Università della Svizzera italiana | Institute of Computing CI

Optimization Methods **Spring 2024**

Student: Birindelli Leonardo **Discussed with:** Morales Jeferson

---

First assigment

Due date: Tuesday, 19 March 2024, 12:00 AM

Exercise 1

Consider the following vector-valued function $f : \mathbb{R}^2 \to \mathbb{R}$:

$$f(x_1, x_2) = 200(x_2 - x_1^2)^2 + (1 - x_1)^2$$

1. Compute the gradient $\nabla f : \mathbb{R}^2 \to \mathbb{R}^2$ and the Hessian $H_f : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$, which are respectively defined as:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix}^T \quad Hf = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}.$$

   First of all , I started to compute the function $f(x_1, x_2)$ so as subsequently I am able to calculate the partial derivatives of variables x_1 and x_2 :

   $f(x_1, x_2) = 200(x_2 - x_1^2)^2 + (1 - x_1)^2$

   $= 200(x_2^2 + x_1^4 - 2x_1^2 x_2) + 1 + x_1^2 - 2x_1$

   $= 200x_2^2 + 200x_1^4 - 400x_1^2 x_2 + 1 + x_1^2 - 2x_1$

   Then , I aimed to compute the $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$ respectively :

   $\frac{\partial f}{\partial x_1} = 200 \cdot 4x_1^3 - 400 \cdot 2x_1 x_2 + 2x_1 - 2$

   $= 800x_1^3 - 800x_1 x_2 + 2x_1 - 2$

   $\frac{\partial f}{\partial x_2} = 200 \cdot 2x_2 - 400x_1^2$

   $= 400x_2 - 400x_1^2$

   Hence, the gradient of the function is:

$$\nabla f = \begin{bmatrix} 800x_1^3 - 800x_1 x_2 + 2x_1 - 2 \\ 400x_2 - 400x_1^2 \end{bmatrix}$$

Once I had obtained partial derivatives I started computing the Hessian matrix H_f:
$\frac{\partial^2 f}{\partial x_1^2} = 800 \cdot 3x_1^2 - 800x_2 + 2$

$= 2400x_1^2 - 800x_2 + 2$

$\frac{\partial^2 f}{\partial x_2^2} = 400$

$\frac{\partial^2 f}{\partial x_1 \partial x_2} = \frac{\partial^2 f}{\partial x_2 \partial x_1} = -800x_1$
In conclusion, the Hessian matrix I achieved is the following:

$$Hf = \begin{bmatrix} 2400x_1^2 - 800x_2 + 2 & -800x_1 \\ -800x_1 & 400 \end{bmatrix}$$

2. Write the Taylor's expansion of f up to the second order around the point $(x_1, x_2) = (0, 0)$.

The Taylor series is a technique for approximating functions using polynomials at a specific point. The method represents a function as an infinite sum of terms based on the function's derivatives , the number of terms depends on the order of the Taylor's expansion.

The second order Taylor's series generic formula is the following:

$f(x^*+h) = f(x^*)+h^T\nabla f(x^*)+\frac{1}{2}h^T\nabla^2 f(x^*)h+o(\|h^2\|)$ where $x^*, h \in R^n$ and $\|h\| << 1$

Thus, the expansion of f around the point $(x_1, x_2) = (0, 0)$ is :

$f(0+h_1, 0+h_2) = f(0,0) + h^T\nabla f(0,0) + \frac{1}{2}h^T\nabla^2 f(0,0)h + o(\|h^2\|)$

$= 1 + \begin{bmatrix} h_1 & h_2 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} h_1 & h_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 400 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + o(\|h^2\|)$

$= 1 - 2h_1 + \frac{1}{2}(2h_1^2 + 400h_2^2) + o(\|h^2\|)$

$= 1 - 2h_1 + h_1^2 + 200h_2^2 + o(\|h^2\|)$

Exercise 2

Consider the quadratic minimization problem

$$\min_{x \in \mathbb{R}^n} J(x) = \frac{1}{2} x^T A x - b^T x,$$

where $A \in \mathbb{R}^{n \times n}$ is symmetric, and $x, b \in \mathbb{R}^n$.

a) Compute the gradient and Hessian of the functional J.

Firstly, I managed to calculate the derivative of $J(x)$ as the following:

$\nabla J(x) = \frac{1}{2} A^T x + \frac{1}{2} A x - b$

$= \frac{1}{2} x (A^T + A) - b$

Given that matrix A is symmetric , it is possible to write $A^T = A$:

$= \frac{1}{2} x (A + A) - b$

$= \frac{1}{2} \cdot 2Ax - b$

$= Ax - b$

Secondly I computed the second grade derivative of $J(x)$ obtaining that:

$\nabla^2 J(x) = A$

b) Write down the first order necessary condition for **a local minimizer** in unconstrained problem $\min_{x \in \mathbb{R}^n} f(x)$

The first order necessary condition says that :
If x^* is a local minimizer of f and f is continuously differentiable in a open neighborhood of x^*, $N(x^*)$, then $\nabla f(x^*) = 0$

Hence in the quadratic form expression we have that :

$$\nabla J(x) = 0 \implies Ax - b = 0$$

c) Write down the second order necessary and sufficient conditions for **a local minimizer** in unconstrained problem $\min_{x\in\mathbb{R}^n} f(x)$

On one hand,the second order necessary condition explains that :

If x^* is a local minimizer of f and $\nabla^2 f(x^*)$ exists and it is continuously in a open neighborhood of x^*, $N(x^*)$, then $\nabla f(x^*) = 0$ and $\nabla^2 f$ is positive semi-definite.

Thus, we have that :
$Ax - b = 0$ and $\nabla^2 f = A$ is positive semi-definite, so $\forall x, x^T Ax \geq 0$ and the eigenvalues of matrix A are $\lambda \geq 0$, $\forall \lambda \in \Lambda \ \wedge \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k$

On the other hand, the second order sufficient condition affirms that :

If $\nabla f(x^*)$ exists, it is continuous in a open neighborhood of x^*,$N(x^*)$, it is positive definite and $\nabla f(x^*) = 0$, then x^* is a strict local minimizer of f.

As a result, we have that :
$Ax - b = 0$ and $\nabla^2 f = A$ is positive definite, so $\forall x, x^T Ax > 0$ and the eigenvalues of matrix A are $\lambda > 0$, $\forall \lambda \in \Lambda \ \wedge \lambda_1 < \lambda_2 < \cdots < \lambda_k$

Exercise 3

Consider the following function:

$$f(x, y) = x^2 + \mu y^2$$

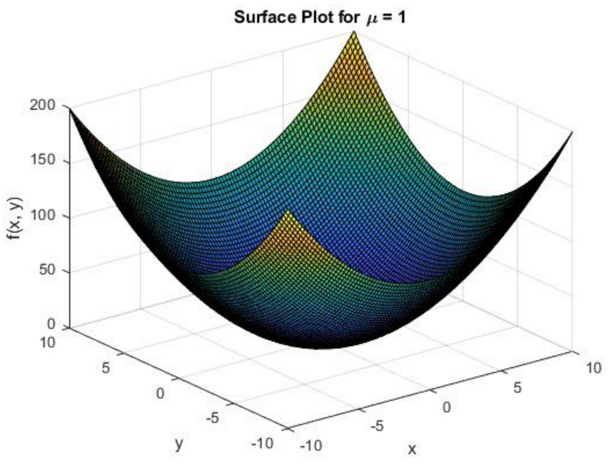
1. Write it down in quadratic form, i.e., $\frac{1}{2}x^T Ax - b^T x$, where $A \in \mathbb{R}^{2\times 2}$ and $b \in \mathbb{R}^2$.

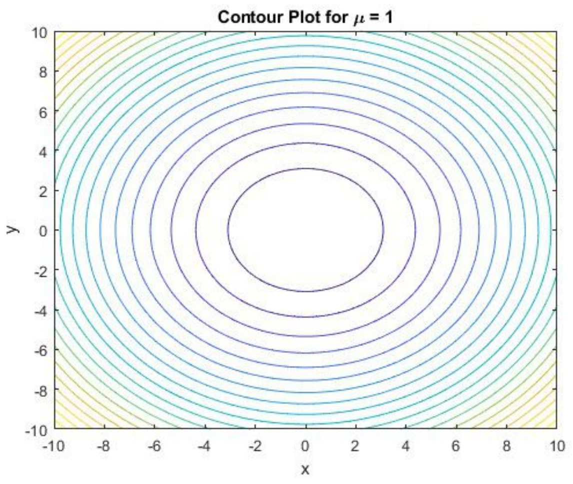
   The function f can be expressed in the quadratic form in the following way:
   $f(x, y) = \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 2\mu \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$

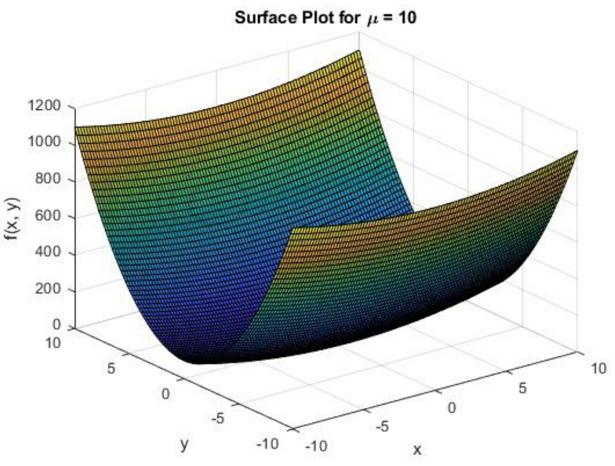
   $= \frac{1}{2}(2x^2 + 2\mu y^2)$

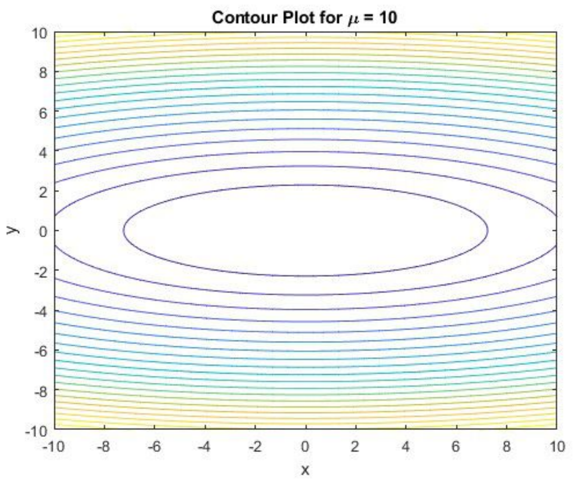
2. Plot the surface of the function (Matlab function: `surf`) and the corresponding contour plot (Matlab function: `contour`) for values $\mu = 1$ and $\mu = 10$. In both cases, use the square $[-10, 10] \times [-10, 10]$. Comment on the behavior of the isolines.

   The plots I obtained from the `surf` and `contour` functions are the following :









   Note: the matlab code I used to generate the plots are contained in the file `Ex3.m` in the section commented as ”3.2”

   It is possible to notice some differences between the two isolines plots:
   They assume different shapes ,indeed when $\mu = 1$ the isolines are circular and it means that the function grows up uniformly on all directions and the curve are symmetric to the origin. On the other hand in the $\mu = 10$ plot case , the lines assume an elliptic shape in which the function seems to grow rapidly along the y-axis due to the stretched curves , this makes realize that is more sensitive to variations along the vertical axis.This stretch along the y-axis is due to the fact that the matrix A in the quadratic form of the function has very different eigenvalues and this difference between them affects the behaviour of the function. When $\mu = 10$ we have that:
   $A = \begin{bmatrix} 2 & 0 \\ 0 & 20 \end{bmatrix}$ where $\lambda_1 = 2$ and $\lambda_2 = 20 \implies 2 << 20$

3. Considering that A is a symmetric positive-definite matrix, find the exact optimal step-length α. Show your computations.

   In order to compute the optimal step-length it is needed the following formula where $x^* \in R^n$:

$$\alpha_{opt} = \frac{\nabla^T f(x^*) \cdot \nabla f(x^*)}{\nabla^T f(x^*) \cdot A \cdot \nabla f(x^*)}$$

   Replacing the $\nabla f(x^*) = \begin{bmatrix} 2x \\ 2\mu y \end{bmatrix}$ where $x^* \in R^2$ and $A = \begin{bmatrix} 2 & 0 \\ 0 & 2\mu \end{bmatrix}$, the formula for compunting the α_{opt} becomes the following :

$$\alpha_{opt} = \frac{\begin{bmatrix} 2x & 2\mu y \end{bmatrix} \cdot \begin{bmatrix} 2x \\ 2\mu y \end{bmatrix}}{\begin{bmatrix} 2x & 2\mu y \end{bmatrix} \cdot \begin{bmatrix} 2 & 0 \\ 0 & 2\mu \end{bmatrix} \cdot \begin{bmatrix} 2x \\ 2\mu y \end{bmatrix}}$$

$$= \frac{4x^2+4\mu^2y^2}{\begin{bmatrix} 4x & 4\mu^2 y \end{bmatrix} \begin{bmatrix} 2x \\ 2\mu y \end{bmatrix}}$$

$$= \frac{4x^2+4\mu^2y^2}{8x^2+8\mu^3y^2}$$

$$= \frac{4(x^2+\mu^2y^2)}{8(x^2+\mu^3y^2)}$$

$$= \frac{1}{2} \cdot \frac{x^2+\mu^2y^2}{x^2+\mu^3y^2}$$

   Now it is possible to write the respective formulas in the case of $\mu = 1$ and $\mu = 10$:

   a) $\mu = 1$

$$\alpha_{opt} = \frac{1}{2} \cdot \frac{x^2 + y^2}{x^2 + y^2} = \frac{1}{2}$$

   b) $\mu = 10$

$$\alpha_{opt} = \frac{1}{2} \cdot \frac{x^2 + 100y^2}{x^2 + 1000y^2}$$

4. Write a Matlab code for the gradient method with a maximum number of iterations $N = 100$ and a tolerance `tol = 1e-8`. Minimize f for $\mu = 1$ and $\mu = 10$ with starting points: $(x_0, y_0) = (10, 0)$, $(0, 10)$, $(10, 10)$.

   The implementation of the steepest method and the computations of the function to the variation of μ is available at the matlab file named `Ex3.m` with the appropriate code comments

5. For each case, plot the iterations on the energy landscape in 2D, the $\log_{10}$ of the norm of the gradient, and the value of the energy function as functions of the iterations. Comment on the results.

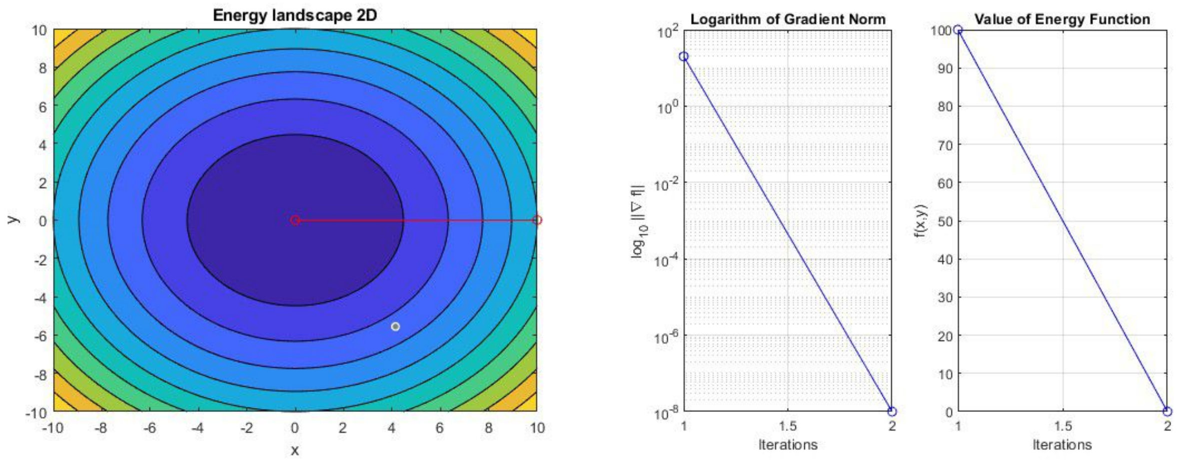


Figure 1: Plots for $\mu = 1$ and $x_0 = \begin{bmatrix} 10 & 0 \end{bmatrix}$

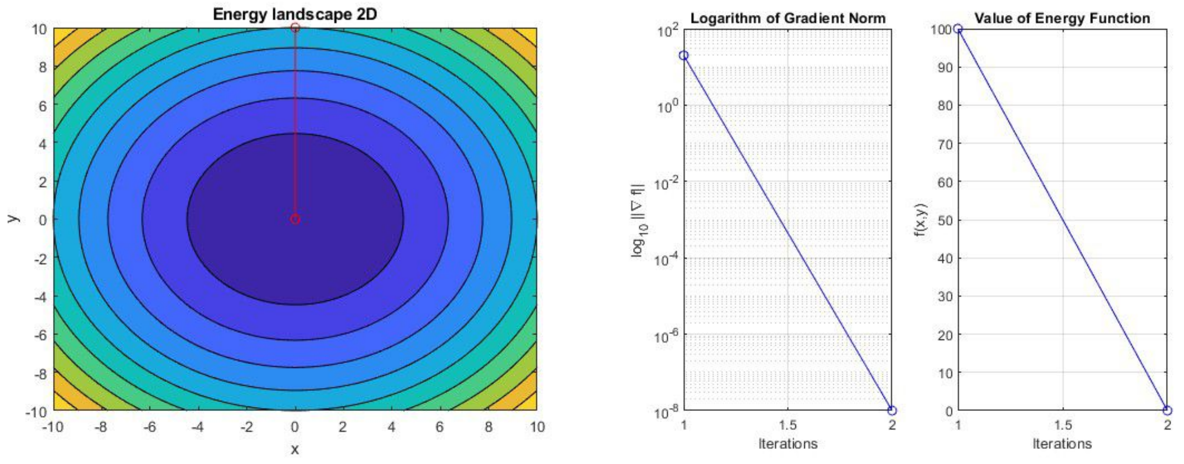


Figure 2: Plots for $\mu = 1$ and $x_0 = \begin{bmatrix} 0 & 10 \end{bmatrix}$

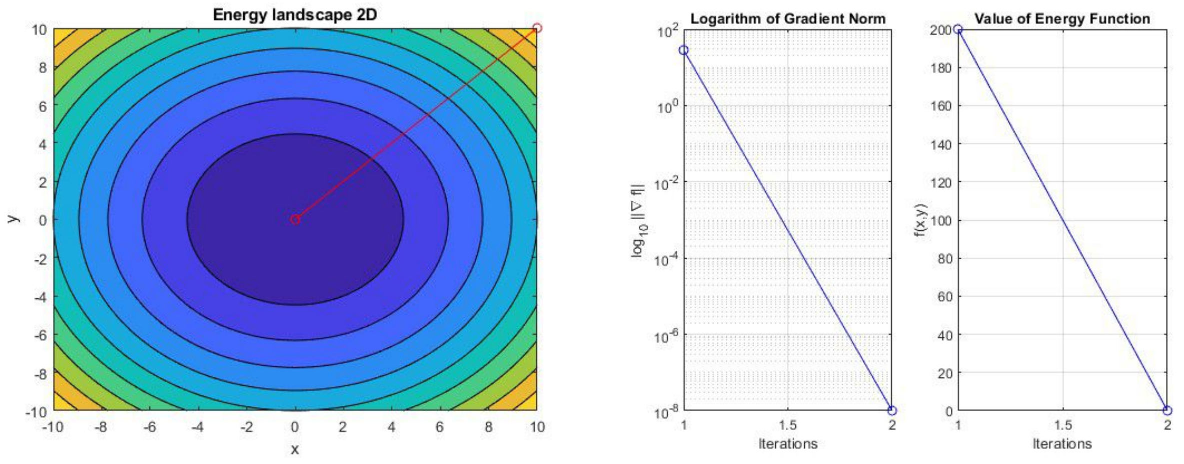


Figure 3: Plots for $\mu = 1$ and $x_0 = \begin{bmatrix} 10 & 10 \end{bmatrix}$

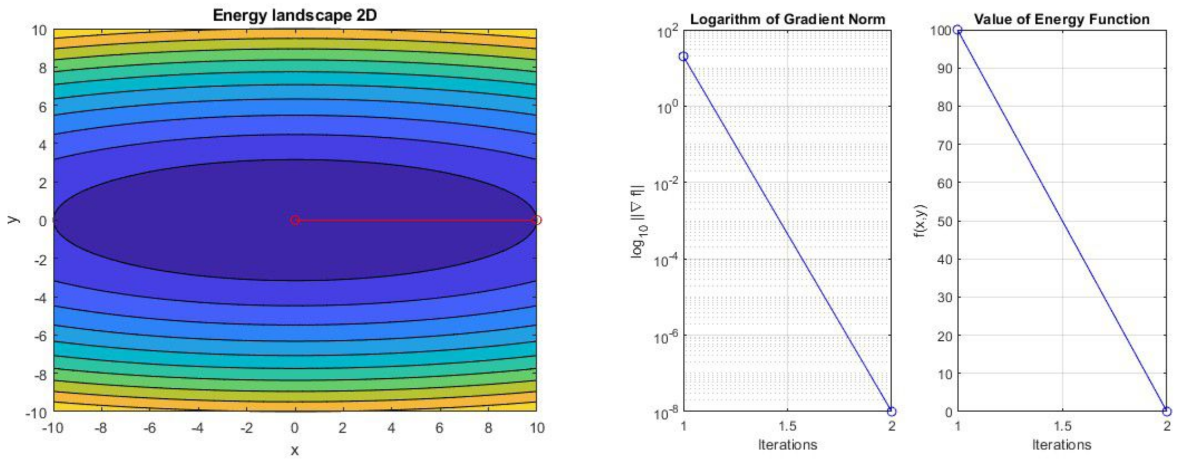


Figure 4: Plots for $\mu = 10$ and $x_0 = \begin{bmatrix} 10 & 0 \end{bmatrix}$

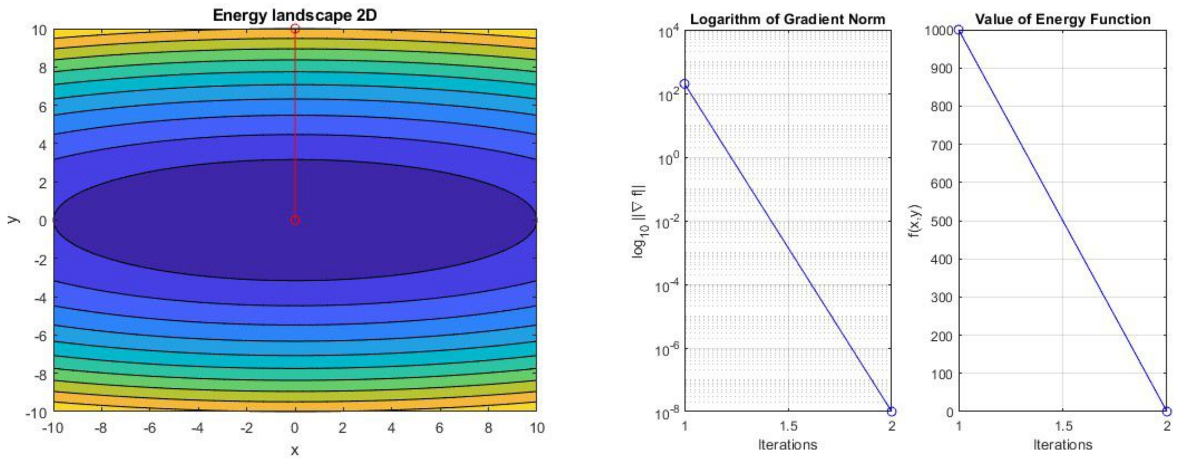


Figure 5: Plots for $\mu = 10$ and $x_0 = \begin{bmatrix} 0 & 10 \end{bmatrix}$

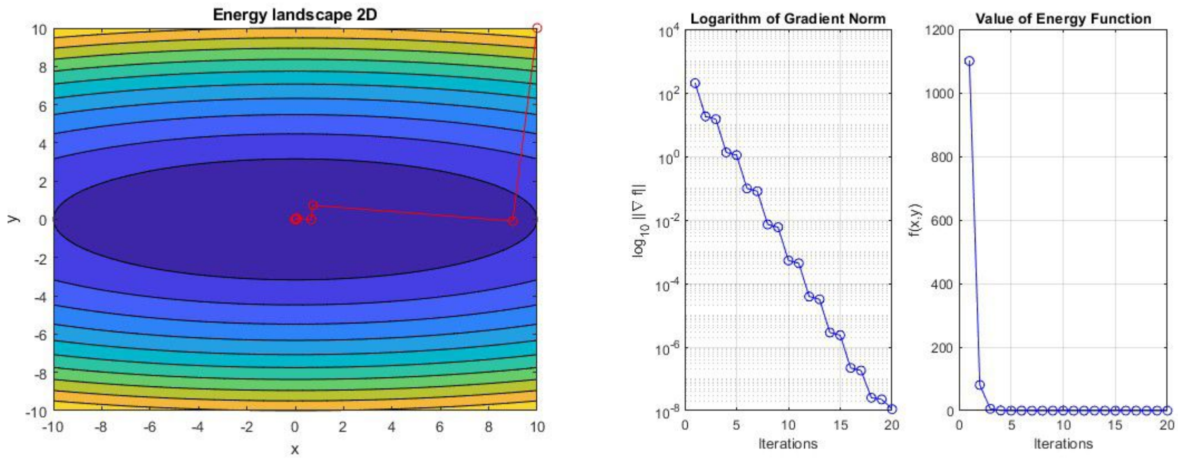


Figure 6: Plots for $\mu = 10$ and $x_0 = \begin{bmatrix} 10 & 10 \end{bmatrix}$

From the plots, it can be seen that ,in the case of $\mu = 1$, regardless of the starting point used, the application of the method converges in exactly one iteration. This behaviour is confirmed by the theorem describing the convergence of the steepest descent method applied to a strongly convex quadratic function (Theorem 3.3 , page 42_1), which shows that if the eigenvalues of the matrix A of the quadratic function have equal eigenvalues, then the method will converge in only one iteration.

On the other hand, in the case of $\mu = 10$, especially when the starting point is $x(10, 10)$, the number of iterations rises to 19 and it can be seen from the energy landscape that the search for the minimum assumes a zigzagging behaviour. This information shows us that the value of the starting point is an important factor that predominantly influences the convergence of the algorithm. In fact, in the cases in which $x = (10, 0)$ and $x = (0, 10)$, the process of searching for the minimum is concluded in a single iteration.

Note: the plots on the right show one iteration more than the actual number of cycles that are carried out to find the minimum of the function. This is because at the end of the computation of the algorithm, I decided to also include the values assumed by the logarithm of the gradient of the norm and the value of the energy function once the minimum of the function had been found, so as to be able to show the trend of the various pieces of information between the starting point and the final point.

References

1. Jorge Nocedal and Stephen J Wright. Numerical optimization. Springer, 1999 avaiable on iCorsi platform