OPTIMITZACIÓ - Practical 1

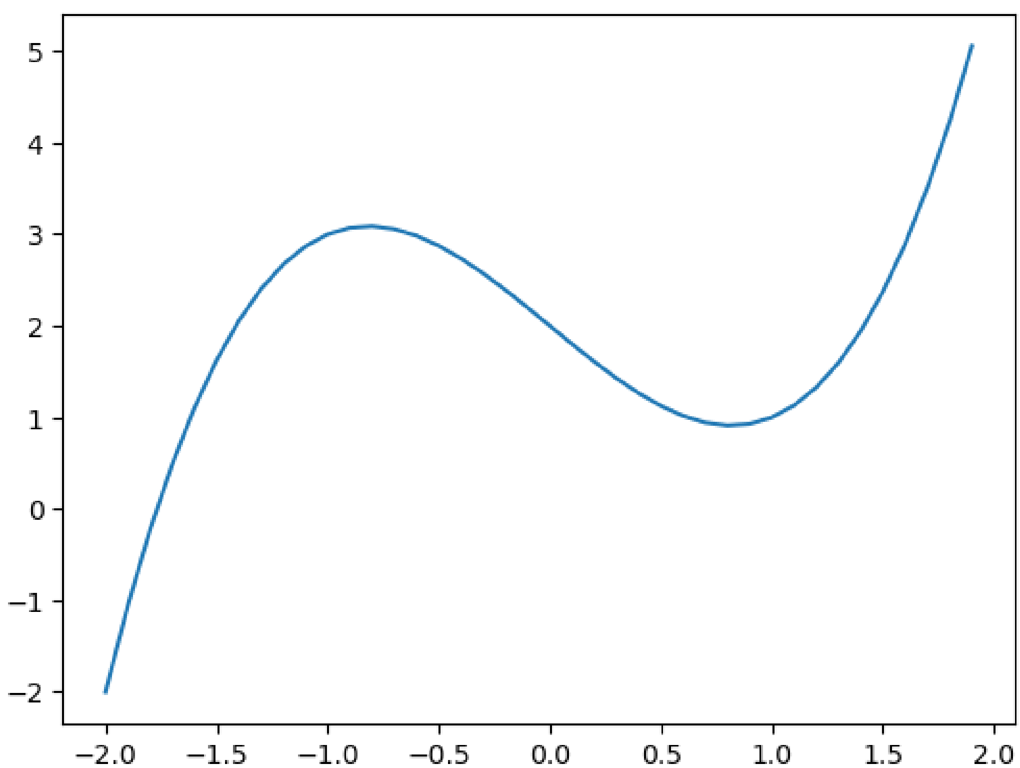
September 30, 2024

Theodoros Lambrou

1 One dimensional case

1. Plot the function $f(x) = x^3 - 2x + 2$ within the range x $[-2, 2]$

[1]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-2, 2, 0.1)
y = x**3 - 2*x + 2;

line, = plt.plot(x,y)
plt.show()
```



2. Compute analytically the points x* that satisfy $f'(x) = 0$

$$\begin{aligned} f'(x) &= 0 \\ \Rightarrow 3x^2 - 2 &= 0 \\ \Rightarrow x^2 &= \frac{2}{3} \\ \Rightarrow x &= \pm\sqrt{\frac{2}{3}} \end{aligned}$$

We can see that there is a maximum between -1.0 and -0.5 and a minimum between 0.5 and 1.0, hence the obtained result is congruent with the plot of the previous point.

3. Checking which of the latter points x* are minima (or a maxima) by using a 2nd order Taylor expansion around point x*

$$f(x^* + d) \approx f(x) + df'(x) + \frac{1}{2}d^2 f''(x^*)$$

In order for x* to be a minimum we need $f''(x)$ to be positive

$f'(x) = 3x^2 - 2$

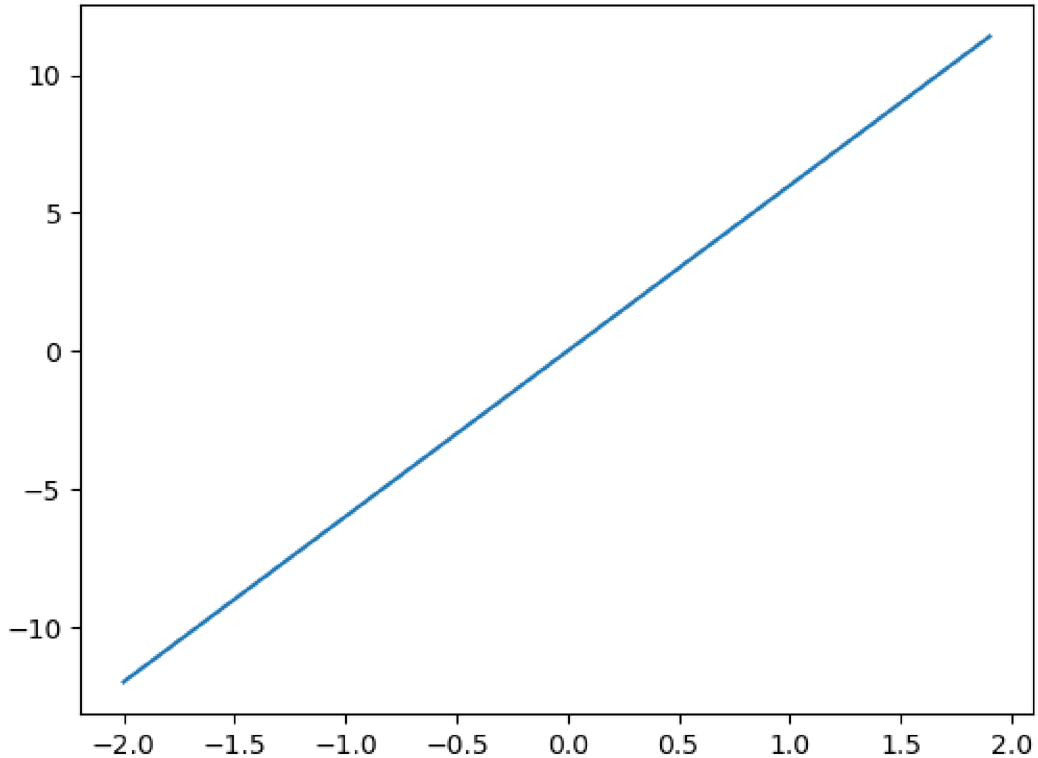
$f''(x) = 6x$

$f''(\sqrt{2/3}) > 0$

Hence, the point $x = \sqrt{2/3}$ is a minimum

4. Plotting $f''(x)$ within the range x $[-2, 2]$

[2]:
```
x = np.arange(-2, 2, 0.1)
y = 6*x

line, = plt.plot(x,y)
plt.show()
```



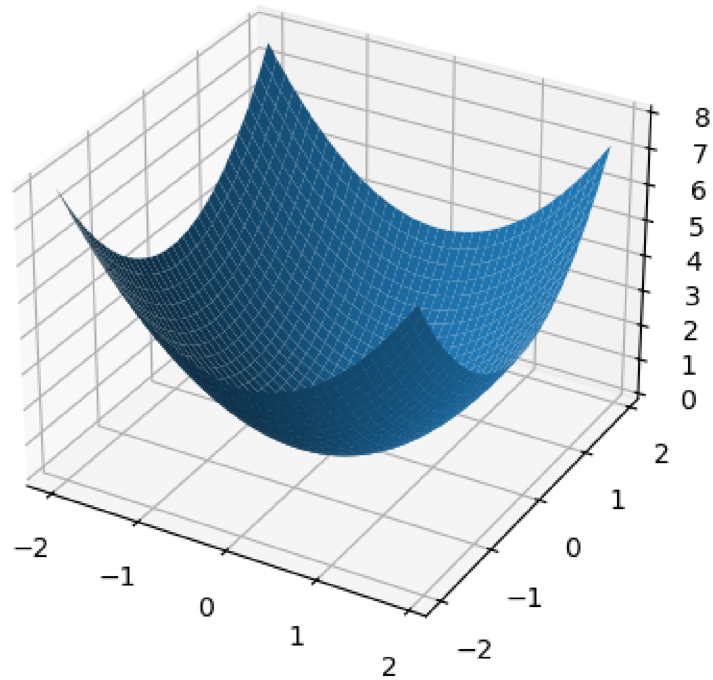
2 Two dimensional case

2.1 A simple two-dimensional function

1. Plot the function $f(x) = x_1^2 + x_2^2$

[3]:
```
def f(x1, x2):
    return x1**2 + x2**2
```

[4]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
x1 = np.arange(-2, 2, 0.1)
x2 = np.arange(-2, 2, 0.1)
x1, x2 = np.meshgrid(x1,x2)
z = f(x1, x2)

s = ax.plot_surface(x1,x2,z)

plt.show()
```



2. Analytically compute the gradient of the function, $\nabla f(x)$, and compute the point x^* at which $\nabla f(x^*) = 0$

$\nabla f(x) = \left(\frac{\delta f(x)}{\delta x_1}, \frac{\delta f(x)}{\delta x_2}\right)^T = (2x_1, 2x_2)^T$

$\nabla f(x) = 0 \quad \Rightarrow \quad (2x_1, 2x_2)^T = (0,0)^T \quad \Rightarrow \quad (x_1, x_2)^T = (0,0)^T$

3. Using the Taylor expansion up to second order and by computing the Hessian matrix, checking if x^* is a minimum or a maximum $\nabla^2 f(x)$ at the point $x = x^*$

$\nabla^2 f(x^*) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$

For a higher dimensional problem the quadratic approximation is convex

We have a maximum if:

$d^T \nabla^2 f(x^*) d < 0, \quad d \neq 0 \quad (4)$ for $d \in \mathbb{R}^2$

We have a minimum if:

$d^T \nabla^2 f(x^*) d > 0, \quad d \neq 0 \quad (3)$ for $d \in \mathbb{R}^2$

In order to calculate the eigenvalues we find the solutions of the characteristic polynomial of $\nabla^2 f(x)$:

$\begin{vmatrix} 2-\lambda & 0 \\ 0 & 2-\lambda \end{vmatrix} = 0 \Rightarrow (2-\lambda)^2 = 0 \Rightarrow \lambda = 2$

We have a minimum at (0,0) as both eigenvalues are positive.

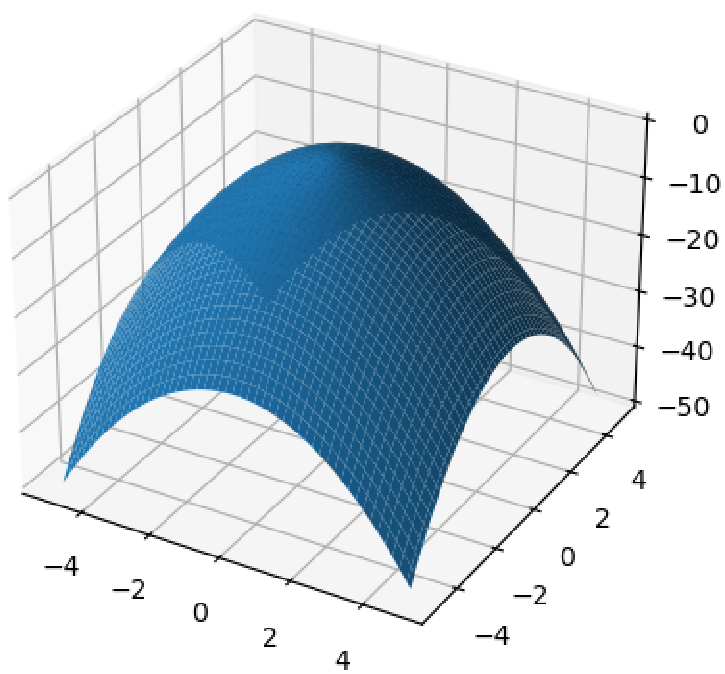
4.(a) Defining the following functions and plotting them:

$$f_A(x) = -x_1^2 - x_2^2 \quad f_B(x) = x_1^2 - x_2^2 \quad f_C(x) = x_1^2$$

[5]:
```
def fA(x1,x2):
    return -x1**2 - x2**2

def fB(x1,x2):
    return x1**2 - x2**2

def fC(x1,x2):
    return x1**2
```

Function A:

[6]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
x1 = np.arange(-5, 5, 0.1)
x2 = np.arange(-5, 5, 0.1)
x1, x2 = np.meshgrid(x1, x2)
z = fA(x1, x2)

surf = ax.plot_surface(x1, x2, z)
plt.show()
```



The maximum is at (0,0)

Verifying analytically

$\nabla f_A(x) = (-2x_1, -2x_2) = (0,0) \Rightarrow x_1 = 0, \quad x_2 = 0$

Hessian

$\nabla^2 f_A(x) = \begin{pmatrix} \frac{\delta^2 f_A}{\delta x_1^2} & \frac{\delta^2 f_A}{\delta x_1 \delta x_2} \\ \frac{\delta^2 f_A}{\delta x_2 \delta x_1} & \frac{\delta^2 f_A}{\delta x_2^2} \end{pmatrix} = \begin{pmatrix} -2 & 0 \\ 0 & -2 \end{pmatrix}$

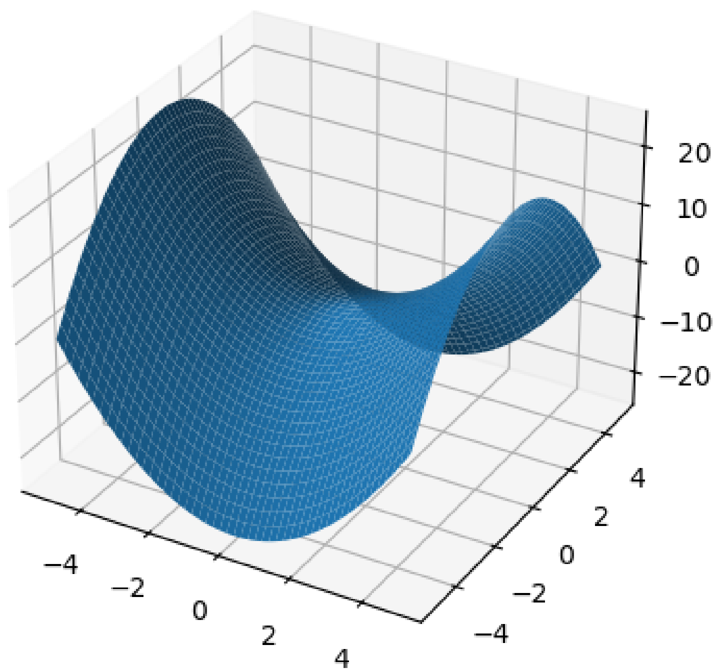
[7]:
```
h_fa = np.matrix('-2 0; 0 -2')
np.linalg.eigvals(h_fa)
```

[7]:
```
array([-2., -2.])
```

There is only one negative eigenvalue in $(x_1, x_2) = (0,0)$ hence there is a maximum at that point

Function B:

[8]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
x1 = np.arange(-5, 5, 0.1)
x2 = np.arange(-5, 5, 0.1)
x1, x2 = np.meshgrid(x1, x2)
z = fB(x1, x2)

surf = ax.plot_surface(x1, x2, z)
plt.show()
```



Seems like the gradient is 0 when $x_1 = x_2 = 0$

$\nabla f_B(x) = (2x_1, -2x_2) = (0,0) \Rightarrow x_1 = 0, \quad x_2 = 0$

Hessian

$$\nabla^2 f_B(x) = \begin{pmatrix} \frac{\delta^2 f_B}{\delta x_1^2} & \frac{\delta^2 f_B}{\delta x_1 \delta x_2} \\ \frac{\delta^2 f_B}{\delta x_2 \delta x_1} & \frac{\delta^2 f_B}{\delta x_2^2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$$

[9]:
```
h_fb = np.matrix('2 0; 0 -2')
np.linalg.eigvals(h_fb)
```

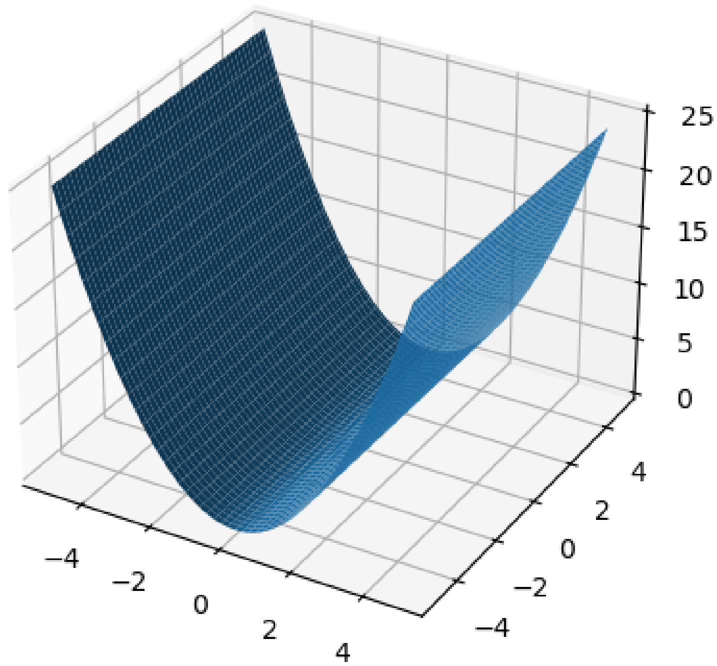
[9]:
```
array([ 2., -2.])
```

The eigenvalues in $(x_1, x_2) = (0, 0)$ are positive/negative hence there is no minimum or maximum at this point.

Function C:

[10]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
x1 = np.arange(-5, 5, 0.1)
x2 = np.arange(-5, 5, 0.1)
x1, x2 = np.meshgrid(x1, x2)
z = fC(x1, x2)
s = ax.plot_surface(x1, x2, z)
```

```
plt.show()
```



The gradient is 0 in the whole line of x_1:

$\nabla f_C(x) = (2x_1, 0) = (0, 0) \Rightarrow x_1 = 0$

Hessian

$\nabla^2 f_C(x) = \begin{pmatrix} \frac{\delta^2 f_C}{\delta x_1^2} & \frac{\delta^2 f_C}{\delta x_1 \delta x_2} \\ \frac{\delta^2 f_C}{\delta x_2 \delta x_1} & \frac{\delta^2 f_C}{\delta x_2^2} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$

[11]:
```
h_fc = np.matrix('2 0; 0 0')
np.linalg.eigvals(h_fc)
```

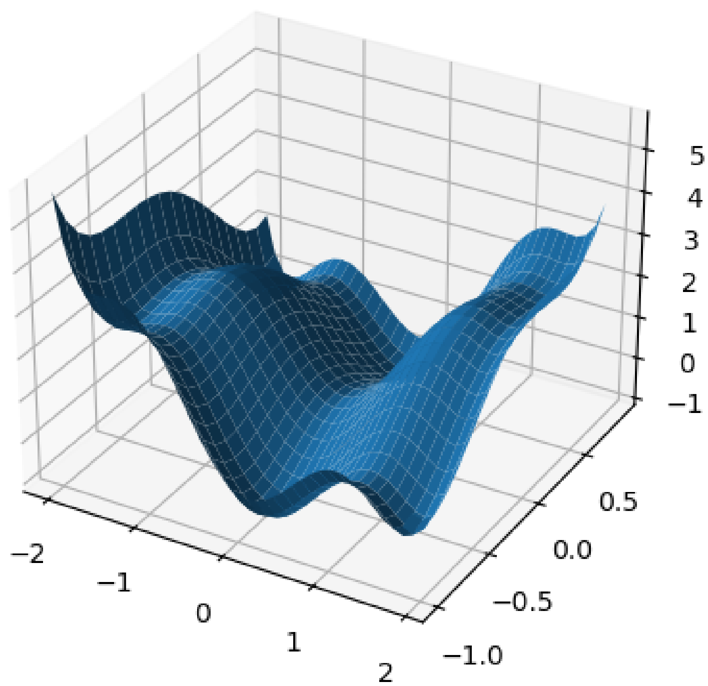
[11]:
```
array([2., 0.])
```

The eigenvalues are positive at $(x_1, x_2) = (0, 0)$ hence there is a minimum at this point

2.2 A two dimensional function with multiple minima

1. Plot the following function within the range x1 $[-2,\ 2]$ and x2 $[-1,\ 1]$, $f(x_1, x_2) = x_1^2\left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4\right) + x_1x_2 + x_2^2\left(-4 + 4x_2^2\right)$ and observe where the minima (and maxima) may be.

[12]:
```
def f(x1, x2):
    return x1**2*(4 - 2.1*x1**2 + 1/3*x1**4) + x1*x2 + x2**2*(-4 + 4*x2**2)

fig = plt.figure()
ax = plt.axes(projection='3d')
x1 = np.arange(-2, 2, 0.1)
x2 = np.arange(-1, 1, 0.1)
x1, x2 = np.meshgrid(x1, x2)
z = f(x1, x2)

s = ax.plot_surface(x1, x2, z)
plt.show()
```



There is a local maximum close to $(x_1, x_2) = (0, -1)$ and a local minimum at $(x_1, x_2) = (-0.5, 0)$

2. Analytically compute the gradient $\nabla f(x)$:

$\nabla f(x) = (8x_1 - 8.4x_1^3 + 2x_1^5 + x_2, x_1 - 8x_2 + 16x_2^3)$

3. Numerically compute an approximation of the points x^* at which $\nabla f(x^*) = 0$.

(a) Evaluating $\|\nabla f(x)\|^2$ at the previous range using a step of 0.005

[13]:
```
import sympy as sp
import numpy as np
```

```
x1, x2 = sp.symbols('x1 x2')
f = x1**2*(4 - 2.1*x1**2 + 1/3*x1**4) + x1*x2 + x2**2*(-4 + 4*x2**2)
gradient_f = [sp.diff(f, var).simplify() for var in (x1, x2)]

def eval_gradient_norm(gradient_f, c1, c2):

  valor_gradient_f = np.array([gradient.subs({x1: c1, x2: c2}) for gradient in␣
 ↪gradient_f])

  norm = np.sum(valor_gradient_f**2)

  return norm


eval_gradient_norm(gradient_f, -1, -2)

x1_values = np.arange(-2, 2, 0.005)
x2_values = np.arange(-1, 1, 0.005)

A = np.zeros((len(x1_values), len(x2_values)))

for i in range(len(x1_values)):
  for j in range(len(x2_values)):
    A[i, j] = eval_gradient_norm(gradient_f, x1_values[i], x2_values[j])

A
```

[13]:
```
array([[290.44      , 286.36550724, 282.41597569, …, 168.87039728,
         170.88970369, 173.01192324],
        [281.25287051, 277.18369845, 273.23946373, …, 161.16045293,
         163.18503241, 165.31254879],
        [272.43049364, 268.36658803, 264.42759587, …, 153.79371809,
         155.82351638, 157.95627532],
        …,
        [153.17515279, 150.93127616, 148.79332839, …, 252.15638055,
         255.96833402, 259.90213734],
        [160.19493924, 157.95627532, 155.82351638, …, 260.61047734,
         264.42759587, 268.36658803],
        [167.54594626, 165.31254879, 163.18503241, …, 269.41712639,
         273.23946373, 277.18369845]])
```