_i(x)$ es convexa si $w_i \geqslant 0$

equivale a probar :

$$g(t.x + (1-t).y) \leqslant t.g(x) + (1-t).g(y)$$
$\forall x, y \quad , \quad t \in [0,1]$

DEMOSTRACIÓN:

sabemos $f_i(x)$ convexas $\forall i \in \{1..k\}$

entonces:

$$f_i(t.x + (1-t).y) \leqslant t.f_i(x) + (1-t).f_i(y) \quad \forall i \in \{1..k\}$$

si $w_i \geqslant 0 \rightarrow$ puedo multiplicar por w_i de ambos lados de la desigualdad, y ésta se mantiene $\rightarrow$

$$w_i . f_i(t.x + (1-t).y) \leqslant w_i\left(t.f_i(x) + (1-t).f_i(y)\right) \quad \forall i \in \{1..k\}$$

Realizo la sumatoria de ambos lados de la desigualdad y ésta se mantiene

$$\underbrace{\sum_i w_i . f_i(t.x + (1-t).y)}_{g(t.x + (1-t).y)} \leqslant \underbrace{\sum_i w_i\left(t.f_i(x) + (1-t).f_i(y)\right)}_{}$$

$$\sum_i \left(t.w_i.f_i(x) + (1-t).w_i.f_i(y)\right)$$

$$\sum_i t.w_i.f_i(x) + \sum_i (1-t).w_i.f_i(y)$$

$$t.\underbrace{\sum_i w_i.f_i(x)}_{g(x)} + (1-t).\underbrace{\sum_i w_i.f_i(y)}_{g(y)}$$

$$g(t.x + (1-t).y) \leqslant t.g(x) + (1-t).g(y)$$

L.Q.Q.D

PARTE B:

probar que $h(x) = \max\{f_i(x) : i = 1..k\}$ es convexa equivale a probar

$h(t \cdot x + (1-t) \cdot y) \leqslant t \cdot h(x) + (1-t) \cdot h(y) \quad \forall x, y \quad t \in [0,1]$

DEMOSTRACIÓN:

Supongamos que

en x $h = f_A \longrightarrow f_A(x) \geqslant f_i(x) \quad i = 1..k$

en y $h = f_B \longrightarrow f_B(y) \geqslant f_i(y) \quad i = 1..k$

en $t \cdot x + (1-t) \cdot y$ $h = f_C \rightarrow f_C(t \cdot x + (1-t) \cdot y) \geqslant f_i(tx + (1-t) \cdot y)$ $i = 1..k$

entonces queremos probar:

$$f_C(t \cdot x + (1-t) \cdot y) \leqslant t \cdot f_A(x) + (1-t) \cdot f_B(y)$$

Sabemos f_A, f_B, f_C convexas $\longrightarrow$

$$f_C(t \cdot x + (1-t) \cdot y) \leqslant t \cdot f_C(x) + (1-t) \cdot f_C(y)$$

tambien sabemos $f_C(x) \leqslant f_A(x)$ entonces:

$f_C(y) \leqslant f_B(y)$

$$t \cdot f_C(x) + (1-t) \cdot f_C(y) \leqslant t \cdot f_A(x) + (1-t) \cdot f_B(y)$$

entonces por transitivo:

$$f_C(t \cdot x + (1-t) \cdot y) \leqslant t \cdot f_A(x) + (1-t) \cdot f_B(y)$$

$$h(t \cdot x + (1-t) \cdot y) \leqslant t \cdot h(x) + (1-t) \cdot h(y)$$

h es convexa.

L.Q.Q.D

PARTE C:

Probar que $L(x) = f_1(A.x+b)$ es convexa

equivale a probar

$L(t.x + (1-t).y) \leq t.L(x) + (1-t).L(y) \quad \forall x,y \quad t \in [0,1]$

$f_1(A.(t.x + (1-t).y) + b) \leq t.f_1(A.x+b) + (1-t).f_1(A.y+b)$

DEMOSTRACIÓN:

$f_1(A(t.x + (1-t).y) + b) = f_1(A.tx + A.y - A.t.y + b) =$

sumo y resto el término $t.b$

$f_1(A.t.x + t.b + Ay - Aty - tb + b) =$

$f_1(t.(Ax+b) + (1-t)(Ay+b))$ *

Por otro lado sabemos f_1 es convexa →

$f_1(t.x + (1-t).y) \leq t.f_1(x) + (1-t).f_1(y) \quad \forall x,y$

si tomamos como $x = A.x+b$ e

$y = A.y+b$

$\underbrace{f_1(t.(A.x+b) + (1-t).(A.y+b))} \leq t.f_1(A.x+b) + (1-t)(Ay+b)$

Por desarrollo *
este término es
equivalente a:

$f_1(A(t.x + (1-t).y) + b)$

entonces:

$f_1(A(t.x + (1-t).y) + b) \leq t.f_1(Ax+b) + (1-t).f_1(Ay+b)$

$L(t.x + (1-t).y) \leq t.L(x) + (1-t).L(y)$

L convexa

L.Q.Q.D

PARTE D:

Probar que $Y = \bigcap_i X_i$ es un conjunto convexo

sean a y b $\in Y$

¿ $t.a + (1-t).b \in Y$? $/ t \in [0,1]$

DEMOSTRACIÓN:

$$\left.\begin{array}{l} a \text{ y } b \in Y \longrightarrow a \text{ y } b \in X_i \quad i=1..p \\ \qquad\qquad\qquad X_i \text{ convexo} \quad i=1..p \end{array}\right\}$$

$$t.a + (1-t).b \in X_i \quad i=1..p$$

$$\downarrow$$

$$t.a + (1-t).b \in \bigcap_i X_i = Y$$

$$\downarrow$$

Y convexo

L.Q.Q.D

PARTE E:

Mostrar que una Bola euclidea

$B(c,r) = \{x : \|c - x\| \le r\}$ es convexa

hay que demostrar que dado $x, y \in$ a la Bola

$t \cdot x + (1-t)\cdot y \in$ Bola

DEMOSTRACIÓN:

$$¿\|c - (t\cdot x + (1-t)\cdot y)\| \le r?$$

$$\|c - (tx + (1-t)\cdot y)\| = \|c - tx - y + ty\| \xrightarrow{\text{sumo y resto } tc}$$

$$= \|c - y - tx + + tc + ty - tc\| =$$

$$\|t(c-x) + (1-t)\cdot(c-y)\| \le$$

$$\|t(c-x)\| + \|(1-t)\cdot(c-y)\| =$$

$$|t| \cdot \underbrace{\|c - x\|}_{\le r} + |1-t| \cdot \underbrace{\|c-y\|}_{\le r} \le$$

$$|t|\cdot r + |1-t|\cdot r = |1 + \cancel{t} - \cancel{t}|\cdot r = r$$

entonces:

$$\|c - (t\cdot x + (1-t)\cdot y\| \le r \longrightarrow$$

$t\cdot x + (1-t)\cdot y \in B(c,r) \longrightarrow$ Bola euclidea es convexa

L.q.q.D

PARTE F:

Sean $x_1 \cdots x_k \in \mathbb{R}^n$ mostrar que

$C = \{ \sum_i \theta_i . x_i : \sum_i \theta_i = 1, \ \theta_i \geq 0 \ \forall i \}$ es convexo

dado $x = \sum \theta_{x_i} x_i \in C$

$y = \sum \theta_{y_i} x_i \in C$

¿$t.x + (1-t).y \in C$?

DEMOSTRACIÓN:

$$t.x + (1-t).y = t. \sum_i \theta_{x_i} . x_i + (1-t) . \sum_i \theta_{y_i} x_i =$$

$$\sum_i t . \theta_{x_i} x_i + \sum_i (1-t) \theta_{y_i} . x_i =$$

$$\sum_i (t \theta_{x_i} x_i + (1-t) . \theta_{y_i} . x_i) =$$

$$\sum_i (t \theta_{x_i} + (1-t) . \theta_{y_i}) . x_i$$

Lo que falta demostrar es que

$$\sum_i (t . \theta_{x_i} + (1-t) . \theta_{y_i}) = 1$$

$$\sum_i t . \theta_{x_i} + (1-t) . \theta_{y_i} = t \cdot \underbrace{\sum \theta_{x_i}}_{\substack{1 \text{ porque} \\ x \in C}} + (1-t) . \underbrace{\sum_i \theta_{y_i}}_{\substack{1 \text{ porque} \\ y \in C}}$$

$$= t + 1 - t = 1$$

entonces

$t . x + (1-t) . y \in C$

L.Q.Q.D

Ejercicio 2 - Interpretación gráfica

Parte A

PARTE A

$$\min_{x,y} \; -\log(y^2 - x^2) \; / \; \begin{array}{l} x^2 + y^2 \leq 1 \\ 2x - y \leq 0 \\ y \geq 1/2 \\ x \geq 0. \end{array}$$

$$-\log(y^2 - x^2) = -\log\big((y+x)\cdot(y-x)\big) =$$

$$-\log(y+x) - \log(y-x)$$

POR PARTE C de ejercicio 1

siendo $f(x) = -\log(x)$ convexa $\rightarrow$ $-\log(y+x)$ y $-\log(y-x)$ son convexas

POR PARTE A de ejercicio 1

si $-\log(y+x)$ y $-\log(y-x)$ son convexas $\rightarrow$ su suma es convexa.

entonces: $-\log(y^2 - x^2)$ es convexa

L.Q.Q.D.

Parte B

PARTE B

Si Probamos

- A: $x^2 + y^2 \leq 1$ convexo
- B: $x \geq 0$ convexo
- C: $2x - y \leq 0$ convexo
- D: $y \geq 1/2$ convexo

entonces por parte d del ejercicio 1 la intersección es convexa por lo que X es convexo.

A. Probar $x^2 + y^2 \leq 1$ es convexo

Demostraremos que los puntos pertenecen a una Bola $(0,1)$ con norma 2.

$$x^2 + y^2 \leq 1$$

$$\sqrt{x^2 + y^2} \leq \sqrt{1}$$

$$\|v\| \leq 1 \quad / \quad v = (x,y)$$

$$\|0 - v\| \leq 1 \longrightarrow v = (x,y) \in B(0,1)$$

Bola euclídea es convexa $\longrightarrow$

$x^2 + y^2 \leq 1$ es convexo

¿ $x \geq 0$ convexo?

dado a y $b \in$ conjunto $\rightarrow a \geq 0$ y $b \geq 0$

¿ $t.a + (1-t).b \geq 0$ con $t \in [0,1]$?

$t \in [0,1] \rightarrow (1-t) \in [0,1] \rightarrow t \geq 0$

$$\left.\begin{array}{l} t \geq 0 \text{ y } a \geq 0 \rightarrow t.a \geq 0 \\ (1-t) \geq 0 \text{ y } b \geq 0 \rightarrow (1-t).b \geq 0 \end{array}\right\} \begin{array}{c} t.a + (1-t).b \geq 0 \\ \downarrow \\ x \geq 0 \text{ convexo} \end{array}$$

¿ $2x - y \leq 0$ convexo?

dado a y $b \in$ conjunto $\begin{cases} \rightarrow 2a_1 - a_2 \leq 0 \\ \rightarrow 2.b_1 - b_2 \leq 0 \end{cases}$

¿ $2.(t.a_1 + (1-t).b_1) - (t.a_2 + (1-t).b_2) \leq 0$
con $t \in [0,1]$?

Desarrollando:

$t.(2a_1 - a_2) + (1-t).(2b_1 - b_2)$

$t \in [0,1] \quad 2.a_1 - a_2 \leq 0 \longrightarrow t.(2a_1 - a_2) \leq 0$

$(1-t) \in [0,1] \quad 2.b_1 - b_2 \leq 0 \longrightarrow (1-t).(2b_1 - b_2) \leq 0$

entonces:

$t.(2a_1 - a_2) + (1-t).(2b_1 - b_2) \leq 0 \longrightarrow 2x - y \leq 0$ convexo.

B. $y \geq 1/2$ convexo

dado a y $b \in$ conjunto $\longrightarrow a \geq 1/2$, $b \geq 1/2$

¿ $t \cdot a + (1-t) \cdot b \geq 1/2$ con $t \in [0,1]$?

$t \cdot a + b - tb - 1/2 = t \cdot (a-b) + b - 1/2 =$

$t \cdot \left(a - 1/2 - b + 1/2 \right) + b - 1/2 =$

$t \cdot \left(a - 1/2 - (b - 1/2) \right) + b - 1/2 =$

$t \cdot (a - 1/2) - t \cdot (b - 1/2) + b - 1/2 =$

$t \cdot (a - 1/2) + (1-t) \cdot (b - 1/2)$

sumo y resto 1/2 en el termino que multiplica t.

sabemos que: $t \geq 0$

$1 - t \geq 0$

$a - 1/2 \geq 0$

$b - 1/2 \geq 0$

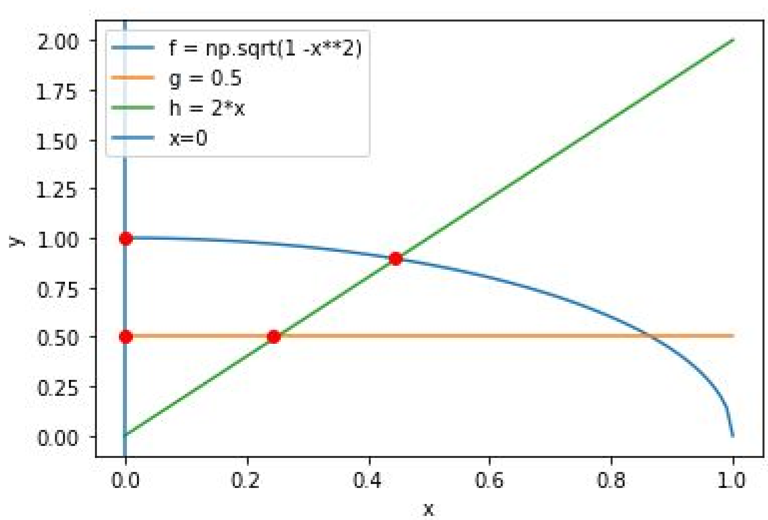
entonces

$t \cdot (a - 1/2) + (1-t) \cdot (b - 1/2) \geq 0 \longrightarrow y \geq 1/2$ convexo

L.Q.Q.D

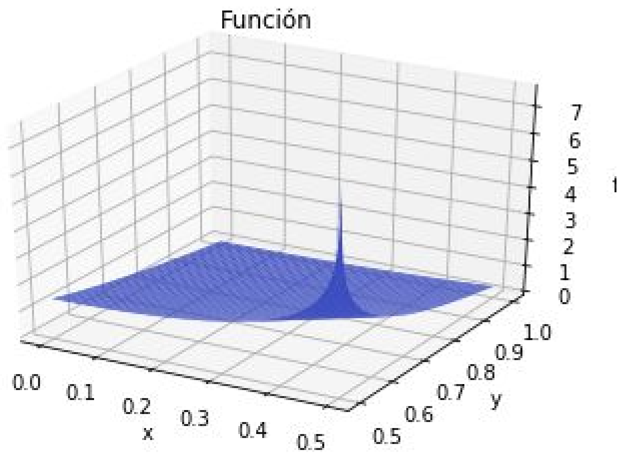
Parte C

Se muestra en la siguiente gráfica el conjunto factible X, interpretado como el área contenida en entre las gráficas f, g, h y x=0, indicada por los vértices rojos.Podemos observar que el conjunto es convexo.

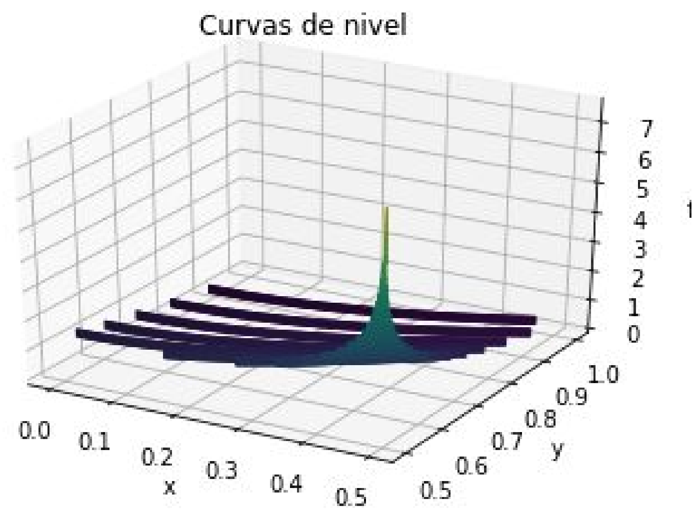


El punto donde se intersecta f y h es (0.44, 0.88)

En la siguiente gráfica observamos la función de costo. Dado el conjunto factible, se elige que el dominio de x sea [0,0.5], y el de y [0.5, 1]

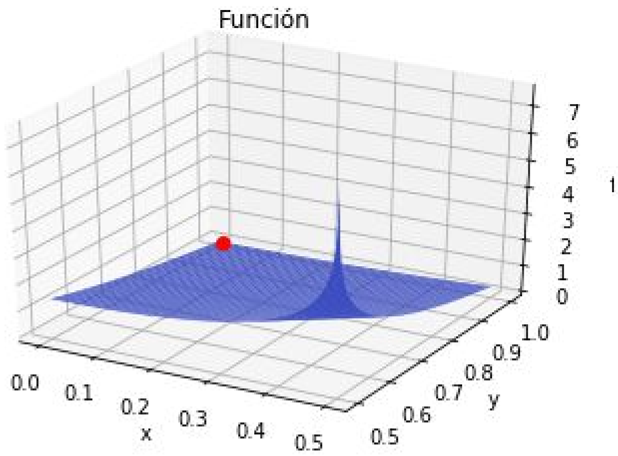


En la siguiente gráfica observamos las curvas de nivel de la función.



Parte D

Según los valores observados en la gráfica y las curvas de nivel, podemos ver que el mínimo se encuentra en el punto marcado con color rojo. Sabemos que ese punto es el (0,1) dado el dominio que elegimos para realizar el gráfico.



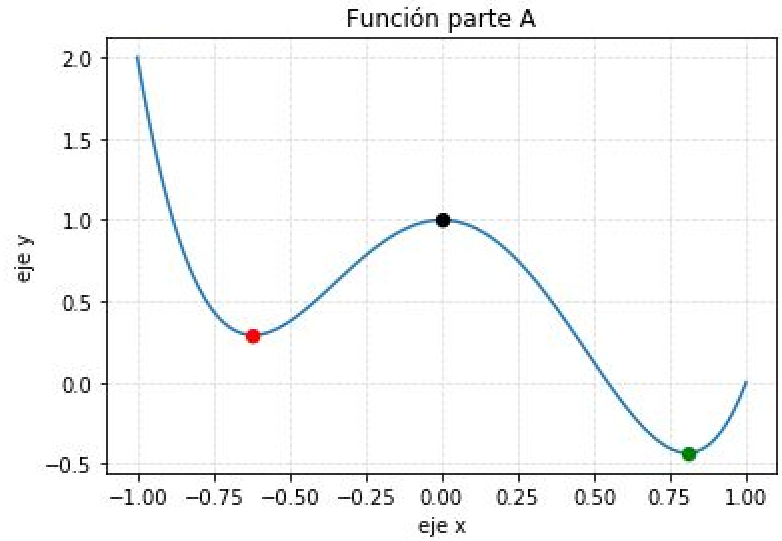
Ejercicio 3 - Puntos críticos y óptimos globales

Para este ejercicio se realizaron las gráficas con Python. El código puede encontrarse en el Zip adjunto a este informe, en el archivo Ejercicio3.py. Se utilizó la biblioteca Matplotlib para el graficado y Numpy para el manejo de los valores y utilización de funciones especiales como la norma.

Se indicará en las imágenes los óptimos locales, globales y puntos silla si los hubiere, dentro de las restricciones de dominio indicadas para cada caso. Seguiremos el siguiente código de colores para indicarlos:

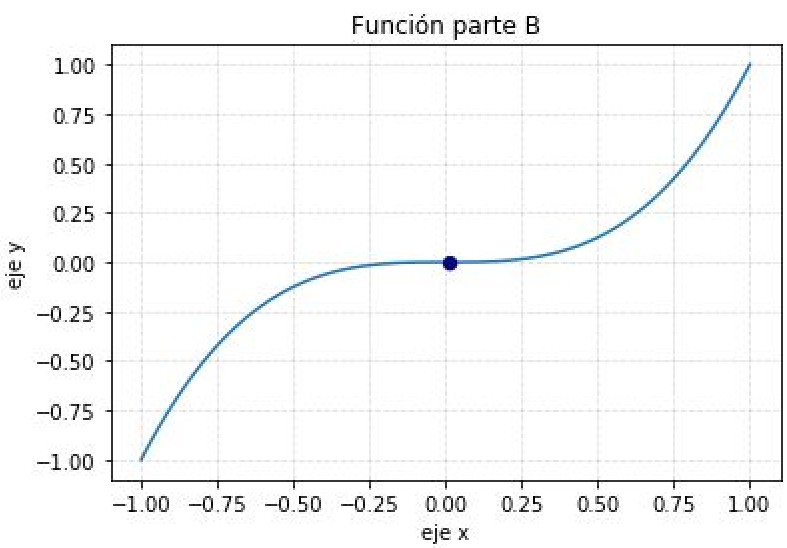
- Mínimo local = rojo
- Mínimo global (que también será local) = verde
- Punto Silla = azul
- Máximo local = Negro
- Máximo global (que también será local) = Violeta

Parte A



No se observan puntos silla. Con respecto al máximo global, no sabemos si existe o si cae fuera del conjunto, ya que en el conjunto de dominio donde estamos trabajando no hay un punto donde se anule el gradiente y sea máximo en el codominio. Encontramos un máximo local, pero sabemos que no es global.

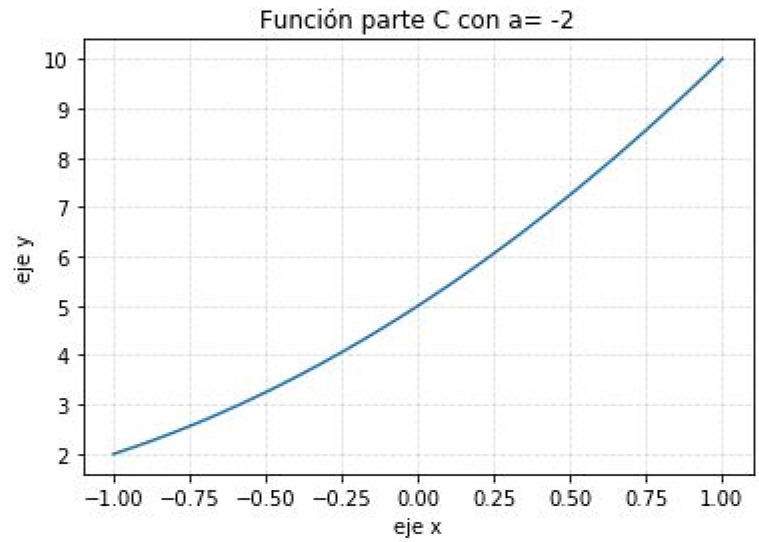
Parte B

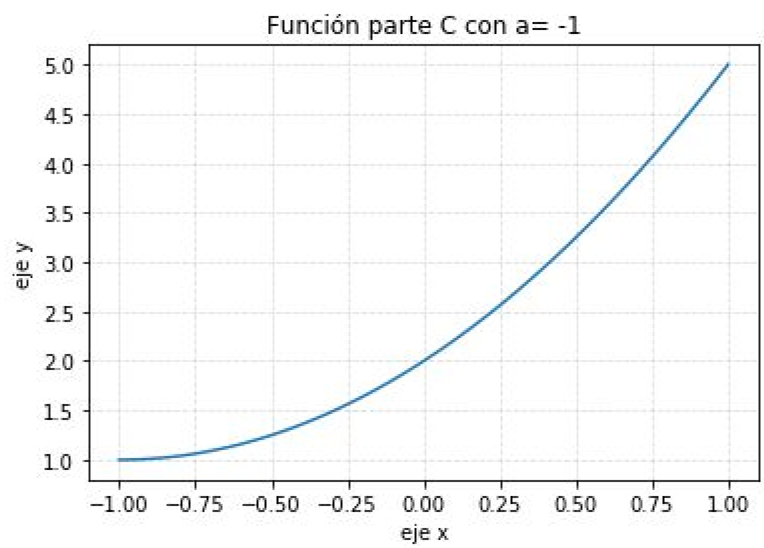


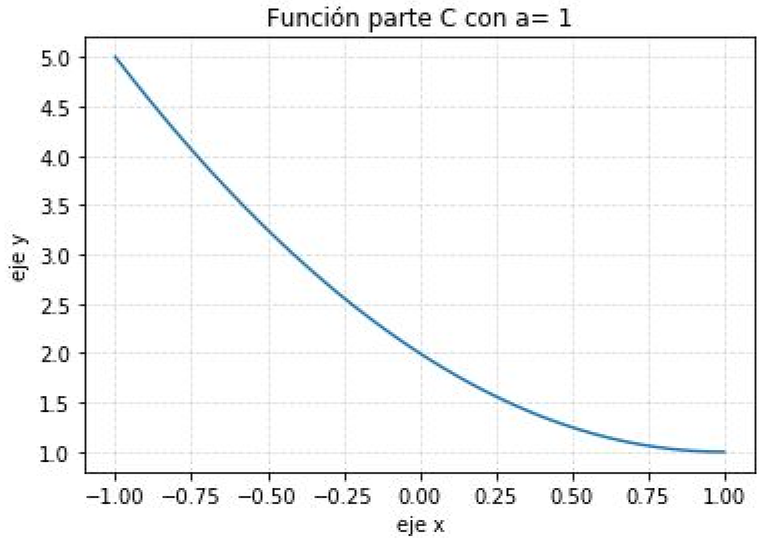
En el único punto donde se anula el gradiente es en el indicado en azul, y es un punto silla. En el conjunto donde trabajamos no hay mínimos ni máximos.

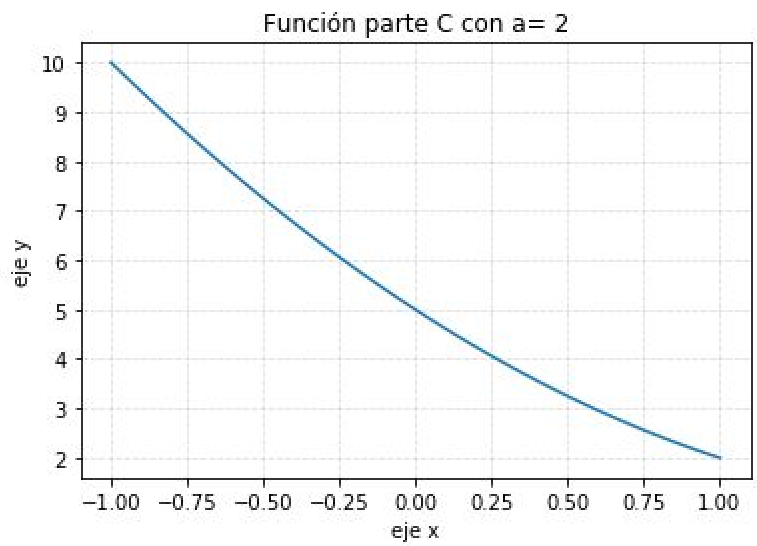
Parte C

Con ***a*** perteneciente a {-2, -1, 1, 2} no hay puntos donde se anule el gradiente en el conjunto de dominio donde trabajamos. A continuación sus gráficas:

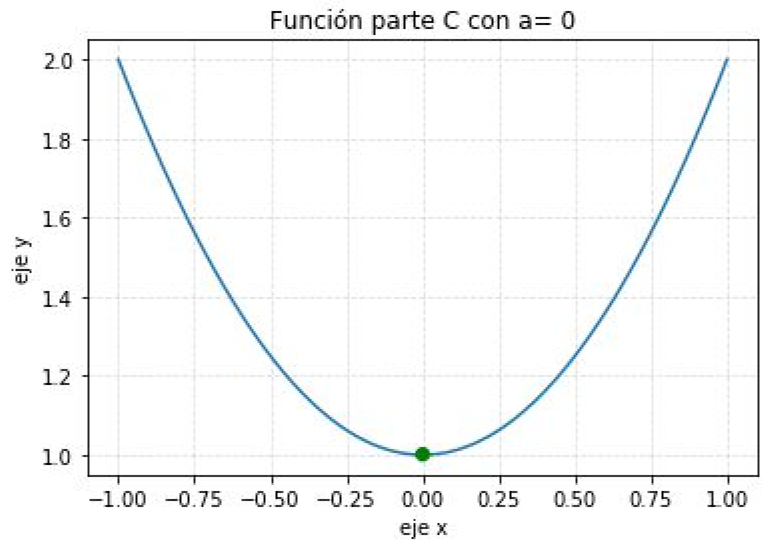






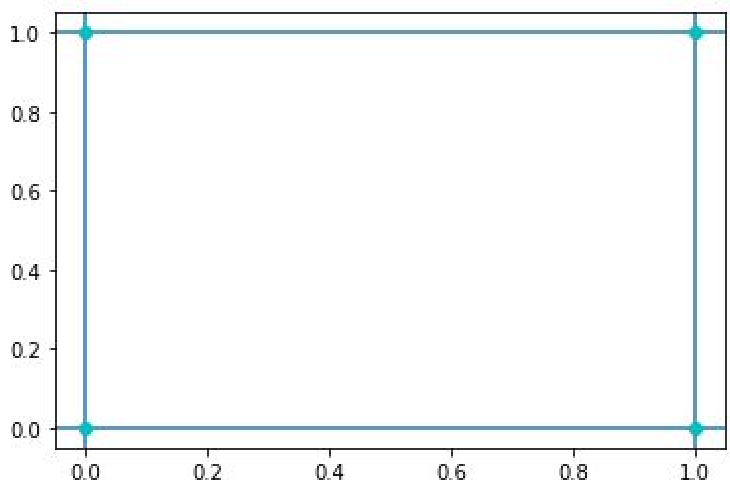


Con ***a=0*** tenemos un mínimo global:

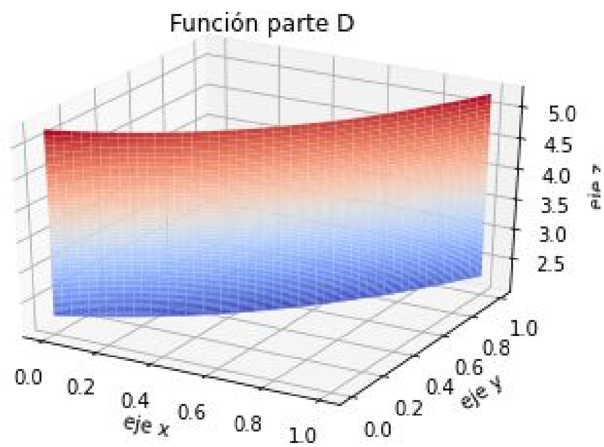


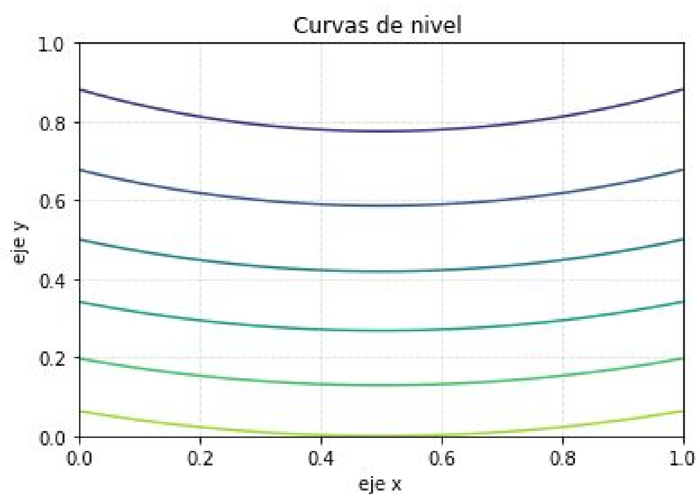
Parte D

Se grafica a continuación el conjunto B como el área delimitada por los gráficos. Podemos observar que el conjunto es convexo.



En el siguiente gráfico se muestra la función, la cual puede observarse que es convexa, así como sus curvas de nivel:





Ejercicio 4 - Decenso por gradiente

Parte A

condición optimalidad:
Estamos buscando puntos críticos donde el gradiente se anula. La idea es buscar mínimos locales. Si además la función es convexa (como es el caso) el mínimo encontrado será global.

Las ecuaciones normales proveen una forma cerrada de calcular los valores de x que minimizan la función de costo. El algoritmo asociado tiene O(n3), por lo que si la matriz A tiene muchas filas o columnas, puede llegar a ser muy lento, y, en esos casos, es preferible utilizar métodos iterativos. La forma dada por las ecuaciones normales para encontrar el x que minimiza ||Ax-b|| es: $x = (A^t.A)^{-1}.A^t.b$

Realizamos dicha operación en Python (El código puede encontrarse en el Zip adjunto a este informe, en el archivo Ejercicio4.py) y el resultado dado para los datos fueron:

[-0.13284693 0.1279282]

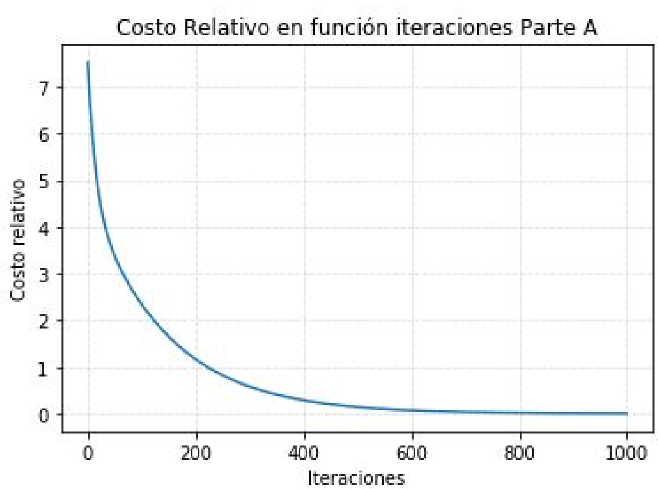
Parte B

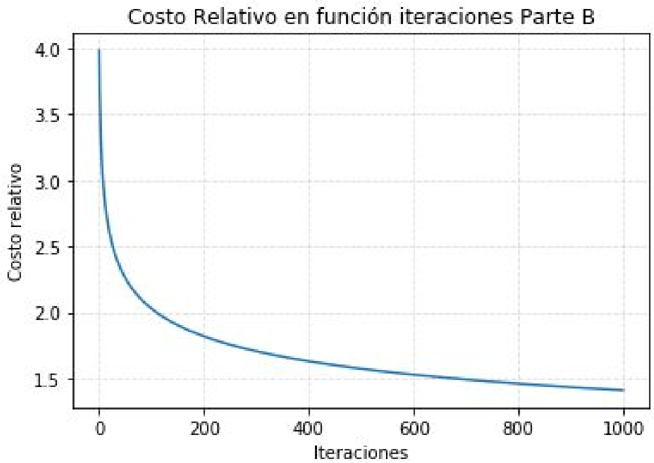
Se realiza el algoritmo, el cual se puede encontrar en el archivo Ejercicio4.py. Los resultados del algoritmo, con los diferentes pasos se encuentran en la siguiente tabla.
La condición de parada que se eligió fue la norma de la dirección menor a 0,001 o alcanzar las 1000 iteraciones.

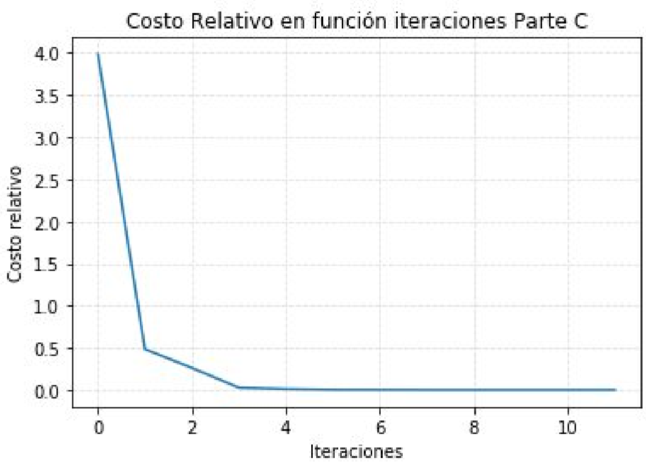
Paso	Resultado de x	Cantidad de iteraciones	Tiempo ejecución
a	[-0.13284573 0.12875739]	1000	0.02857685089111328
b	[-0.13245453 0.38774732]	1000	0.022561073303222656
c	[-0.13284687 0.12792824]	11	0.008019208908081055
d	[-0.13284693 0.1279287]	82	0.004542112350463867

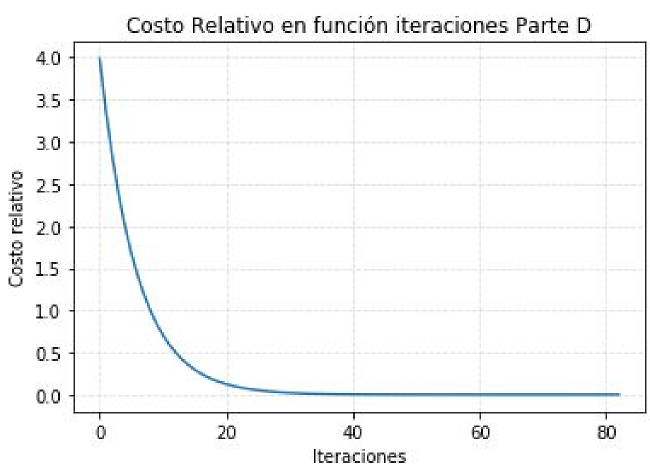
Parte C

A continuación se presentan las gráficas del costo relativo en función de las iteraciones.




Costo Relativo en función iteraciones Parte B
Costo relativo
Iteraciones


Costo Relativo en función iteraciones Parte C
Costo relativo
Iteraciones


Costo Relativo en función iteraciones Parte D
Costo relativo
Iteraciones

Parte D

Para este ejercicio se implementó el algoritmo de descenso por gradiente de regresión lineal, con la opción de elección del tipo de paso. El algoritmo implementado calcula también el error relativo de la solución encontrada en cada iteración para luego poder graficarla.
A continuación el código referente a la implementación de descenso por gradiente:

```
def gradient_descent(A, b, x, iterations, solution, part, epsilon):
    m = A.size
    vector_errorRelativo = list()
    count =1000

    for i in range(iterations):
        'calculos previos'
        error = np.dot(A, x) - b

        'determino alpha segun el ejercicio'
        if (part=="A"):  alpha = 1/(2*(np.linalg.norm(A))**2)
        elif (part=="B"):  alpha = 0.001 * (1/(i+1))
        elif (part=="C"):  alpha = find_Alpha_LineSearch(0.01, 50, A, b, x, error, m)
        else: alpha = find_Alpha_Armijo(0.001, A, b, x, error, m)

        'hago calculo de desenso'
        direccion = (1/m) * np.dot(A.T, error)
        x = x - (alpha * direccion)

        'guardo error relativo'
        errorRelativo = np.linalg.norm(solution - x)/np.linalg.norm(solution)
        vector_errorRelativo.append(errorRelativo)

        'si la dirección es menor a epsilon, finalizo búsqueda'
        if (np.linalg.norm(direccion) <= epsilon):
            count = i
            break

    return x, vector_errorRelativo, count
```

Para el caso de la elección de paso aplicando *Line Search* o *Armijo*, se implementaron funciones auxiliares para hallar el paso. El código de los algoritmos implementados se detalla a continuación, con los comentarios pertinentes:

```python
def find_Alpha_LineSearch(s, iterations, A, b, x, error, m):
    'seteo mínimos para alpha y el valor mínimo de la función'
    min_Alpha = s
    min_func = eval_f(A,b, x)
    'seteo el valor del paso con el que me voy a mover en el segmento [0,s]'
    'para buscar el mejor alpha que minimiza la función'
    step = s/iterations
    for i in range(iterations,0,-1):
        'me muevo, calculo x en ese punto y evalúo la función'
        x_i = x - (step*i * (1/m) * np.dot(A.T, error))
        func = eval_f(A,b, x_i)
        'si la función en menor al mínimo hasta el momento,'
        'actualizo min_Alpha y min_func'
        if (func < min_func):
            min_Alpha = step*i
            min_func = func
    'Devuelvo el mejor alpha'
    return min_Alpha
```

```python
def find_Alpha_Armijo(betha, A, b, x, error, m):
    'inicializo variables de rango de busqueda, factor y sigma'
    s, p = 1, 0
    sigma = 0.1
    'inicializo variables necesarias para validar la desigualdad'
    grad = (1/m) * np.dot(A.T, error)
    func = eval_f(A,b, x)

    while (True):
        'me muevo, calculo x en ese punto y evalúo la función'
        p = p +1
        x_i = x - ((betha**p)* s * grad)
        func_betha = eval_f(A,b, x_i)
        condition = sigma * (betha**p)* s *  np.linalg.norm(np.dot(grad.T,grad))**2
        'si se da la siguiente desigualdad, encontré un punto donde la función disminuye
        'entonces finalizo búsqueda'
        if ( func_betha - func <= condition): break
    'cuando salimos del while, devuelvo el paso encontrado'
    return (betha**p)*s
```

Los resultados obtenidos con cada algoritmo, y las gráficas del error relativo vs. Iteraciones se encuentran en la parte B y C respectivamente

Análisis de los resultados

De los resultados obtenidos, podemos afirmar que la búsqueda del paso que ayude a acercarse más rápidamente al mínimo, realmente hace una diferencia sustancial con los métodos que utilizan un paso arbitrario o decreciente, sin tomar en cuenta el valor de la función para hallarlos.

El método de Armijo, no busca “el mejor paso” pero si busca un paso en el que me aseguro un valor de la función menor. En este caso obtenemos aproximaciones aún mejoras a las de de paso fijo, pero con menos de un 10% de las iteraciones que lleva el método de gradiente con ese tipo de paso.

El método Line Search (aunque en realidad se implementó limited line search, para acotar la búsqueda del paso en un rango) busca el mejor paso. Vemos que utilizando este método con solo 11 iteraciones llegamos a valores que tienen el error relativo aceptado.

¿Qué utilice menos iteraciones nos asegura que el método sea más veloz? No necesariamente, ya que tanto en el método de Line Search como en Armijo, es necesario realizar una búsqueda del paso óptimo o que cumple con un cierto criterio, lo cual también lleva tiempo. En la implementación realizada se decidió colocar contadores de tiempo para poder también evaluar los distintos métodos desde ese punto de vista. Los resultados los podemos encontrar también en la tabla de la parte B.

En particular, vemos que el método de Armijo es muy veloz y además lleva pocas iteraciones. Line Search, si bien lleva poquísimas iteraciones (11) sus iteraciones son más lentas en comparación a los otros métodos. Igualmente, este método es preferible a los métodos de paso fijo o decreciente arbitrario, ya que aunque sus iteraciones son más costosas, sumarizando, es menos costoso que el método A y B.

Ejercicio 5 - Problemas equivalentes

Parte A

a) PROBAR $\min_{x \in \mathbb{R}^n} \frac{1}{2} \cdot \|A.x - b\|^2 + \sum_1^n |x_i|$ es convexo.

Demostración

Por ejercicio 1, parte a, basta con demostrar que $\frac{1}{2} \|A.x - b\|^2$ y $\sum_1^n |x_i|$ son convexas

Demostrar $\frac{1}{2} \cdot \|Ax - b\|^2$ convexa

$f(x) = \frac{1}{2} \cdot \|Ax - b\|^2$

$\nabla f(x) = \cancel{\frac{1}{2}} \cdot \cancel{2} \cdot A^T \cdot (Ax - b)$

$\nabla^2 f(x) = A^T \cdot A \longrightarrow$ matriz simétrica.

Como $\nabla^2 f(x)$ se puede descomponer en la multiplicación de una matriz transpuesta y ella misma $(A^T \cdot A) \longrightarrow \nabla^2 f(x)$ es semidefinida positiva $\longrightarrow f(x)$ convexa.

Demostrar $\sum_1^n |x_i|$ convexa

$\sum_1^n |x_i| \longrightarrow \|x\|_1$

Por teórico las bolas en $\mathbb{R}^n$ son convexas

$\sum_1^n |x_i|$ convexa

Parte B

b) Demostrar PO y QP son problemas equivalentes donde

PO $\Rightarrow \min_{x \in \mathbb{R}^n} \frac{1}{2} \cdot \|Ax - b\|^2 + \sum_{1}^{n} |x_i|$

QP $\Rightarrow \min_{\substack{x \in \mathbb{R}^n \\ t \in \mathbb{R}^n}} \frac{1}{2} \cdot \|Ax - b\|^2 + \sum_{1}^{n} t_i$

$x_i \leq t_i$

$-x_i \leq t_i$

Demostración

Sea x^* la solución a PO. Hay que probar que x^* es la solución a QP, en particular para el término $\sum_{1}^{n} t_i$

entonces:

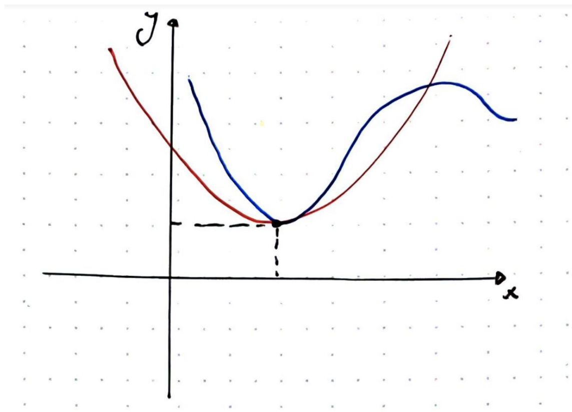
sea x^* que minimiza $\sum_{1}^{n} |x_i|$

tenemos las restricciones $x_i \leq t_i$ y $-x_i \leq t_i \rightarrow$
$|x_i| \leq t_i \longrightarrow |x_i^*| \leq t_i \quad i \in \{1 .. n\} \longrightarrow$
$\sum_{1}^{n} |x_i^*| \leq \sum_{1}^{n} t_i \longrightarrow$ x^* es quien minimiza $\sum_{1}^{n} t_i$
$\longrightarrow x^*$ es solución de QP.

Parte C

En la Parte A se demostró que cada uno de los términos era una función convexa. La función QP mantiene el primer término de la función PO, y para el segundo cambia la función valor absoluto por la función identidad, considerando las restricciones de los valores de ti. Podemos observar que ese segundo término preserva la convexidad. Además las restricciones son conjuntos convexos ya que es la intersección de semiplanos, los cuales son convexos. Por lo que QP es convexo.

Ahora bien, en este caso ambos problemas equivalentes son convexos, pero podría darse que para dos problemas que comparten la misma solución, uno sea convexo y el otro no. Supongamos las siguientes gráficas: ambas comparten el mismo mínimo global, pero la función roja es convexa, y la azul no.



Parte D

Los valores óptimos encontrados por el programa son:

- Para x [-0.13274639 0.12730003]
- Para t [0.13274639 0.12730003]

Vemos que |x| = t.
En la parte b demostramos que los problemas P0 y QP eran problemas equivalentes, es decir que la solución para ambos problemas es la misma. Aquí podemos ver que para el Problema P0, si utilizamos el x óptimo encontrado para el primer término, y el valor óptimo de t sustituído en el segundo, encontramos el valor mínimo de la función.

Ejercicio 6 - Mínimos cuadrados con restricción

Parte A

Demostración de convexidad de la función:

PARTE A.

$f(x) = 5x^2 + 5y^2 + 5x - 3y - 6xy + 5/4$

Restricción $x^2 + y^2 \leq R^2$ / $R \in \mathbb{R}^+$ constante.

Calcularemos la matriz Hessiana y probaremos que es semidefinida positiva.

$$Hf = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix} = \begin{pmatrix} 10 & -6 \\ -6 & 10 \end{pmatrix}$$

$\frac{\partial f}{\partial x} = 10x + 5 - 6y$ | $\frac{\partial f}{\partial y} = 10y - 3 - 6x$

$\frac{\partial^2 f}{\partial x^2} = 10$ | $\frac{\partial^2 f}{\partial y^2} = 10$

$\frac{\partial^2 f}{\partial x \partial y} = -6$

Calculo de valores propios:

$$|Hf - \lambda \cdot I| = \begin{vmatrix} 10-\lambda & -6 \\ -6 & 10-\lambda \end{vmatrix} = (10-\lambda)^2 - 36 = 0$$

$100 - \lambda^2 - 20\lambda - 36 = 0$

$+\lambda^2 - 20\lambda + 64 = 0 \Rightarrow \lambda = \frac{20 \pm \sqrt{400 - 4 \cdot (+1) \cdot 64}}{+2}$

$\lambda = \begin{cases} 16 \\ 4 \end{cases} \longrightarrow$ Los valores propios son positivos $\rightarrow f$ convexa.

Demostración de convexidad de la restricción:

La restricción del problema representa una bola euclídea de centro 0, y radio R. Por lo demostrado en el ejercicio 1, el conjunto es convexo

Parte B

hallaremos el punto donde el gradiente se anula

$$Df(x) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 10x + 5 - 6y \\ 10y - 3 - 6x \end{pmatrix}$$

$$\left.\begin{array}{l} 10x + 5 - 6y = 0 \\ 10y - 3 - 6x = 0 \longrightarrow y = \frac{6x+3}{10} \end{array}\right\} 10x + 5 - 6 \cdot \left(\frac{6x+3}{10}\right) = 0$$

$$6{,}4x + 3{,}2 = 0$$

$$x = -1/2$$

$$y = \frac{6 \cdot (-1/2) + 3}{10} = 0.$$

el mínimo de la función se da en $(-1/2, 0)$

¿el punto está adentro del conjunto de restricción?

Restricción $x^2 + y^2 \leq R^2$

como funciona para $R < 1/2$?

con $x = -1/2$ $y = 0 \longrightarrow (-1/2)^2 + 0^2 = \frac{1}{4}$

con $R = 1/2$ la restricción no está activa ya que el punto la cumple.

con $R < 1/2$ la restricción está activa

Parte C

Para este ejercicio se implementó el algoritmo PGD. Sabiendo que alpha=0, se tiene una simplificación del cálculo del siguiente punto. También la restricción del problema ayuda ya que el cálculo de la proyección se simplifica.

Con respecto al ejercicio 4, además de la de la forma de cálculo del siguiente punto, cambia la condición de parada. Ahora, no buscaremos que el gradiente sea menor a un cierto epsilon, ya que la solución podría no estar dentro del conjunto factible.

Como calculamos en los puntos anteriores, la restricción está activa para R<½, en particular para R=¼, por lo que la solución que encontremos para este problema no será (-½,0), que es la solución al problema sin restricciones.

Aquí se mostrará el algoritmo PGD con los comentarios, y también el cálculo de la proyección. En nuestro código existen otras funciones auxiliares, como el cálculo del gradiente, la función booleana de pertenencia al conjunto, o la función de búsqueda de sk por line search, entre otras; las cuales podrán ser vistas en el archivo Ejercicio6.py

```
def gradient_descent(x, iterations, part, epsilon, R, EnConjunto):
    vector_valorFuncionCosto= list()
    vector_puntosX= list()
    vector_puntosY= list()
    vector_NormaDiferencia = list()
    count =iterations
    sk = 0.001

    for i in range(iterations):

        'determino sk segun el ejercicio'
        if (part=="Decresciente"):  sk = (1/(i+1))
        else:  sk = find_s_LineSearch(1, 50, x, sk, R)

        'hago calculo de desenso'
        valAnterior = x
        valProx = x - sk * gradF(x)
        if (EnConjunto(valProx,R)): #si el punto pertenece al conjunto, no usamos la proyeccion.
            x = valProx
        else: # si el punto no pertenece al conjunto, debemos proyectar
            x = Proy_Point((valProx), R) #conociendo que alpha=1

        'guardo valores a graficar'
        vector_valorFuncionCosto.append(eval_f(x))
        vector_NormaDiferencia.append(np.linalg.norm(x - valAnterior))
        vector_puntosX.append(x[0])
        vector_puntosY.append(x[1])

        'si la norma entre el punto anterior y el nuevo es menor a epsilon, finalizo búsqueda'
        if (np.linalg.norm(x - valAnterior) <= epsilon):
            count = i
            break

    return x, vector_valorFuncionCosto, vector_NormaDiferencia, vector_puntosX, vector_puntosY, count
```

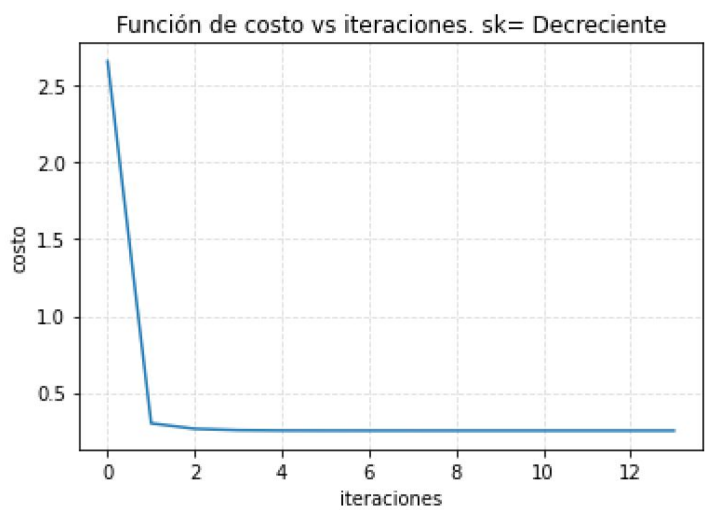
```
def Proy_Point(x, R):
    return (x/np.linalg.norm(x))*R
```

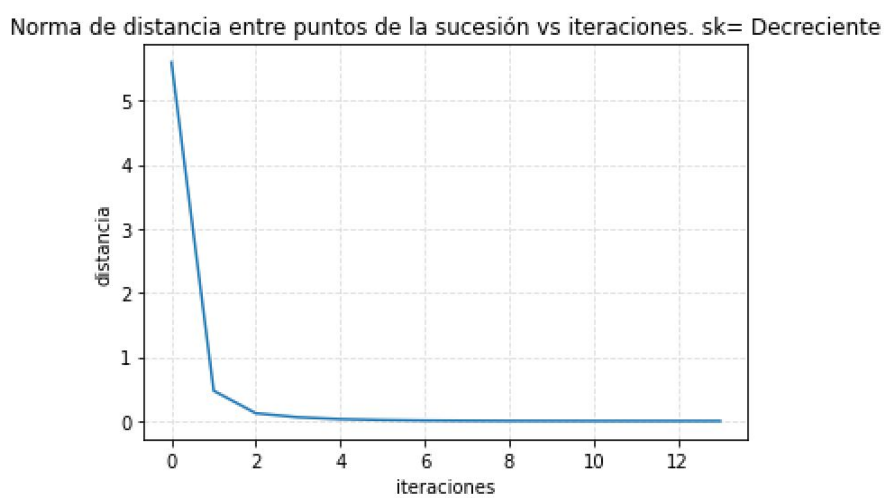
Los resultados numéricos los para los diferentes métodos de obtención de sk se resumen en la siguiente tabla.

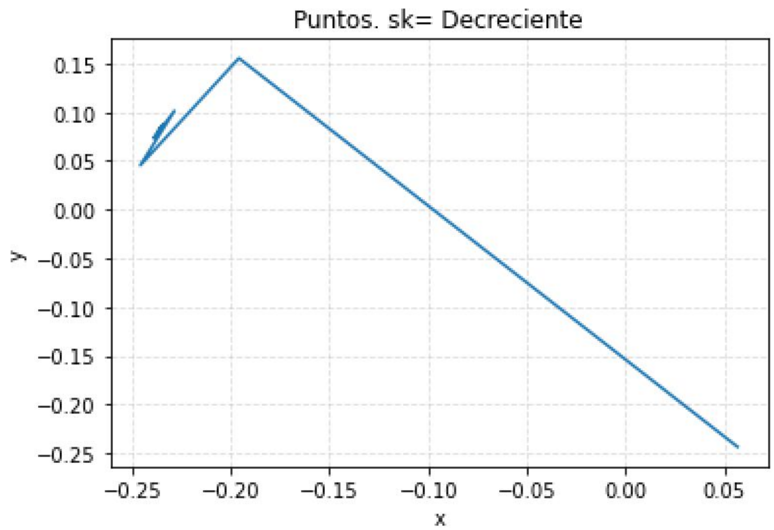
Se probó con diferentes puntos iniciales (1,1), (2,2), (2,5), (0,0) y los resultados se mantienen (la sucesión de puntos cambia pero convergen al mismo). En la siguiente tabla, mostraremos los resultados con puntos inicializadores dentro y fuera del conjunto. Para los gráficos, mostramos los resultados para el punto inicial (2,5).

Paso	Resultado de x	Iteraciones	Punto inicial	Tiempo ejecución
Line Search	[-0.23579949 0.08305781]	13	(2,5)	0.01646280288696289
Decreciente	[-0.23579949 0.08305781]	13	(2,5)	0.01772785186767578
Line Search	[-0.23580265 0.08304885]	12	(0,0)	0.015914201736450195
Decreciente	[-0.23580265 0.08304885]	12	(0,0)	0.018887996673583984

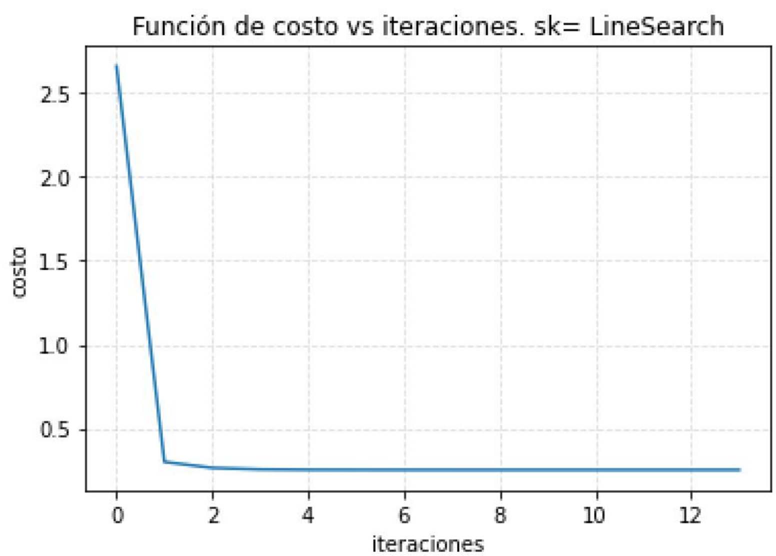
Los gráficos para sk decreciente fueron los siguientes :

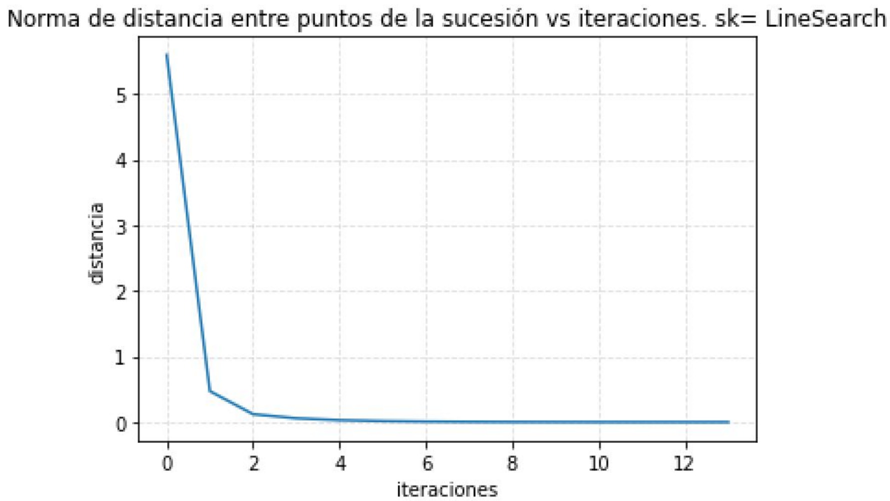


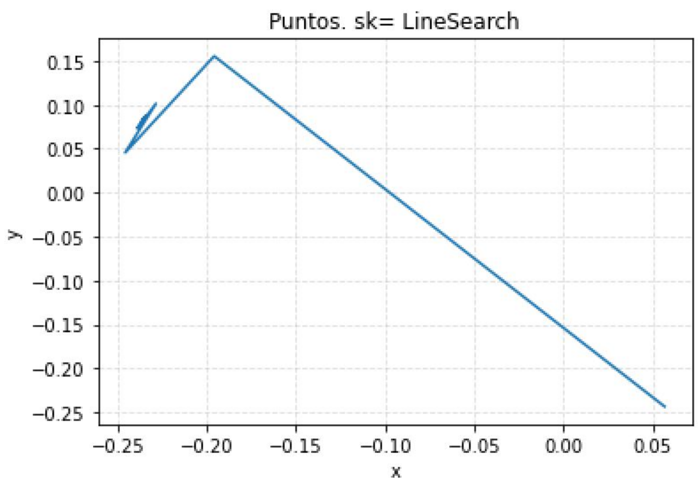




Los gráficos para sk con line search fueron los siguientes:







Análisis de los resultados

Podemos observar que los algoritmos al cambiar de método de elección de sk se comportan de forma muy similar. Se alcanza el mismo resultado, y en la misma cantidad de iteraciones cuando el primer punto de la sucesión es el mismo.

Del gráfico de la sucesión de puntos se puede observar que los puntos elegidos en cada paso son los mismos (o cercanos). Se puede ver también que en el primer paso el acercamiento a la solución final es de mucho mayor magnitud que en el resto de los pasos.

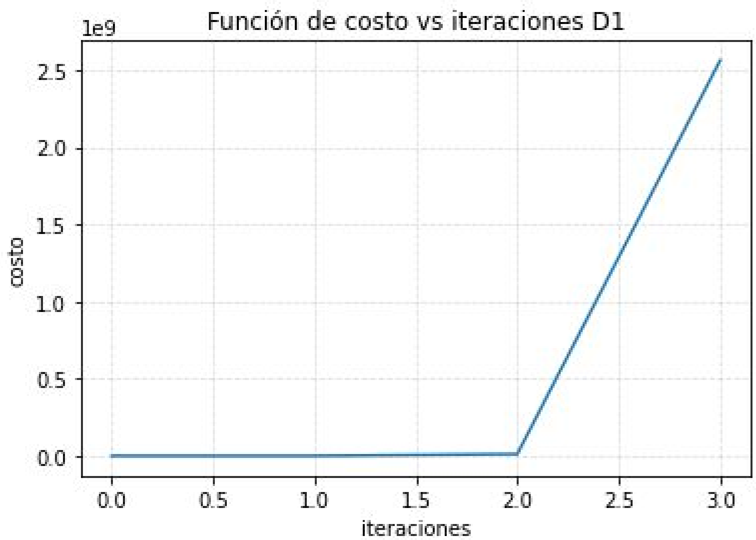
Ambos Algoritmos son muy veloces en encontrar un punto que cumpla con la condición de optimalidad, requieren pocas iteraciones. Comparando las formas de elección de sk, observamos que el método de line search es más performante que la de paso decreciente, de la misma forma que sucedió en el ejercicio 4.

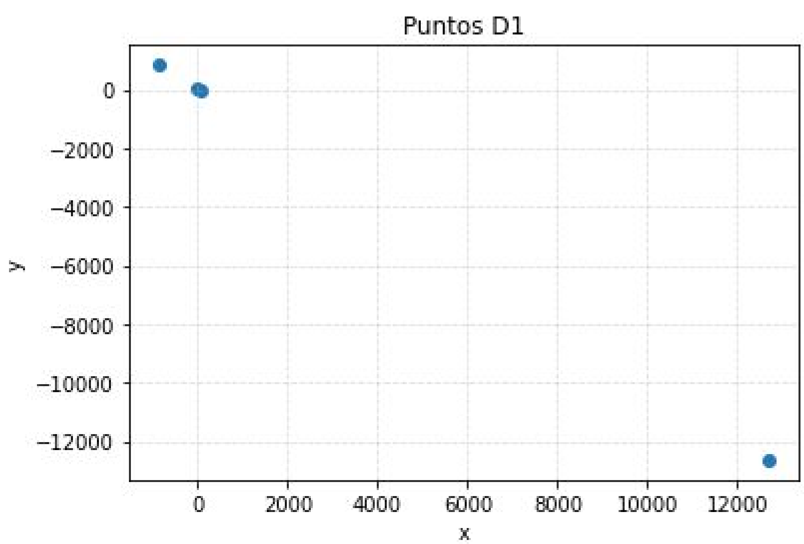
Parte D

No es un problema convexo ya que aunque la función sea la misma, la restricción no es un problema convexo.

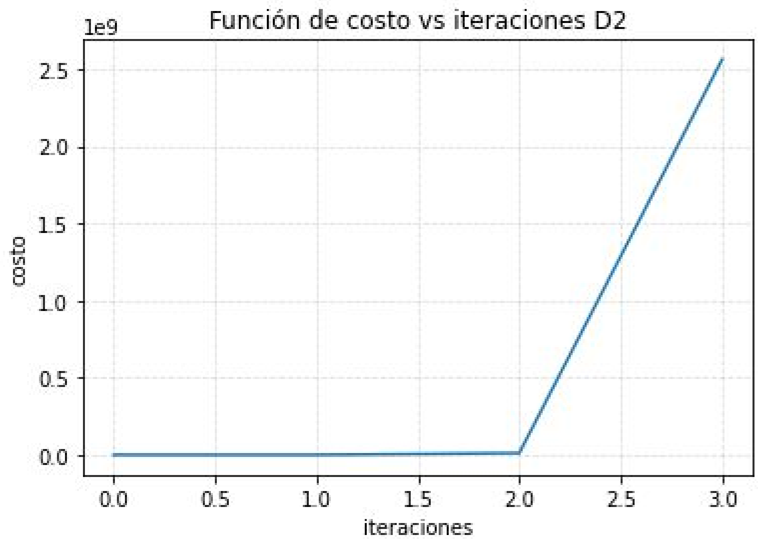
Para este ejercicio se tomaron los puntos iniciales (0,0) y (0.5,0.5). En un principio se tomaron como límite 15 iteraciones. Se observa que la sucesión de puntos divergen y que no se llega a encontrar el valor deseado. Se toman entonces menos iteraciones (4) para realizar y analizar los gráficos.

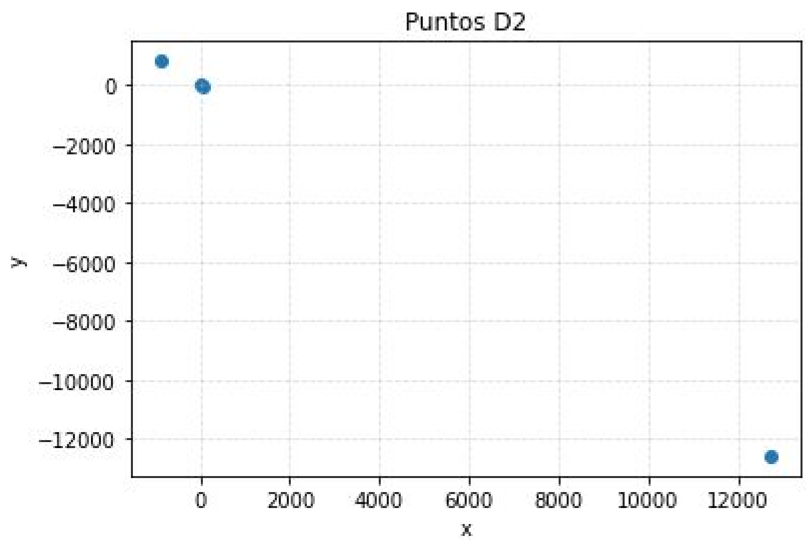
Para el punto inicial (0,0) vemos como la función de costo comienza a crecer, así como los puntos comienzan a alejarse.





Para el punto inicial (0.5,0.5) vemos el mismo comportamiento que para el punto (0,0).


Función de costo vs iteraciones D2
1e9
2.5
2.0
1.5
1.0
0.5
0.0
costo
0.0
0.5
1.0
1.5
2.0
2.5
3.0
iteraciones


Puntos D2
0
−2000
−4000
−6000
−8000
−10000
−12000
y
0
2000
4000
6000
8000
10000
12000
x