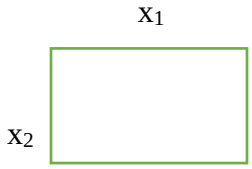
Technical Report - Project A - Optimization Techniques

Mahan RajaeiGolsefidi

School of Informatics, Aristotle University of Thessaloniki

Spring 2022

Question 1. Part a.



$x_1, x_2 \in \mathcal{R}$

$x_1 > 0 \ and \ x_2 > 0$

$area: S = x_1 x_2$

$2(x_1 + x_2) = L \Longrightarrow x_2 = (\frac{L}{2} - x_1)$

$S = \left(\frac{L}{2} x_1 - {x_1}^2\right)$

$maximize\ S = \left(\frac{L}{2} x_1 - {x_1}^2\right)$

General optimization formulation:

$$\min_{x_1 \in \mathcal{R}} f = \left({x_1}^2 - \frac{L}{2} x_1\right)$$

$$s.t. \quad x_1 > 0$$

$$\frac{L}{2} - x_1 > 0$$

$\frac{df}{dx_1} = 2x_1 - \frac{L}{2}$

Necessary condition: $\frac{df}{dx_1} = 0 \Rightarrow x_1 = \frac{L}{4} \Rightarrow x_2 = \frac{L}{2} - \frac{L}{4} = \frac{L}{4}$

$\frac{d^2 f}{d{x_1}^2} = 2 > 0 \Rightarrow (x_1, x_2) = \left(\frac{L}{4}, \frac{L}{4}\right)$ $is\ the\ Minimizer\ (Maximizer\ for\ the\ original\ S).$

Question 1. Part b.

Let's assume that the new antenna is installed at a location with the following coordinates: (x, y)

The coordinates of the locations of the 4 customers are:

$(x_1, y_1) = (5, 10)$, $(x_2, y_2) = (10, 5)$, $(x_3, y_3) = (0, 12)$, and $(x_4, y_4) = (12, 0)$

The coordinates of the locations of the 2 already existing antennas are:

$(x_5, y_5) = (-5, 10)$ and $(x_6, y_6) = (5, 0)$

The distances between the new antenna and the customers are:

$$d_i = ((x - x_i)^2 + (y - y_i)^2)^{\frac{1}{2}} \qquad i = 1, 2, 3, 4$$

The distances between the new antenna and the already existing antennas are:

$$d_j = ((x - x_j)^2 + (y - y_j)^2)^{\frac{1}{2}} \qquad j = 5, 6$$

The weights (priorities) of the customers are:

$w_1 = 200$, $w_2 = 150$, $w_3 = 200$, $w_4 = 300$

$$\min_{x,\ y \in \mathcal{R}} \sum_{i=1}^{4} w_i d_i$$

$$s.t.\ d_j < 10 \ \ for\ j = 5, 6$$

$$\min_{x,\ y \in \mathcal{R}} 200((x-5)^2 + (y-10)^2)^{\frac{1}{2}} + 150((x-10)^2 + (y-5)^2)^{\frac{1}{2}} + 200(x^2 + (y-12)^2)^{\frac{1}{2}} + 300((x-12)^2 + y^2)^{\frac{1}{2}}$$

$$s.t. \quad ((x+5)^2 + (y-10)^2)^{\frac{1}{2}} < 10$$

$$((x-5)^2 + y^2)^{\frac{1}{2}} < 10$$

Question 2. (Implementation in Python)

Function #1:

$$f(x_1, x_2) = 8{x_1}^2 + 3x_1x_2 + 7{x_2}^2 - 25x_1 + 31x_2$$

Figure 1 shows the results of running the algorithm to solve the optimization problem for the first function. A local minimizer is found. As Figure 2 and 3 show, this point is also a global minimizer.

```
In [2]: runfile('/home/ryan/Desktop/0_0_Project_A/Test/Project_A_Q2.py', wdir='/home/ryan/Desktop/0_0_Project_A/Test')

Function:
    2                         2
8·x₁  + 3·x₁·x₂ - 25·x₁ + 7·x₂  + 31·x₂

Gradient:
⎡16·x₁ + 3·x₂ - 25⎤
⎢                 ⎥
⎣3·x₁ + 14·x₂ + 31⎦

List of Points satisfying the necessary condition:
[{x1: 443/215, x2: -571/215}]

General form of the Hessian Matrix:
⎡16  3 ⎤
⎢      ⎥
⎣3   14⎦


***** Methods: Minors of the Hessian Matrix & Eigenvalues of the Hessian Matrix *****


&&& Point number 1 that satisfies the necessary condition:

(443/215, -571/215)

The Hessian matrix at point (443/215, -571/215):

⎡16  3 ⎤
⎢      ⎥
⎣3   14⎦

### Method #1: Minors:
Delta_1 = 16
Delta_2 = 215
Point (443/215, -571/215) is a LOCAL MINIMIZER and the value of the function at this point is: -66.92093023255813

### Method #2: Eigenvalues:
List of eigenvalues of the Hessian Matrix:
[sqrt(10) + 15, 15 - sqrt(10)]
Point (443/215, -571/215) is a LOCAL MINIMIZER and the value of the function at this point is: -66.92093023255813

--------------------------------------------
```

Figure 1

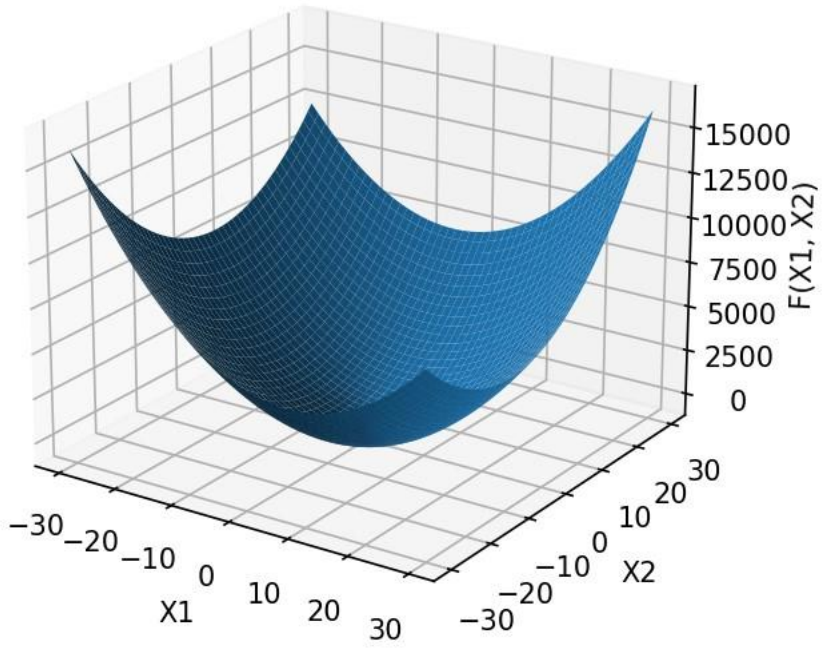


Figure 2

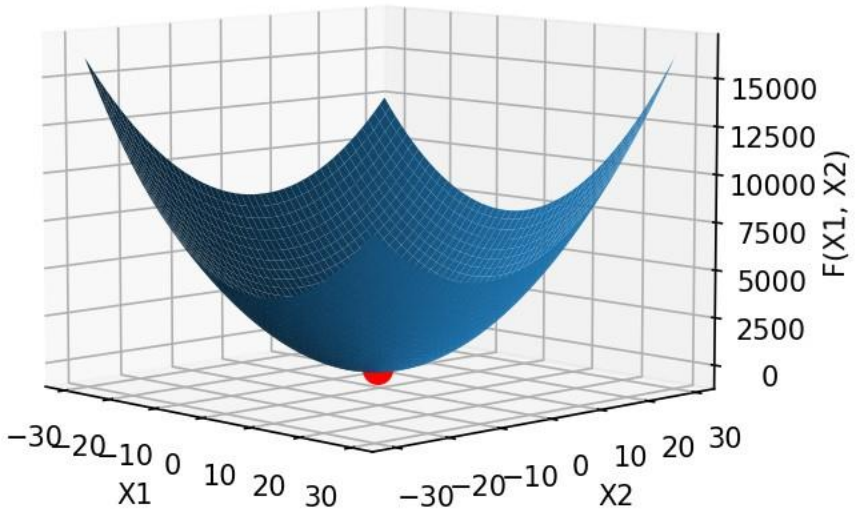


Figure 3

Function #2:

$$f(x_1, x_2) = {x_1}^2 - 2x_1{x_2}^2 + {x_2}^4 - {x_2}^5$$

Figure 4 shows the results of running the algorithm to solve the optimization problem for the first function. A local optimizer was not found. As Figure 5 and 6 show, the point that satisfies the necessary condition does not qualify as a local or global optimizer.

```
In [3]: runfile('/home/ryan/Desktop/0_0_Project_A/Test/Project_A_Q2.py', wdir='/home/ryan/Desktop/0_0_Project_A/Test')

Function:
  2          2     5     4
x₁  - 2·x₁·x₂  - x₂  + x₂

Gradient:
⎡                  2      ⎤
⎢       2·x₁ - 2·x₂       ⎥
⎢                         ⎥
⎢                4       3⎥
⎣-4·x₁·x₂ - 5·x₂  + 4·x₂ ⎦

List of Points satisfying the necessary condition:
[{x1: 0, x2: 0}]

General form of the Hessian Matrix:
⎡  2             -4·x₂          ⎤
⎢                               ⎥
⎢                    3        2⎥
⎣-4·x₂  -4·x₁ - 20·x₂  + 12·x₂ ⎦


***** Methods: Minors of the Hessian Matrix & Eigenvalues of the Hessian Matrix *****


&&& Point number 1 that satisfies the necessary condition:

(0, 0)

The Hessian matrix at point (0, 0):

⎡2  0⎤
⎢    ⎥
⎣0  0⎦

### Method #1: Minors:
Delta_1 = 2
Delta_2 = 0
Point (0, 0) is NOT a local optimizer

### Method #2: Eigenvalues:
List of eigenvalues of the Hessian Matrix:
[2, 0]
Point (0, 0) is NOT a local optimizer

---------------------------------------------
```

Figure 4

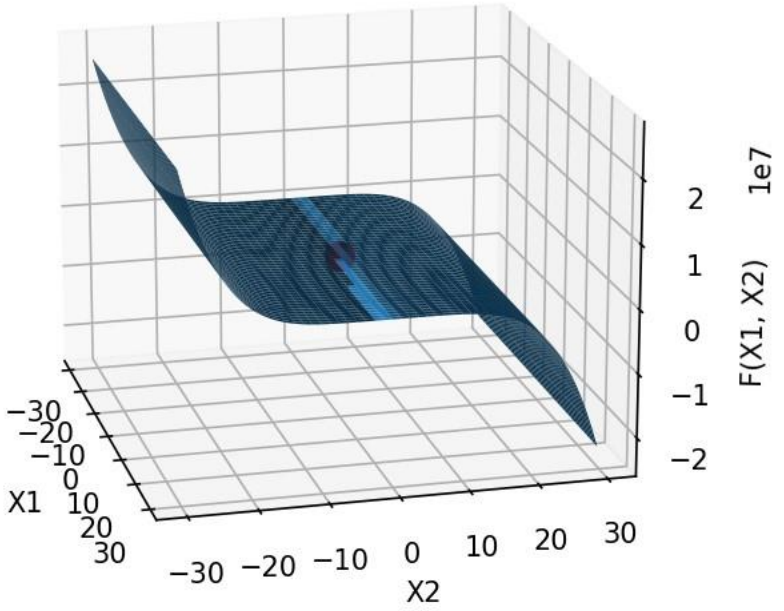


Figure 5

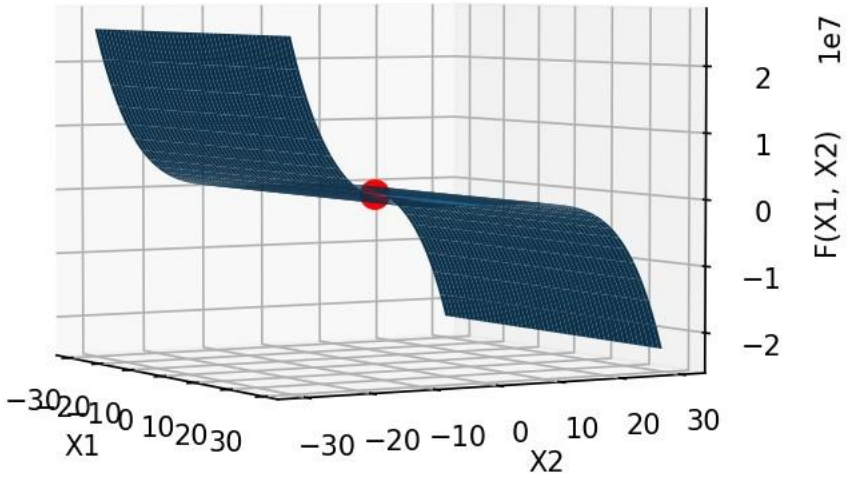


Figure 6

Function #3:

$$f(x_1, x_2) = 100(x_2 - {x_1}^2)^2 + (1 - x_1)^2$$

Figure 7 shows the results of running the algorithm to solve the optimization problem for the first function. A local minimizer is found. As Figure 8 and 9 show, this point is also a global minimizer.

```
In [4]: runfile('/home/ryan/Desktop/0_0_Project_A/Test/Project_A_Q2.py', wdir='/home/ryan/Desktop/0_0_Project_A/Test')

Function:
                          2
       2       ⎛    2     ⎞
(1 - x₁)  + 100⋅⎝- x₁  + x₂⎠

Gradient:
⎡          ⎛    2     ⎞           ⎤
⎢- 400⋅x₁⋅⎝- x₁  + x₂⎠ + 2⋅x₁ - 2⎥
⎢                                 ⎥
⎢               2                 ⎥
⎣       - 200⋅x₁  + 200⋅x₂        ⎦

List of Points satisfying the necessary condition:
[{x1: 1, x2: 1}]

General form of the Hessian Matrix:
⎡       2                        ⎤
⎢1200⋅x₁  - 400⋅x₂ + 2  -400⋅x₁  ⎥
⎢                                ⎥
⎣       -400⋅x₁           200    ⎦


***** Methods: Minors of the Hessian Matrix & Eigenvalues of the Hessian Matrix *****


&&& Point number 1 that satisfies the necessary condition:

(1, 1)

The Hessian matrix at point (1, 1):

⎡802   -400⎤
⎢          ⎥
⎣-400  200 ⎦

### Method #1: Minors:
Delta_1 = 802
Delta_2 = 400
Point (1, 1) is a LOCAL MINIMIZER and the value of the function at this point is: 0.0

### Method #2: Eigenvalues:
List of eigenvalues of the Hessian Matrix:
[sqrt(250601) + 501, 501 - sqrt(250601)]
Point (1, 1) is a LOCAL MINIMIZER and the value of the function at this point is: 0.0

--------------------------------------------
```

Figure 7

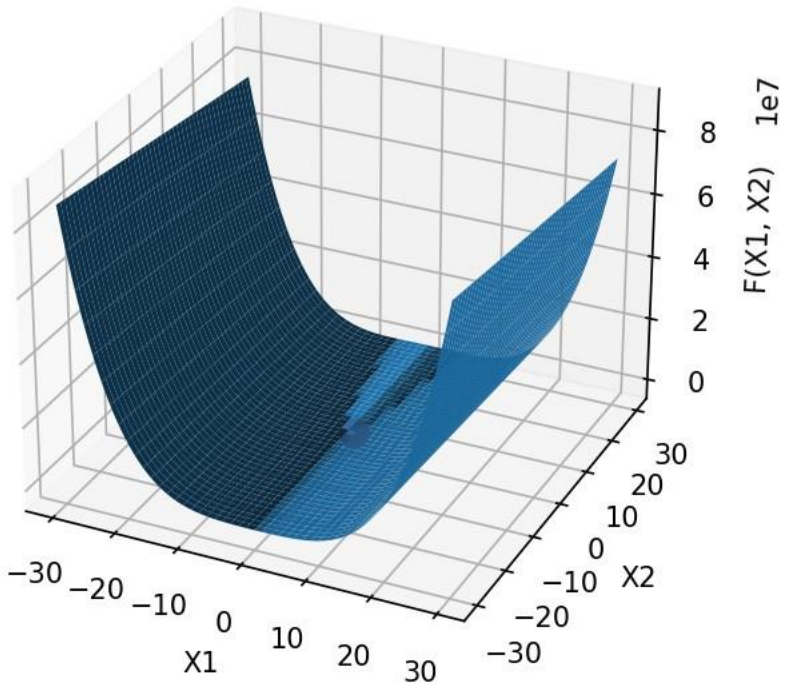


Figure 8

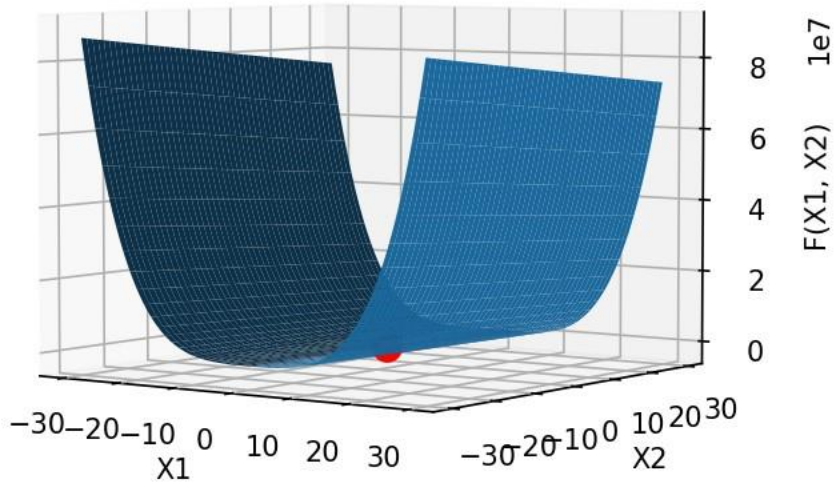


Figure 9

Question 3. (Implementation in Python)

The algorithm is implemented with two different modes. If the user defines a value for alpha, then the algorithm will use that constant value in all of the iterations. Otherwise, alpha is calculated in each iteration based on the following equation:

$$\alpha_k = \frac{\|\nabla f(\boldsymbol{X}_k)\|^2}{\nabla f(\boldsymbol{X}_k)^T Q \nabla f(\boldsymbol{X}_k)}$$

Function #1:

$$f(\boldsymbol{X}) = \frac{1}{2}\boldsymbol{X}^T Q \boldsymbol{X} - \boldsymbol{b}^T \boldsymbol{X} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 25 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

For this function, the algorithm converges to a local minimizer. The approximate values are as below:

$$\boldsymbol{X}^* = (-1, -0.2, -0.04) \quad and \quad f(\boldsymbol{X}^*) = -0.62$$

In cases where the value of alpha is calculated by the algorithm at each iteration, the speed of convergence is higher. The supplementary Microsoft Excel files contain detailed and comprehensive information regarding all of the steps taken to solve this problem for different initial points and different scenarios regarding the value of alpha. Figures 10 to 15 show the points selected at each step. The "red" point indicates the initial point.

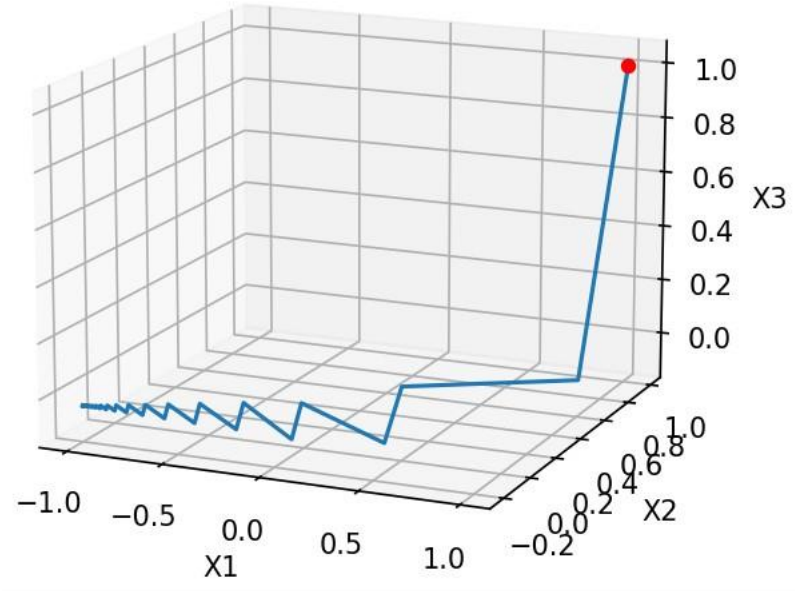


Figure 10. Initial Point: (1, 1, 1) and the value of alpha calculated by the algorithm at each step. (Converges in 82 steps)

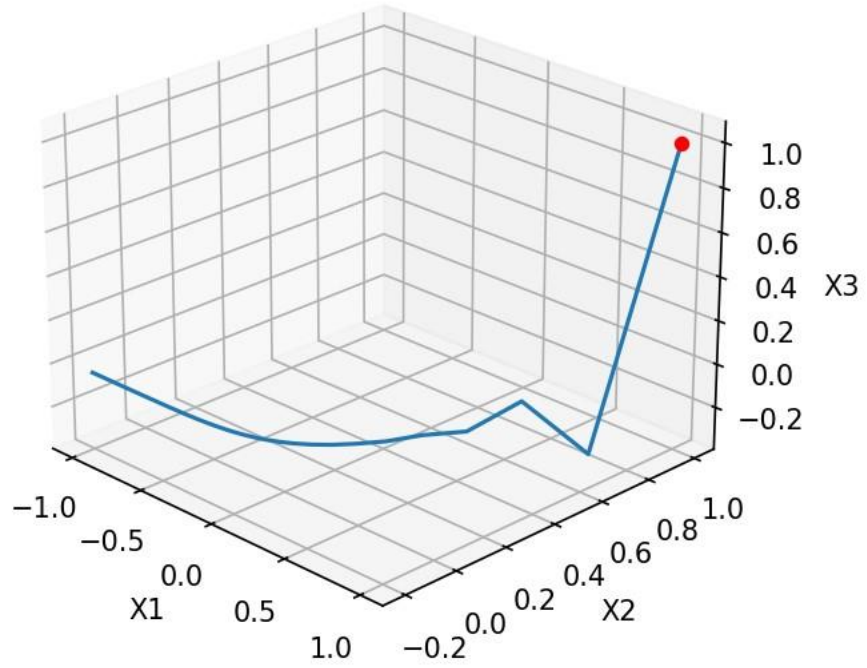


Figure 11. Initial Point: (1, 1, 1) and the value of alpha is set to 0.05 by user. (Converges in 239 steps)

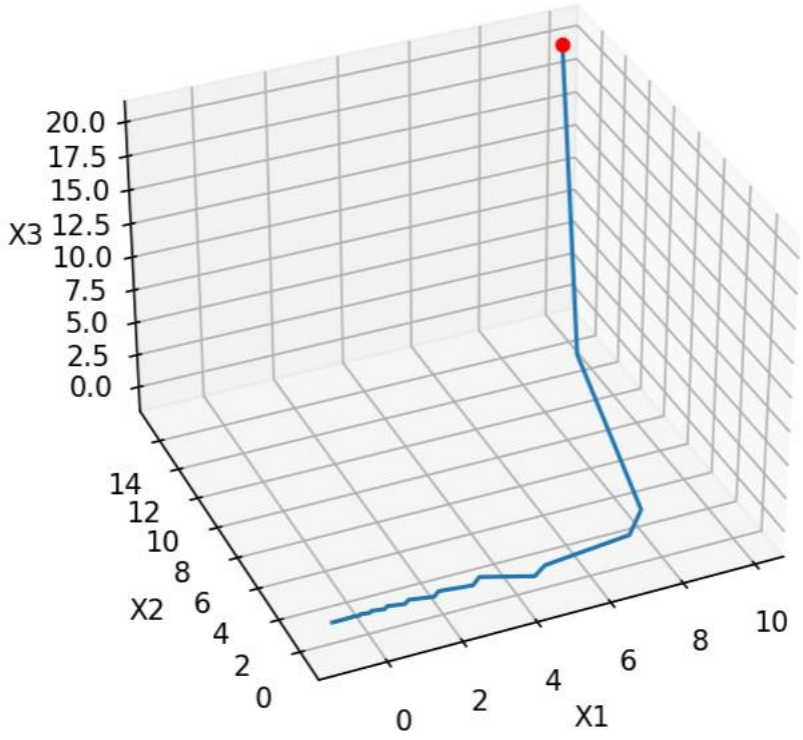


Figure 12. Initial Point: (10, 15, 20) and the value of alpha calculated by the algorithm at each step. (Converges in 86 steps)

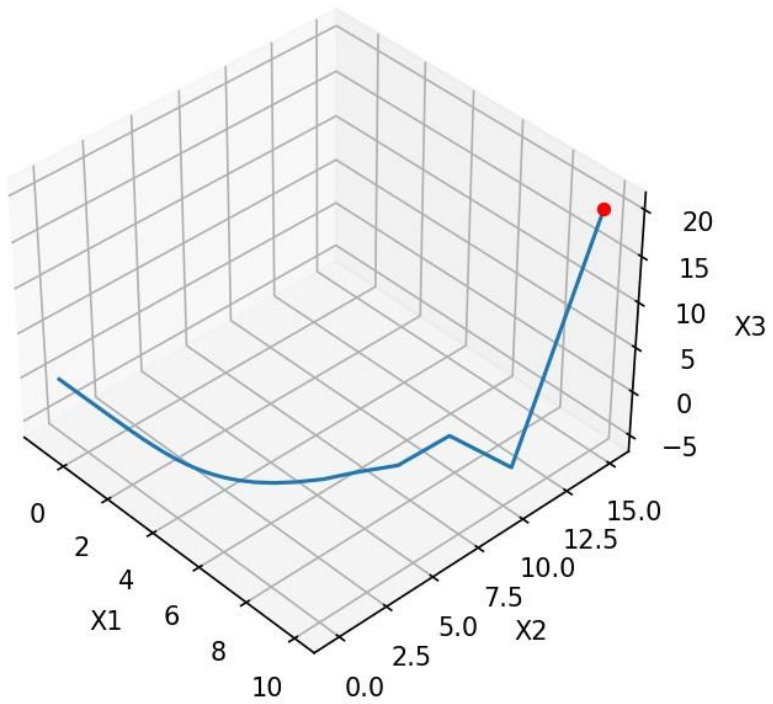


Figure 13. Initial Point: (10, 15, 20) and the value of alpha is set to 0.05 by user. (Converges in 273 steps)

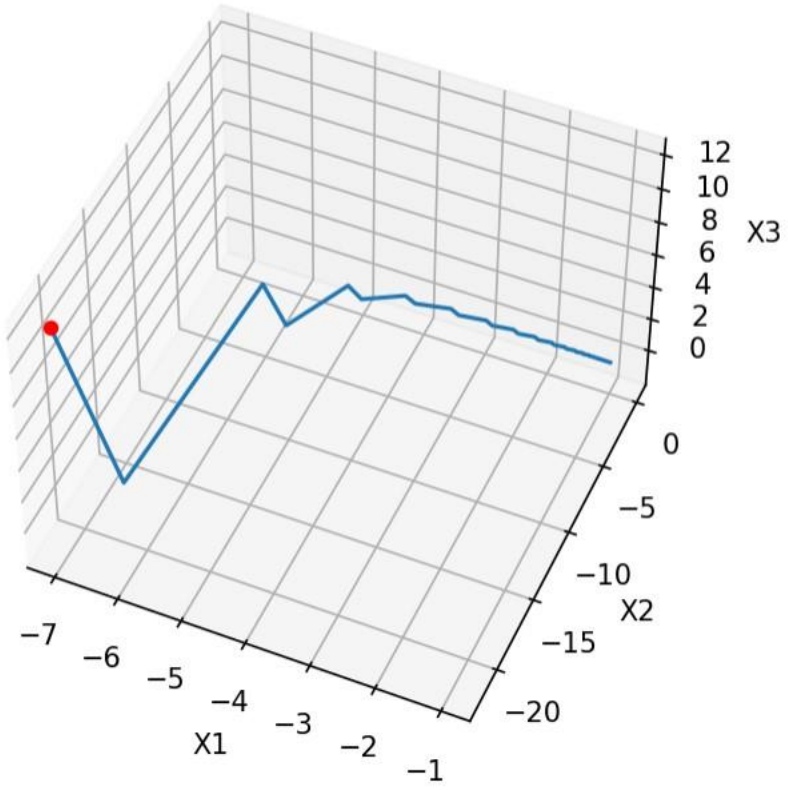


Figure 14. Initial Point: (-7, -22, 12) and the value of alpha calculated by the algorithm at each step. (Converges in 110 steps)

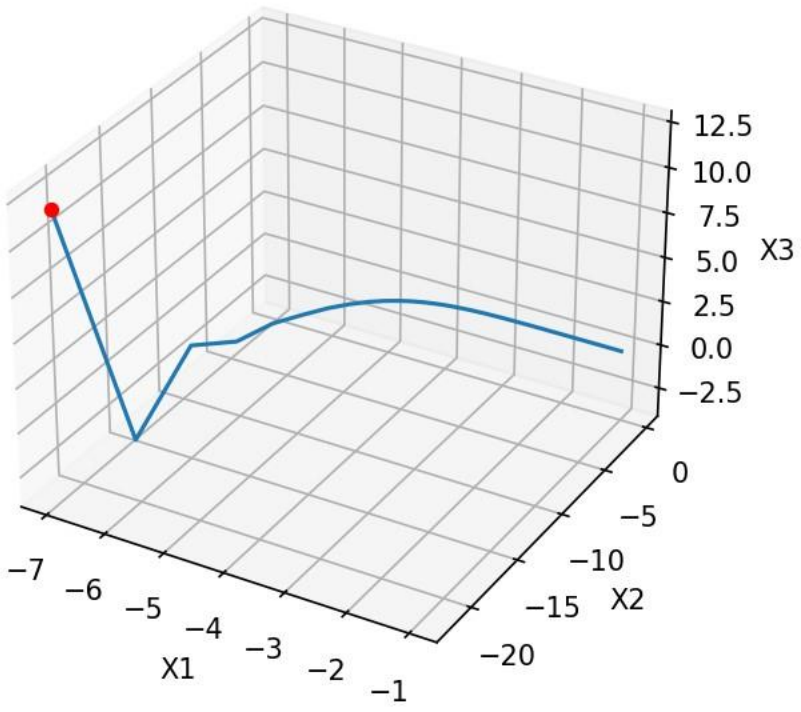


Figure 15. Initial Point: (-7, -22, 12) and the value of alpha is set to 0.05 by user. (Converges in 261 steps)

Function #2:

$$f(\boldsymbol{X}) = \frac{1}{2}\boldsymbol{X}^T Q \boldsymbol{X} - \boldsymbol{b}^T \boldsymbol{X} \qquad Q = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 25 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

For this function, the algorithm does not necessarily converge to a local minimizer. Since the Hessian matrix of the function is **not positive-definite**, the convergence is not guaranteed. After trying many initial points, I concluded that the algorithm converges only if the first element of the initial point is 1. Initial points with coordinates of X = (1, x_2, x_3) will cause the algorithm to converge to the following result and even slight variations in x_1 (for example: x_1 = 1.1) will cause the algorithm to diverge.

$$\boldsymbol{X}^* = (1, -0.2, -0.04) \quad and \quad f(\boldsymbol{X}^*) = 0.38$$

The supplementary Microsoft Excel files contain detailed and comprehensive information regarding all of the steps. Figures 16 to 18 show the step points in each iteration for some of the trials. Red points indicate the initial point.

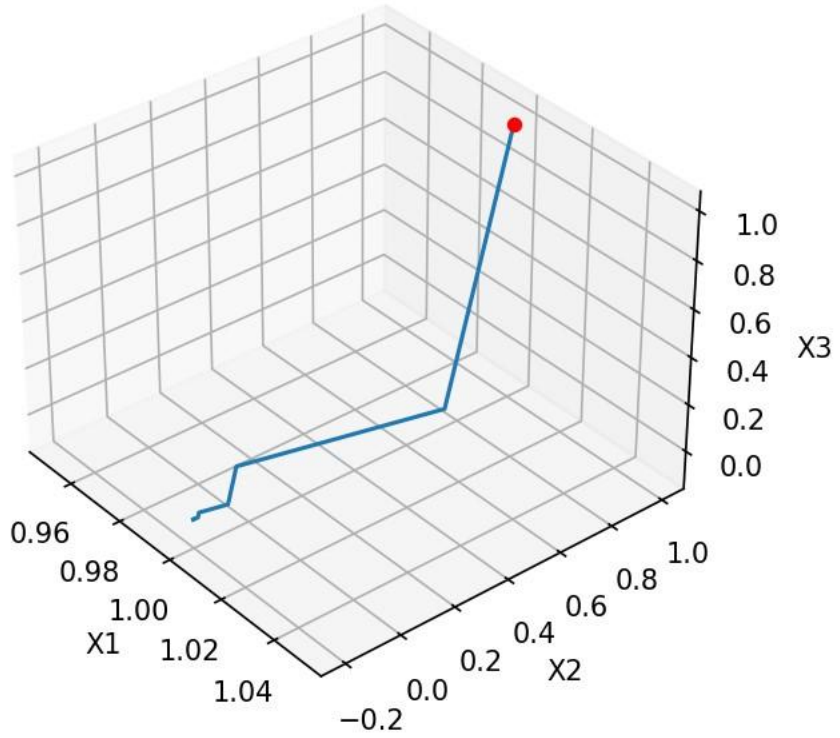


Figure 16. Initial Point: (1, 1, 1) and the value of alpha is set to 0.05 by user. (Converges in 48 steps)

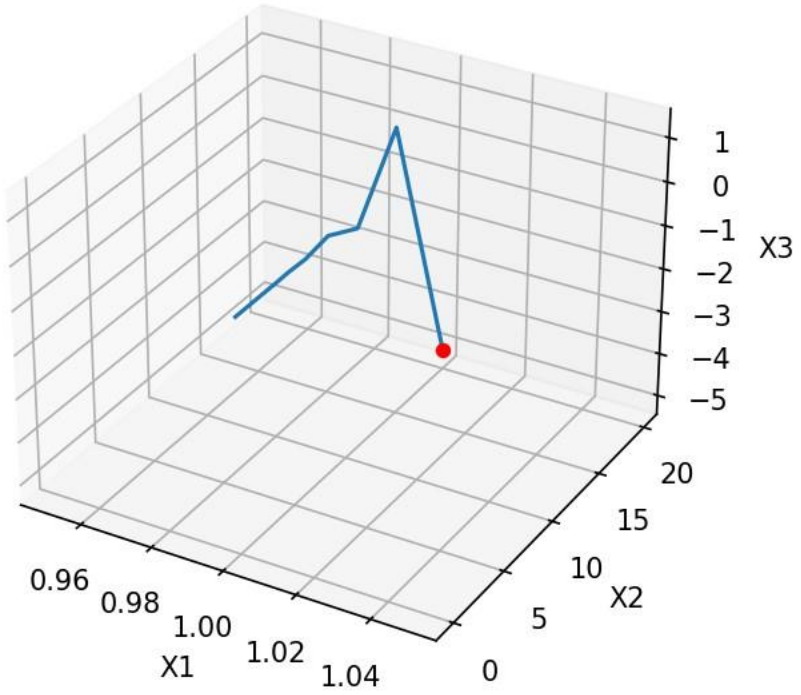


Figure 17. Initial Point: (1, 20, -5) and the value of alpha is set to 0.05 by user. (Converges in 69 steps)

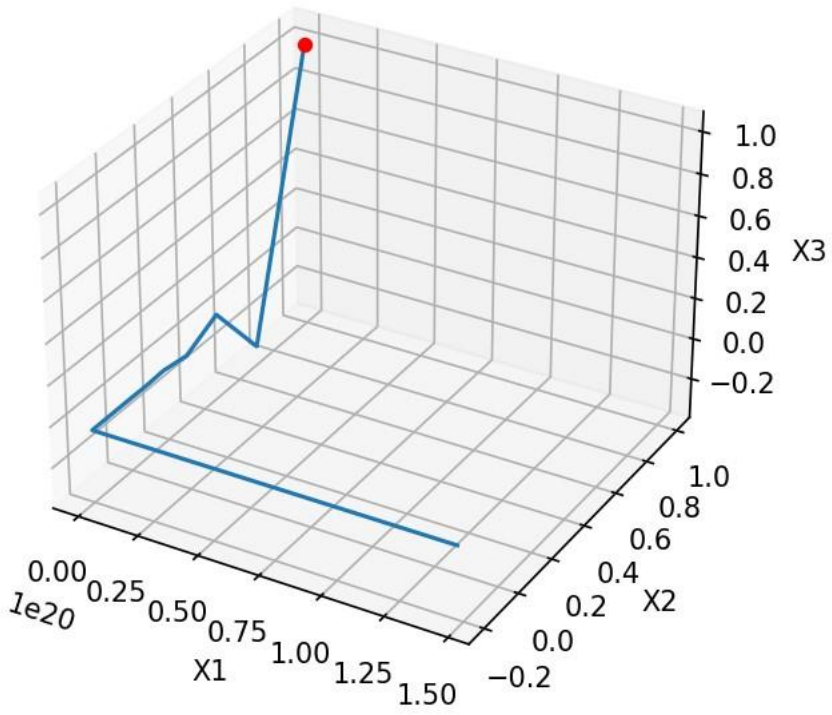


Figure 18. Initial Point: (1.1, 1, 1) and the value of alpha is set to 0.05 by user. (Does NOT converge)