Università della Svizzera italiana | Institute of Computing CI

Optimization Methods **Spring 2024**

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

Second assigment

Due date: Tuesday, 12 April 2024, 12:00 AM

Exercise 1

Consider the highly non-linear Rosenbrock's function:

$$f(x, y) := (1 - x)^2 + 100(y - x^2)^2 \tag{1}$$

1. Implement in MATLAB two functions:
   - the Newton's method (`Newton.m`)
   - the Gradient method (`GD.m`)

   where both methods can be run with or without the backtracking algorithm (`backtracking.m`), for reference see J. Nocedal and S. Wright, *Numerical optimization*, page 37. You can reuse your previous implementation of Gradient descent and simply extend it in a way that it can ensure the Wolfe's sufficient decrease conditions. Use the following values for backtracking parameters: $\alpha = 1$, $\rho = 0.9$. You can choose the parameter c such that it is in the interval $\{0.5, 10^{-4}\}$.

   The implementations of both algorithms are available in the respective matlab files named `Newton.m` and `GD.m` with the appropriate code comments

2. Minimize the Rosenbrock's function 1 by using the Steepest Descent (Gradient) method with backtracking and fixed step size $\beta = 1$. Use starting value $(0, 0)$, maximum number of iterations $N = 50000$, and tolerance TOL $= 10^{-6}$.

   First of all , I started to compute the first partial derivatives of x and y of the function $f(x, y) := (1 - x)^2 + 100(y - x^2)^2$ in order to calculate the gradient ∇f:

$$\begin{aligned}\frac{\partial f}{\partial x} &= 2 \cdot (1 - x) \cdot (-1) + 2 \cdot 100 \cdot (y - x^2) \cdot (-2x) \\ &= -2 - 2x - 400x \cdot (y - x^2) \\ &= -2 - 2x - 400xy + 400x^3 \\ &= 400x^3 - 400xy - 2x - 2\end{aligned}$$

$$\begin{aligned}\frac{\partial f}{\partial y} &= 2 \cdot 100 \cdot (y - x^2) \cdot 1 \\ &= 200 \cdot (y - x^2) \\ &= -200x^2 + 200y\end{aligned}$$

Hence, the gradient ∇f of the Rosenbrock's function is the following:

$$\nabla f = \begin{bmatrix} 400x^3 - 400xy - 2x - 2 \\ -200x^2 + 200y \end{bmatrix}$$

Once I obtained the necessary information for the Steepest descent algorithm , I compute the minimizations of the function with the designed parameters. The matlab code I created for this task is contained in the file `Ex1.m` in the section commented as "Exercise n° 1.2 - Steepest method"

3. Minimize the Rosenbrock's function 1 by using the Newton method with backtracking and fixed step size $\beta = 1$. Use starting value $(0, 0)$, maximum number of iterations $N = 50000$, and tolerance $\text{TOL} = 10^{-6}$.

   To begin with , I computed the second partial derivatives of x and y of the Rosenbrock's function in order to calculate the Hessian matrix $\nabla^2 f$:

$$\begin{aligned}\frac{\partial f}{\partial x^2} &= 3 \cdot 400x^2 - 400y - 2 \\ &= 1200x^2 - 400y - 2 \\ \frac{\partial f}{\partial y^2} &= 200 \\ \frac{\partial f}{\partial xy} &= \frac{\partial f}{\partial yx} = -400x\end{aligned}$$

Therefore, the Hessian matrix $\nabla^2 f$ of the Rosenbrock's function is the following:

$$\nabla^2 f = \begin{bmatrix} 1200x^2 - 400y - 2 & -400x \\ -400x & 200 \end{bmatrix}$$

When I compute the Hessian matrix, required for the Newton method , I aimed to minimize the function with the designed parameters. The matlab code I created for this task is contained in the file `Ex1.m` in the section commented as "Exercise 1.3 - Newton method"

4. Plot the obtained iterates on the energy landscape in 2D.

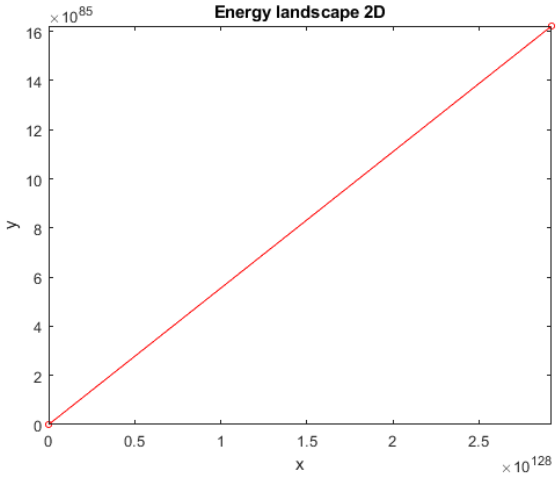

Figure 1: Without backtracking

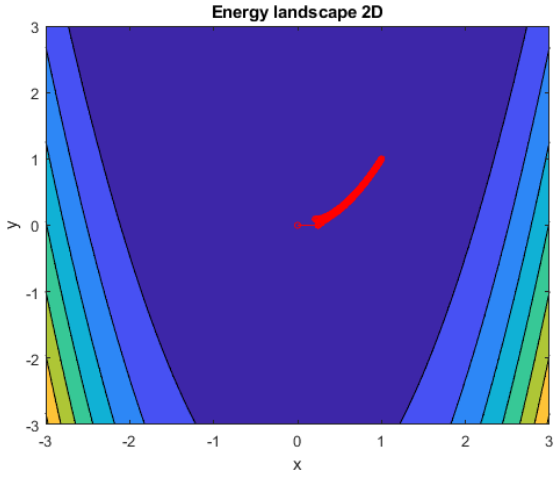

Figure 2: With backtracking

Figure 3: Gradient method results

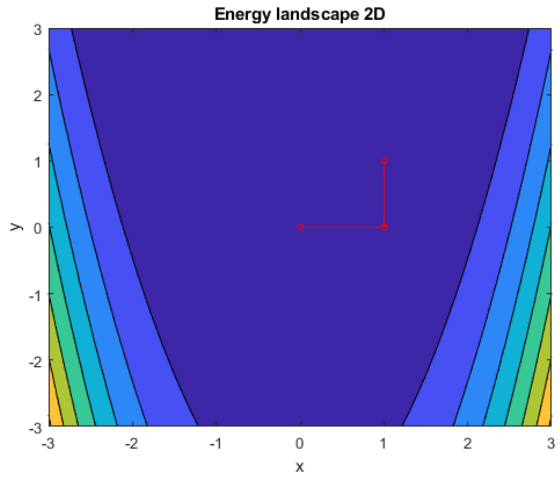

Figure 4: Without backtracking

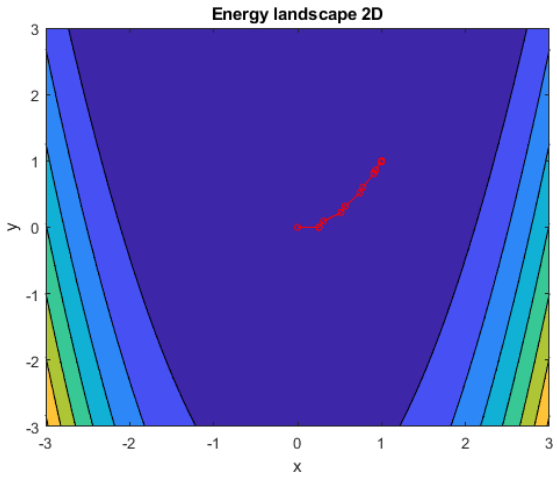

Figure 5: With backtracking

Figure 6: Newton method results

Note: the matlab code I used to generate the plots are contained in the file `Ex1.m` in the section commented as "Exercise n° 1.2 - Steepest method" and "Exercise 1.3 - Newton method"

5. Analyze convergence behaviour of the methods by plotting the gradient norm and the function value at each iteration.

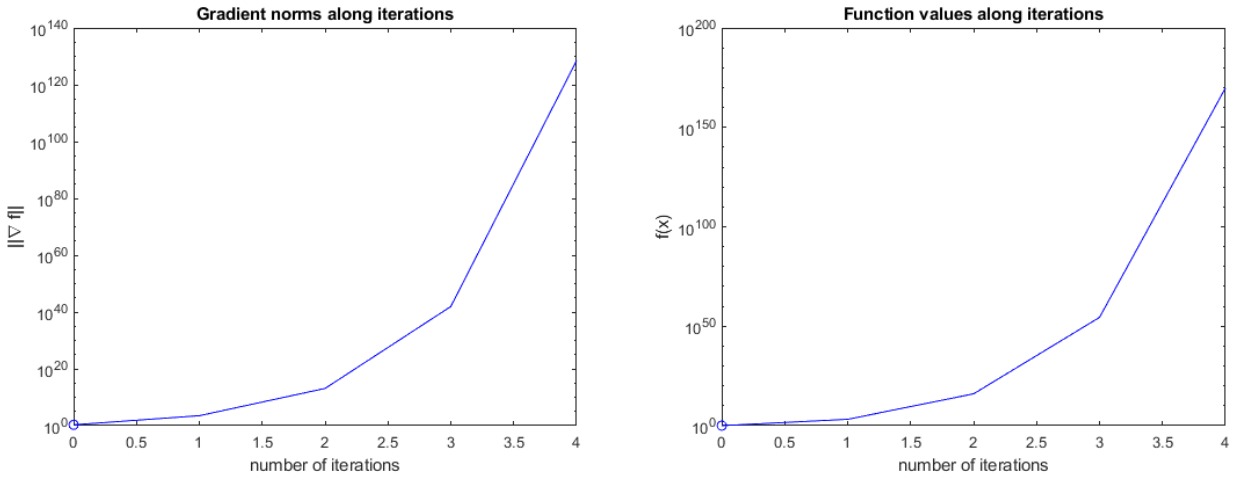

Figure 7: Gradient method convergence behaviour without backtracking

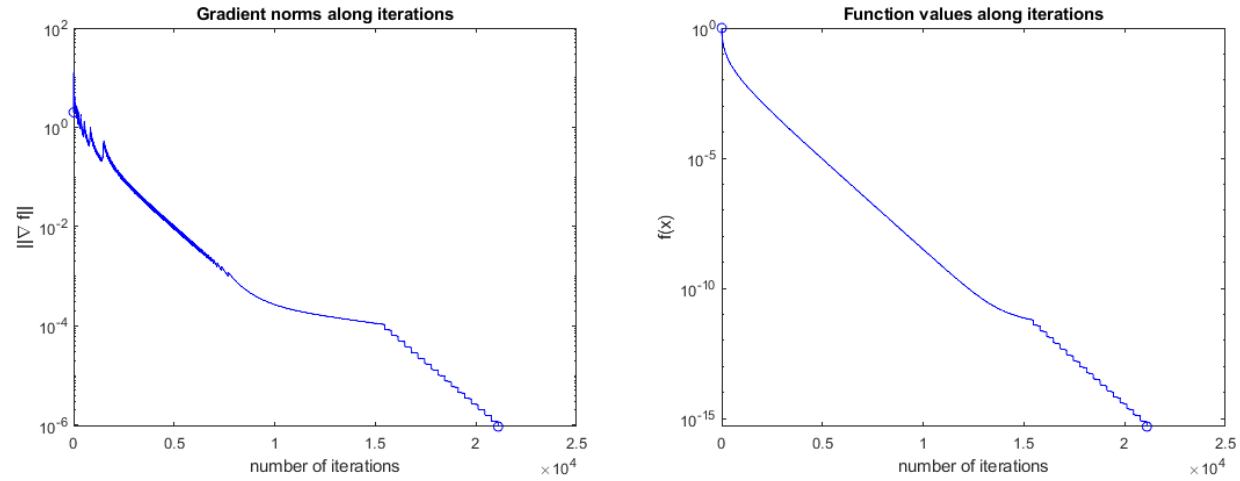

Figure 8: Gradient method convergence behaviour with backtracking

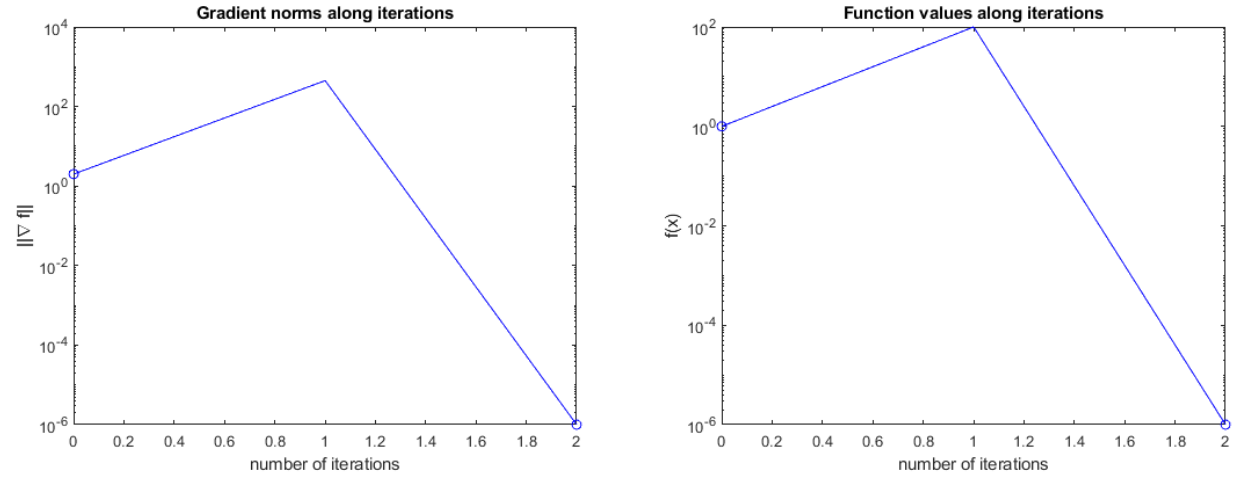

Figure 9: Newton method convergence behaviour without backtracking

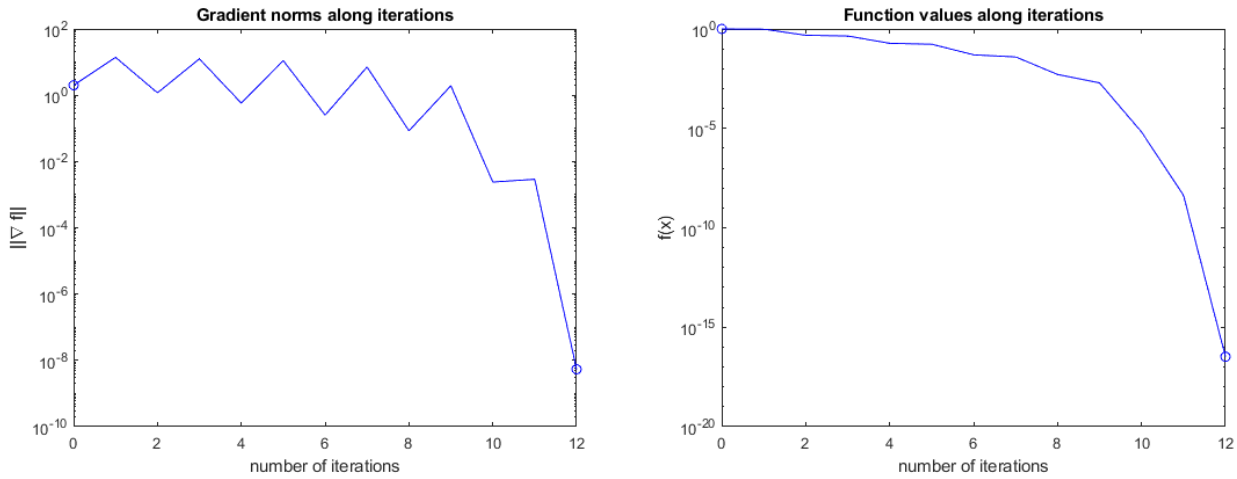

Figure 10: Newton method convergence behaviour with backtracking

6. Compare and comment on the performances of the different methods.

Firstly, it is possible to notice evident differences in term of convergence behaviour when the backtracking algorithm is applied to the Gradient method : the use of backtracking is essential to reach the minimum of the function in the Rosenbrock's function 1 otherwise the method tends to diverge to infinity without reaching its goal. Naturally the number of iteration is really pronounced due to this anomalous behaviour, in fact , without the backtracking the function diverges in few steps compared to the other version in which the method converges in almost 20 thousands iterations but in this case it reachs the minimum as it is possible to see from the gradient norms plots.
Secondly, focussing on the Newton method, the algorithm converges to absolute minimum of the function both with backtracking application and without it as it is possible to notice from the energy function plots . Moreover, the case without applying the backtracking appears to be extremely efficient and fast in reaching the minimum due to the 2 single iterations required in the process , probably this behaviour depends directly on the used starting point $x = (0, 0)$ given that it is quite near to the global minimum. Whereas with the application of the backtracking, the Newton method converges in few iterations but it is still a good result if it is compared with the Gradient method results for example.
In conclusion, it is possible to notice several differences between the two iterative methods and those ones are :

- *Number of iterations*: the steepest descent tends to converge slowly, or diverge if we consider the case without backtracking, due to the long and narrow valley around the global minimum of the function so the number of iterations is much high compared to the Newton method which converges in few steps also thanks to proximity at global minimum.
- *Backtracking application effect*: the application of the backtracking algorithm is essential to make the difference in reaching the convergence in fact thanks to that the steepest method is able to reach the global minimum allowing it to adapt the length of the pitch according to the local slope. On the other hand , the backtracking generally improve the convergence of the iterative method but in the Newton method case it is not necessary thanks to the the higher level of accuracy given by the usage of Hessian matrix in the process and the initial point pretty near to the global minimum.

Exercise 2

Consider again the highly non-linear Rosenbrock's function:

$$f(x, y) := (1 - x)^2 + 100(y - x^2)^2$$

1. Implement the BFGS method (`BFGS.m`) with backtracking. For reference see J. Nocedal and S. Wright, Numerical optimization, page 141.

   The implementation of the algorithm is available in the respective matlab function file named `BFGS.m` with the appropriate code comments

2. Test your implementation by minimizing the Rosenbrock's function. Use starting values $x_0 = (0, 0)$, $H_0 = I$, maximum number of iterations $N = 500$, and tolerance TOL $= 10^{-6}$.

   I compute the minimization of the function with the designed parameters and for completing the task I created a matlab code file named `Ex2.m`

3. Plot the obtained iterates on the energy landscape in 2D.

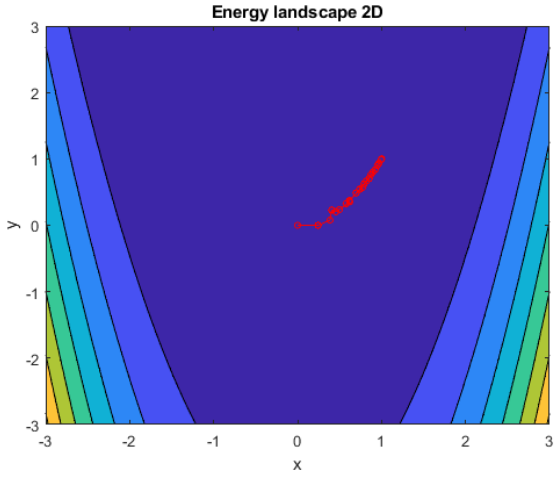

Figure 11: BFGS results

4. Analyze convergence behavior of the methods by plotting the gradient norm and the function value at each iteration.

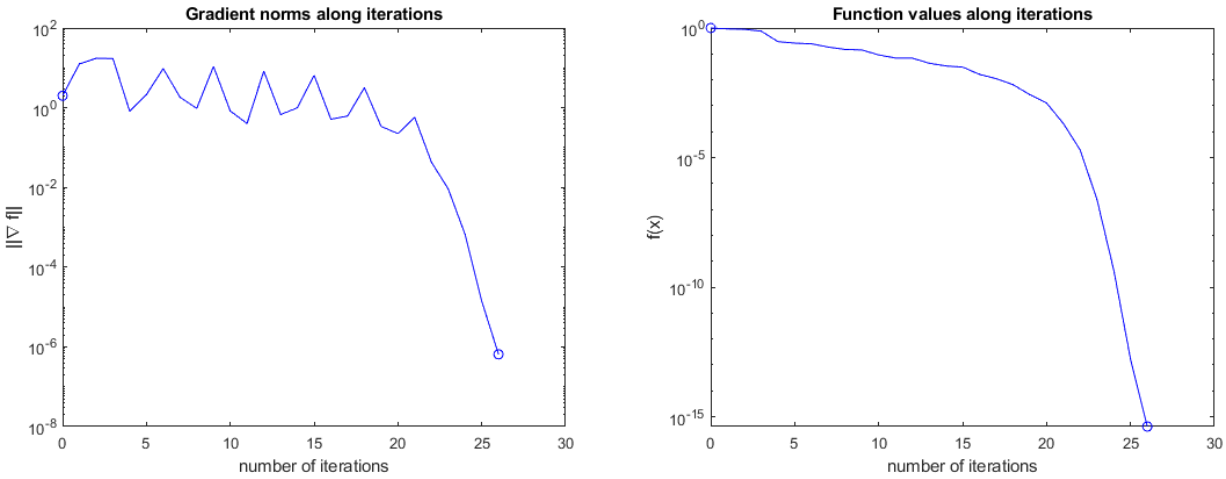

Figure 12: BFGS method convergence behaviour

5. Produce a table in which you compare the number of iterations required by BFGS, by Newton's method (with backtracking), and by Gradient method (with backtracking). You can use the results from the previous exercise. Comment on the results by comparing the different methods.

BFGS	Newton method	Gradient method
26	12	21102

Analizying the data inside the table can be noticed that the Newton method converge faster than the other two methods when the backtracking is applyed. The BFGS also reaches the convergence in few steps in a fast way quite near to the Newton method result. Despite the methods have both a quadratic convergence rate, the BFGS uses an approximation of the Hessian matrix in its iterative process and performs $O(n^2)$ arithmetic operations compared to the Newton method which requires the second derivatives of function and to solve a linear system at cost of $O(n^3)$. The difference between the approximation and the real Hessian matrix causes an higher number of iterations in the Quasi-Newton method but, despite that, the BFGS results faster and more efficient thanks to the cost of each iteration. On the other hand, the Gradient method appears to have difficulty in reaching the minimum as it is possible to see from the high number of iterations probably this is due to different facts such as its linear convergence rate and also the valley shape that the function assumes around the area of global minimum.

Exercise 3: Steepest Descent (20/100)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given by $f(x) = \frac{1}{2}x^T Ax - b^T x$ with A symmetric positive definite. How many iterations does the Steepest Descent (SD) method take to minimize the function f if we use the optimal step length? Please, prove your answer.

To begin with, from the hypothesis that the function can be written in the quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x$ in which the matrix A is symmetric and positive definite, thus $x^T Ax > 0$ $\forall x \in \mathbb{R}^n$, it is possible to affirm that the function $f(x)$ is strictly convex. Starting from these assumptions we can compute and obtain that the gradient of the function is equal to $\nabla f = Ax - b$ and that the minimizer point x^* is obtained by posing the gradient equal to zero : $Ax - b = 0 \to Ax = b$. As we know, the SD method decreases the value of the function at each iteration, due to its strictly convex behaviour, with the aim of reaching the global minimum point according to the iterative procedure described by the expression :

$$x_{i+1} = x_i - \alpha_i \nabla f(x_i)$$

where i is the number of the iteration.
The initial conditions affirm the method use the optimal step length which is computed as :

$$\alpha_i = \frac{\nabla f(x_i)^T \nabla f(x_i)}{\nabla f(x_i)^T A \nabla f(x_i)}$$

Thus it is possible to rewrite the procedure as the following:

$$x_{i+1} = x_i - \frac{\nabla f(x_i)^T \nabla f(x_i)}{\nabla f(x_i)^T A \nabla f(x_i)} \nabla f(x_i)$$

When the Gradient method is executed with the optimal step length on a strictly convex quadratic function we have that the error norm, i.e. a norm that measures the difference between the current objective value of the function and the optimal objective function value and is used to measure the rate of convergence described as :

$$\frac{1}{2}||x - x^*||_A^2 = f(x) - f(x^*)$$

satisfies the following inequality :

$$\|x_{k+1} - x^*\|_A^2 \leq \underbrace{\left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2}_{<1} \cdot \|x_k - x^*\|_A^2 \quad 1$$

where $\lambda \geq 0$, $\forall\ \lambda \in \Lambda \wedge \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of the matrix A.
This expression underlines how the error norm decreases linearly at each iteration of the process and in particular illustrates how the number of iterations depends on the difference between the eigenvalues of the matrix A: a particular case occurs when all the matrix eigenvalues are equal, in that case the method will converge in a single iteration.

References

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

2. "Notes on the optimization" avaiable on iCorsi platform