Computational Mathematics for data analysis and learning

Michele Morisco (505252), Margherita Pensa (532252)
Department of Computer Science, University of Pisa, Pisa, Italy
A.Y. 2022/2023

1 Introduction

We start to provide a short description of the following problem and then we describe its study:

- (P) is the problem of estimating the matrix norm $\|A\|_2$ for a (possibly rectangular) matrix $A \in \mathbb{R}^{m \times n}$, using its definition as an (unconstrained) maximum problem.
- (A1) is a standard gradient descent (steepest descent) approach.
- (A2) is a quasi-Newton method such as BFGS.

1.1 Matrix norm

In this introductory section, we define a matrix norm. A matrix mxn can be viewed as a vector in the mn-dimentional space. In general, a matrix norm must satisfy the three vector norm conditions applied in the mn-dimensional vector space of matrices:

$$\|A\| \geq 0 \; and \; \|A\| = 0 \; only \; if A = 0,$$

$$\|A + B\| \leq \|A\| + \|B\|,$$

$$\|\alpha A\| = |\alpha| \|A\|.$$

To measure the "size" of such a matrix we can use any mn-dimentional norm with some special rules: *induced matrix rules*.

Given vector norms $\|\cdot\|_{(n)}$ and $\|\cdot\|_{(m)}$ on the domain and a matrix $A \in \mathbb{C}^{m \times n}$, the induced matrix norm $\|A\|_{(m,n)}$ is the smallest number C for which the following inequality holds for all $x \in \mathbb{C}^n$

$$\|Ax\|_{(m)} \leq C \|x\|_{(n)} \tag{1}$$

Since $\|ax\| = |a|\|x\|$, the action of A is determined by its action on unit vectors, so the matrix norm can be equivalently defined as:

$$\|A\|_{(m,n)} = \sup_{x \in \mathbb{C}^n, x \neq 0} \frac{\|Ax\|_{(m)}}{\|x\|_{(n)}} = \sup_{x \in \mathbb{C}^n, \|x\|_{(n)} = 1} \|Ax\|_{(m)} \tag{2}$$

1.2 The 2-norm of a matrix as unconstrained optimization problem

We now focus on the 2-norm of a matrix-induced using the 2-norm vector definition [[3], p. 23].

$$\|A\|_2 := \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \quad \text{where } x \in \mathbb{R}^n \text{ and } A \in \mathbb{R}^{m \times n} \tag{3}$$

We use the definition of 2-norm [[7], p. 18]

$$\|x\|_2 := \sqrt{x^T x}$$

to re-write the formula (3) as follows

$$\|A\|_2 = \max \sqrt{\frac{(Ax)^T Ax}{x^T x}} = \max \sqrt{\frac{x^T A^T Ax}{x^T x}} = \max \sqrt{\frac{x^T Qx}{x^T x}}$$

where $Q = A^T A$. The products between matrix A and its transpose are symmetric square matrices using their own transpose:

$$(A^T A)^T = A^T (A^T)^T = A^T A$$

In addition, we can see that Q is also a diagonalizable matrix and it is a positive semi-definite matrix because

$$x^T Qx = \|Ax\|_2^2 \geq 0 \iff A^T A \succeq 0$$

where $x^T A^T Ax = \|Ax\|_2^2$.
We can notice that $\max h(g(.)) = h(\max g(.))$, because we have the sufficient condition that the h is a monotone increasing, indeed the square root is a monotonic function because with $f(x) = \sqrt{x} \iff f'(x) = \frac{1}{2\sqrt{x}} > 0$ if $x > 0$ so we have

$$\|A\|_2 = \sqrt{\max \frac{x^T Qx}{x^T x}}$$

This is a maximum problem, for convenience we write our optimisation problem as a minimum problem, remembering that a maximum problem can always be transformed into a minimum problem: let f a function and X a domain we have $x_0 \in X$ is a global minimum point of the function $f : X \to \mathbb{R}$, if $f(x_0) \leq f(x) \forall x \in X$ consequently, we can have

$$-f(x_0) \geq -f(x) \iff min\{f(x)\} = -max\{-f(x)\}$$

Therefore, our optimization problem is

$$\min_x -\frac{x^T Qx}{x^T x} \tag{4}$$

1.3 Function properties

In this section, we will discuss some helpful properties of our objective function for the following chapters. We first define the function's domain, which is D = $\mathbb{R}^n \setminus 0$.

1.3.1 Gradient of the function

Our function is twice continuously differentiable under the condition that Q is positive definite. We need to define the gradient of the function we need for the next chapter, therefore we have to find the partial derivatives of the function because they are the components of gradient [[2], p. 640-642]. Then, the objective function can be written as

$$f(x) = -\frac{x^T Q x}{x^T x} = -\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij}}{\sum_{i=1}^{n} x_i^2}$$

where q_{ij} are the elements of Q.
After we can apply the partial derivatives to it:

$$\begin{aligned}\frac{\partial f}{\partial x_k} = -\frac{\partial(x_k^2 q_k) * (\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij}) * \partial(x_k^2)}{(\sum_{i=1}^{n} x_i^2)^2} \\ \iff \frac{2x_k(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij})}{(\sum_{i=1}^{n} x_i^2)^2} - \frac{\sum_{j=1}^{n} 2x_j q_{kj}}{\sum_{i=1}^{n} x_i^2} \\ \iff \frac{2x(x^T Q x)}{(x^T x)^2} - \frac{2Qx}{x^T x}\end{aligned} \tag{5}$$

In the end, we can write the gradient of the function as

$$\nabla f(x) = \frac{2x(x^T Q x)}{(x^T x)^2} - \frac{2Qx}{x^T x} \tag{6}$$

We can proceed to compute the second-order partial derivatives:

$$\begin{aligned}\frac{\partial f}{\partial x_k} = -\frac{\partial(2x_k(x_k^2 q_k)) * (\sum_{i=1}^{n} x_i^2)^2 - (2x_k \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j q_{ij}) * \partial((x_k^2)^2)}{(\sum_{i=1}^{n} x_i^2)^4} \\ + \frac{\partial(2x_k q_k)) * (\sum_{i=1}^{n} x_i^2) - (\sum_{j=1}^{n} x_j q_{kj}) * \partial(x_k^2)}{(\sum_{i=1}^{n} x_i^2)^2} \\ \iff \nabla^2 f(x) = \frac{4(x^T Q x)}{(x^T x)^2} - \frac{8(x^T Q x) * (x^T x)}{(x^T x)^3} + \frac{2(x^T Q x) * I}{(x^T x)^2} - (\frac{2Q}{x^T x} - \frac{4x^T Q x}{(x^T x)^2})\end{aligned} \tag{7}$$

This involves taking the partial derivatives of the first-order partial derivatives concerning each component of x.

1.3.2 Convex and continuous function

We want to know whether the function is convex because if it is, every local minimizer is also a global minimizer. This will be helpful for the next topics that we will cover in the next chapters. To study convexity we have to see first if the function is continuous in its

domain [[1], p. 47-49]. We can check the derivatives to study the continuity and differentiability of the function. The partial derivatives (5) are continuous and differentiable in all points except in 0 and we can achieve the partial derivatives for all vectors $x \in \mathbb{R}^n_{\neq 0}$.
We can conclude that the objective function is continuous in $\mathbb{R}^n_{\neq 0}$, that is why if we have a stationary point it may not be the global optimum because the function is not convex [[5], p. 12-14].
The formal definition of convexity for a function f is as follows: A function f defined on a convex set C is convex if, for every pair of points x_1, x_2in C and for every λ such that $0 < \lambda < 1$, it is verified that:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

To prove that our function is not convex, we can look for a counterexample. Let us take Q, a symmetrical and positively semi defined matrix and two vectors x_1 and x_2 as:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

So we have

$$f(x_1) = -1, f(x_2) = -2,$$
$$f(\lambda x_1 + (1-\lambda)x_2) = f\left(\begin{bmatrix} \lambda \\ 1-\lambda \end{bmatrix}\right) = -\frac{3\lambda^2 - 4\lambda + 2}{2\lambda^2 - 2\lambda + 1},$$
$$\lambda f(x_1) + (1-\lambda) f(x_2) = \lambda - 2$$

if we take $\lambda = \frac{3}{4}$ we'll have

$$-\frac{11}{10} \leq -\frac{5}{4}$$

In this case, the property is violated so the function is not convex.

1.3.3 Lipschitz continuity

We want to check if the objective function is Lipschitz continuous because it will be an important property to use in the next chapters. Firstly, we can notice that the function is bounded below in the domain $\mathbb{R}^n_{\neq 0}$, for its definition we have,

$$\exists M \in \mathbb{R} : |f(x)| \geq M \ \ \forall x \in \mathbb{R}$$

therefore applying our function (4), we have:

$$\begin{aligned} & \exists M \in \mathbb{R}^+ : -f(x) = \frac{x^T Q x}{x^T x} \leq M \ \ \forall x \in \mathbb{R}^n_{\neq 0} \\ \iff & \| - f(x)\| = \| \frac{x^T Q x}{x^T x} \| \leq \frac{\|x\| \cdot \|Q\| \cdot \|x\|}{\|x\| \cdot \|x\|} \leq \frac{\|x\|^2 \|Q\|}{\|x\|^2} \leq \|Q\| \leq M \end{aligned} \tag{8}$$

We write the definition of Lipschitz continuity: considering a function f and an interval S if $\exists c > 0$ s.t. $|f(x) - f(y)| \leq c|x - y| \quad \forall x, y \in S$ where c is a constant. Moreover, f is locally Lipschitz continuous at x if $\exists \epsilon > 0$.
We have seen that our function $f \in \mathbb{R}^+$ is continuous and differentiable in all points except in 0, if the gradient is bounded the function will be Lipschitz continuous.
We have shown that each of the two parts of the function is limited. Therefore, the function $\nabla f(x)$ is limited. Consequently, we have to keep track of the closed unit ball in the vector space for norm-2, its unit ball is spherical $\{x \in S_\epsilon : \|x\| = 1\}$, so we can notice that the gradient (6) is bounded if

$$\begin{aligned} \exists M \in \mathbb{R} : |\nabla f(x)| \leq M \quad \forall x \in \mathbb{R}^n_{\neq 0} \\ \iff \|\frac{2x(x^TQx)}{(x^Tx)^2} - \frac{2Qx}{x^Tx}\| \leq M \\ \|\frac{2x(x^TQx)}{(x^Tx)^2} - \frac{2Qx}{x^Tx}\| \leq \|\frac{2x(x^TQx)}{(x^Tx)^2}\| + \|\frac{2Qx}{x^Tx}\| \\ \leq \frac{|2| \cdot \|x\| \cdot \|x\| \cdot \|Q\| \cdot \|x\|}{(\|x\| \cdot \|x\|)^2} + \frac{|2| \cdot \|Q\| \cdot \|x\|}{\|x\| \cdot \|x\|} \leq 4\|Q\| \leq M, \quad ||x|| = 1 \end{aligned} \tag{9}$$

from every set $S_\epsilon = \{x : \|x\| \geq \epsilon > 0\}$ from M/ϵ out of its ball because in different circumstances the gradient could invalidate the equation since the denominators could be 0.

1.3.4 Computing exact line search

In this section we calculate the formula to perform the exact line search because it will be useful for our algorithms. So, we compute the tomography of our function:

$$\Phi(\alpha) = f(x_i + \alpha \cdot d_i), \quad \alpha > 0$$

then we derive it with respect to α and we put the derivative equal to 0 to find the

stationary points:

$$
\begin{aligned}
\frac{d\Phi(\alpha)}{d\alpha} &= 0 \\
-\frac{d((x_i+\alpha\cdot d_i)^T Q(x_i+\alpha\cdot d_i)) * ((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} & \\
-\frac{(x_i+\alpha\cdot d_i)^T Q(x_i+\alpha\cdot d_i) * d((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} &= 0 \\
\iff -\frac{(2x_i^T Q d_i + 2\alpha\cdot d_i^T Q d_i) * ((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} & \\
-\frac{(x_i+\alpha\cdot d_i)^T Q(x_i+\alpha\cdot d_i) * (2x^T d + 2\alpha\cdot d^T d)}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} &= 0 \\
\iff \frac{\alpha^2((d_i^T Q d_i)(x_i^T d_i) - (x_i^T Q d_i)(d_i^T d_i))}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} & \\
+\frac{\alpha((d_i^T Q d_i)(x_i^T x_i) - (x_i^T Q x_i)(d_i^T d_i))}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} & \\
+\frac{(x_i^T Q d_i)(x_i^T x_i) - (x_i^T Q x_i)(x_i^T d_i)}{((x_i+\alpha\cdot d_i)^T(x_i+\alpha\cdot d_i))^2} &= 0
\end{aligned}
\tag{10}
$$

Given x_i, d_i and $Q \in R^{n\times n}$, a symmetric matrix, we have $Q^T = Q$ so $(x_i^T Q d_i) = (x_i^T Q d_i)^T = (d_i^T Q x_i)$ and we got a polynomial of α with degree of 2 in the numerator. We check if the denominator is not equal to 0:

$$
\begin{aligned}
((x_i+\alpha d_i)^T(x_i+\alpha d_i))^2 &\neq 0 \\
\iff (x_i+\alpha d_i)^T(x_i+\alpha d_i) &\neq 0 \\
\iff ||(x_i+\alpha d_i)||_2^2 &\neq 0 \\
\iff (x_i+\alpha d_i) &\neq 0 \\
\iff x_i &\neq -\alpha d_i
\end{aligned}
\tag{11}
$$

where we have a solution for α if only if x and d are linearly independent.
If, absurdly, $(x_i + \alpha d_i) = 0$, the conditions for choosing the step would be contradicted since the step is chosen to gradually reduce the value of the objective function.
In addition, we can prove if the Wolfe conditions

$$\phi'(\alpha) \geq m\phi'(0) \quad 0 < m < 1$$

are satisfied for α such that $\Phi'(\alpha) = 0$, $\Phi'(0) = \nabla f(x)^T d$. In the SGD $d = -\nabla f(x)$ is negative so $\Phi'(0) < 0$ and so the conditions are verified. In the case of the BFGS, $d = -H_i \nabla f(x)$ hence $\Phi'(0) = \nabla f(x)^T(-H_i \nabla f(x))$, since H is symmetrical, $\Phi'(0) = -\nabla f(x)^T H_i \nabla f(x))$. Because H is positive definite, we can say that $\nabla f(x)^T H_i \nabla f(x) \geq 0$ for every $\nabla f(x) \neq 0$ so $\Phi'(0) < 0$ and the Wolfe conditions are verified.

2 Standard Gradient Descent

In this section we introduce the first algorithm we implemented to solve the optimization problem presented in Chapter 1, the standard gradient descent. We briefly explain how it works and analyse the convergence and complexity of the algorithm.
The standard gradient descent, also known as the steepest descent method, is used to find the minimum of a function and is commonly employed in machine learning and optimization problems. It is an iterative optimization algorithm of the form:

$$x_{i+1} = x_i + \alpha_i d_i, \tag{12}$$

it use the *line search* strategy: the algorithm first chooses the direction d, which in this case is the negative gradient of the function, and then chooses the length of the alpha step with which to move along this direction. Thus, at each iteration:

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

to obtain from x_i a new iterate x_{i+1} with a lower value of the function since the negative gradient of a function always points to the direction in which the function decreases the most. To verify this claim, we appeal to Taylor's theorem [[5], Theorem 2.1]:

Theorem 2.1 *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $\in \mathbb{R}^n$. Then we have that*

$$f(x+p) = f(x) + \nabla f(x+tp)^T p,$$

for some $t \in (0,1)$. Moreover, if f is twice continuously differentiable, we have that

$$\nabla f(x+p) = \nabla f(x) + \int_0^1 \nabla^2 f(x+tp)p dt,$$

and that

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp)p,$$

for some $t \in (0,1)$.

which tells us that for any search direction d and step-length parameter α, we have

$$f(x_i + \alpha d) = f(x_i) + \alpha d^T \nabla f(x_i) + \frac{1}{2}\alpha^2 d^T \nabla^2 f(x_i + td)d \quad \text{for some } t \in (0, \alpha)$$

The rate of change of f along the direction d at x_i is the coefficient of α, i.e. $d^T \nabla f(x_i)$. Therefore, the fastest decreasing unit direction d is the solution to the problem.

$$\min_d d^T \nabla f(x_i), \quad \text{subject to } \|d\| = 1$$

If we call θ the angle between d and $\nabla f(x_i)$ we know that $d^T \nabla f(x_i) = \|d\| \|\nabla f(x_i)\| \cos\theta = \|\nabla f(x_i)\| \cos\theta$, so we can infer that the minimizer is reached when $\cos\theta = -1$ and $d =$

$-\nabla f(x_i)/\|\nabla f(x_i)\|$ is orthogonal to the function.

By descending along the steepest direction, the algorithm attempts to find the optimal solution that minimizes the objective function. A general outline of the algorithm:

Algorithm 1 SGD

procedure x = SGD(f, x, ϵ):
while $\|\nabla f(x)\| > \epsilon$ **do**:
$d = -\nabla f(x)$;
$\alpha = stepsize(f, x, d)$
$x = x + \alpha * d$;
end while

where ϵ is a certain tolerance.

2.1 Exact line search

The crucial part of the algorithm is how the stepsize is chosen. Need to avoid two opposite problems:

- Scylla : α_i must not be too large to avoid $f(x_{i+1}) > f(x_i)$
- Charybdis : α_i must not be too small to avoid stalling far from f_*

Since we are trying to minimize $f(x)$, the ideal step size α would be the global minimizer of the tomograph of function defined by:

$$\Phi(\alpha) = f(x_i + \alpha \cdot d_i), \quad \alpha > 0$$

This would require solving an additional optimisation problem, undoubtedly simpler than the original one, since it involves minimizing a function of only one variable. However, this problem may require too many evaluations of $f(x)$ and its gradient, but in some cases, we can exploit the fact that f must be evaluated at points along the half-line $\{x_i + \alpha \cdot d_i | \alpha > 0\}$ so we can consider the restriction of f along the chosen direction to reduce the total computational effort. We then proceed to the so-called *exact line search*, i.e. the new optimisation problem is solved: α is chosen to minimize f along the ray $\{x_i + \alpha \cdot d_i | \alpha > 0\}$:

$$\alpha = argmin_{t>0} f(x_i + \alpha \cdot d_i) \tag{13}$$

In our case as shown in chapter we compute the exact line search differentiating the tomography of our function for α and put the derivative equal to 0 to find the stationary points.

2.2 Convergence of the Standard Gradient Descent

The convergence of the steepest descent algorithm depends on several factors, including the properties of the function being optimized and the step size used at each iteration.

Regarding the properties of the function, as we explained in section 1.2.2, the function is unbounded below, the partial derivatives are continuous and differentiable in all points except in 0 and the function is Lipschitz continuous in a sphere S = $x : \|x\| \geq \epsilon > 0$. So we can look for our solution in a domain D = $\mathbb{R}^n / \ S(0,\epsilon) \forall \epsilon > 0$ in which the function is Lipschitz continuous, we can choose $\epsilon < \|x_0\|_2$ and $\|x_k\|_2 \in$ D $\forall k \geq 0$. The sequence of points x_i produced by the algorithm has the following property:

$$\langle \nabla f(x_i), \nabla f(x_{i+1}) \rangle = 0$$

At each iteration, we perform an exact-line search, so the next point is chosen precisely because its directional derivative along $d_i = -\nabla f(x_i)$ is zero, i.e. the gradient in x_{i+1} is perpendicular to the gradient in x_i. This property guarantees that if the succession $x_i \to x$ converges, then it converges to a stationary point, in fact:

$$lim_{x\to\infty} \langle \nabla f(x_i), \nabla f(x_{i+1}) \rangle = \langle \nabla f(x), \nabla f(x) \rangle = \|\nabla f(x)\|^2 = 0$$

We must now prove that the function converges. The following theorem [[5], Theorem 3.2], due to Zoutendijk, shows that the steepest descent method is globally convergent.

Theorem 2.2 *Consider any iteration of the form 12, where d_i is a descent direction and α_i satisfies the Wolfe condition. Suppose that f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set N containing the level set $L = x : f(x) \leq f(x_0)$, where x_0 is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on N, that is, there exists a constant $L > 0$ such that*

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\|, \quad \textit{for all } x, \tilde{x} \textit{ in } N.$$

As we said in section 1.3.5, α_i satisfies the Wolf condition.
The theorem requires that the gradient ∇f is Lipschitz continuous so that it is guaranteed that ∇f cannot change too fast. To check this assumption we can consider the 7 and we could apply the same procedure we saw in Section 1.3.3 on $\nabla^2 f$ and prove if it is bounded, but the calculation could be complex. So it is not certain that this assumption is valid for our problem.
This theorem implies that $\sum_{i=1}^{\infty} cos^2(\theta_i) = \infty$ (with θ_i the angle between the direction d_i and the steepest descent direction $-\nabla f_i$) $\implies$ $\|\nabla f(x_i)\| \to 0 \equiv d_i$ does not get $\perp \nabla f(x_i)$ "too fast" $\Rightarrow$ convergence.
Zoutendijk's theorem states that if the search direction at each iteration points towards a descent direction and an appropriate line search method is used to determine the step size, the algorithm will converge to a point where the gradient is zero.

2.3 Convergence rate

In this chapter we analyze the efficiency, to do this we try to estimate the speed of convergence to the optimal value. The steepest descent method is the globally convergent

algorithm par excellence, but it is rather slow in practice. To get an idea of the convergence rate we consider the the case in which the objective function is quadratic such as:

$$f(x) = \frac{1}{2}x^T Px - b^T x.$$

In the tail of the convergence process, we are very close to the optimal, so the function appears almost indistinguishable from the second-order model. Thus, the result that applies to quadratic functions applies in general nonlinear objective functions.

Let's consider the following result [[5], Theorem 3.4]

Theorem 2.3 *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, and that the iterates generated by the steepest-descent method with exact line searches converge to a point x^* at which the Hessian matrix $\nabla^2 f(x^*)$ is positive definite. Let r be any scalar satisfying*

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1\right)$$

where $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are the eigenvalues of $\nabla^2 f(x^*)$. Then for all k sufficiently large, we have

$$f(x_{k+1}) - f(x_*) \leq r^2[f(x_k) - f(x_*)]$$

The theorem 2.3 shows that convergence of the steepest descent method can be very slow, even when the Hessian is moderately well conditioned, e.g. if the condition number ($k(P) = \lambda_n/\lambda_1$) is 800, $f(x_1) = 1$, and $f(x_*) = 0$, 2.3 suggests that the function value will still be about 0.08 after one thousand iterations of the steepest descent method with exact line search. When the condition number is larger, e.g. 1000 or more the gradient method is so slow that it is useless in practice. The starting point or initialization of the algorithm can also affect its convergence. If the initial point is close to the minimum, the algorithm may converge faster. However, if the initial point is far away from the minimum or in a region with a high gradient, it may take more iterations for the algorithm to converge.

To address the slow convergence of the steepest descent algorithm, various modifications and extensions have been developed, such as accelerated gradient methods (e.g., Nesterov's accelerated gradient descent), conjugate gradient methods, and quasi-Newton (e.g., BFGS which we will see in the next chapter) or others. These methods aim to improve the convergence rate by considering additional information about the function or its derivatives.

2.4 Complexity

In this paragraph, we will discuss the complexity of the steepest descent algorithm. An advantage of the standard gradient descent is that it requires the calculation of the gradient of the function but not of the second derivatives. However, it can be extremely slow on difficult problems. The relevant operations for the cost of the algorithm are the calculation

of $f(x)$ and $\nabla f(x)$ and the cost for the Inexact line search.

The complexity of $f(x)$:

The operations required to calculate $f(x) = -\frac{x^T Q x}{x^T x}$ are: the calculation of $Q = A^T A$ which we perform only once at the beginning and which requires $O(mn^2)$, $x^T Q$ which requires $O(n^2)$, $(x^T Q)x$ which is also $O(n^2)$ ad $x^T x$ which requires $O(n)$. So at each iterarion we spend $O(n^2)$

The complexity of $\nabla f(x)$:

To calculate$\nabla f(x) = \frac{2xf(x)}{x^T x} - \frac{2Qx}{x^T x}$ we can keep the partial results we obtained for $f(x)$. The only operation left to do is $xf(x)$ which takes $O(n)$ at each iteration.

The cost for the LS:

The calculation of the derivative of topography requires a series of matvec operations so the cost is again $O(n^2)$.

In conclusion we can say that the complexity of an iteration of the Steepest Descent direction algorithm is $O(n^2)$. the only case in which the complexity decreases is if Q is a sparse matrix since there are several optimization techniques to efficiently perform the matvec operation with sparse matrices.

3 BFGS

In this section, we'll discuss the BFGS, a quasi-Newton algorithm.

This method is a second-order optimization algorithm and it uses an inverse Hessian matrix but, unlike the Newton methods, the quasi-Newton algorithms approximate it using the gradient. Indeed, we don't need to compute exactly the matrix at each step so the calculation is less expensive than the Newton methods.

BFGS uses the Hessian matrix to determine both the direction and the step size to move to minimize our function.

We introduce the general outline of the algorithm:

Algorithm 2 BFGS

Given starting point x_0, convergence tolerance $\epsilon > 0$, inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$ **do**
  Compute search direction $p_k = -H_k \nabla f_k$;
  Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed using the exact line search in 1.3.4;
  Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
  Compute H_{k+1} by means of $H_{k+1} = H_k + \rho_k((1 + \rho_k y_k^T H_k y_k) s_k s_k^T - (H_k y_k s_k^T + s_k y_k^T H_k))$;
  $k \leftarrow k + 1$
end while

where $\rho_k = \frac{1}{y_k^T s_k}$, $y_k^T s_k$ represent the curvature condition applying the Wolfe condition because the inequality $s_k^T y_k > 0$ will not always hold for non-convex functions.

Consequently, the strong Wolfe condition allows guaranteeing to hold the condition. In-

deed, we have

$$\begin{aligned} |\nabla f_{k+1}^T p_k| &\leq c_2 |\nabla f_k^T p_k| \\ \iff |f_{k+1}^T| \frac{|s_k|}{|\alpha_k|} &\leq c_2 |\nabla f_k^T| \frac{|s_k|}{|\alpha_k|} \\ \iff |\nabla f_{k+1}^T s_k| &\leq c_2 |\nabla f_k^T s_k| \\ \iff |y_k^T s_k + \nabla f_k^T s_k| \leq |y_k^T s_k| + |\nabla f_k^T s_k| &\leq c_2 |\nabla f_k^T s_k| \\ \iff |y_k^T s_k| &\leq (c_2 - 1) |\nabla f_k^T s_k| \\ \iff |y_k^T s_k| &\leq (c_2 - 1) |\nabla f_k^T \alpha_k p_k| \end{aligned} \tag{14}$$

This is quite similar to using the Wolfe condition since $c_2 < 1$ and p_k is a descent direction the second member is positive, except now the directional derivative continuing in the same direction cannot be too positive, in any case, the $y_k^T s_k$ holds as the Wolfe condition. In [[4], p.2] tells the condition $s_k^T y_k > 0$ is guaranteed to hold if the stepsize α_k is determined by the exact line search

$$f(x_k + \alpha_k d_k) = min_{\alpha > 0} f(x_k + \alpha d_k)$$

3.1 Properties

The BFGS algorithm has self-correcting properties, indeed if the matrix H_k incorrectly estimates the curvature in the function and this bad estimate slows down the iteration, the Hessian approximation will tend to correct itself within a few steps. These properties hold only when it is applied by an adequate line search, especially with Wolfe conditions. In our study, we chose to implement the exact line search defined in the chapter 1.3.4.
In addition, we have already shown in the same chapter that the line search satisfies the Wolfe conditions.

Lemma 3.1 *If the BFGS with the strong Wolfe condition line search is applied to a continuously differentiable function f that it is bounded below, and if there exists a constant $M > 0$ such that*

$$\frac{\|y_k\|^2}{y_k^T s_k} \leq M \quad \forall k \tag{15}$$

then

$$\liminf_{k \to \infty} \|g(x_k)\| = 0$$

This lemma is considering the study of [[6], p. $53 - 57$], we can also notice that if f is twice continuously differentiable and uniformly convex, then (15) always holds. Therefore, the global convergence of the BFGS method follows from Lemma 3.1 immediately. The same theory will be applied using the exact line search in the next chapters, but still with the convex objective functions.

3.2 Convergence

BFGS guarantees global convergence for the convex objective functions, unfortunately, our function is non-convex as we already discussed in section 1.3.2. Then we will expect that the algorithm could not reach the convergence with problem 1.2.
Indeed, Lemma 3.1 where f is nonconvex, it seems difficult to guarantee 15. In addition, in the [[5], Theorem 6.5], the assumption is not satisfied so we can't apply the theorem to guarantee global convergence.
If we apply the following update:

$$H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

defined by the paper [[4], p.4], describes proof that it guarantees global convergence without the function being convex. Unfortunately, this update is not efficient for the unconstrained minimization because the cost of the step computation becomes $O(n^3)$, so we decided to use the H_{k+1} update written on the Alg.2.
Considering the assumption in [[6], Assumption A] the function has to the Lipschitz continuity which we have already proved in section 1.3.4, then the theorem of [[6], Theorem 3.3]:

Theorem 3.2 *Let [[6], Assumption A] holds and the result of 2 with α_k being determined by Wolfe line search then*

$$\liminf_{k\to\infty} \|g(x_k)\| = 0$$

holds.

The full proof can be found on [[6], p. $10 - 11$].
On the contrary, BFGS applies a superlinear convergence rate to general nonlinear (so non-convex) objective functions; in other words, if the algorithm gets closer to the optimal solution, the rate of convergence can accelerate, leading to faster progress toward the minimum. Firstly, as for the standard gradient descent algorithm, we already discussed that our function is Lipschitz continuous.
The $\nabla^2 f(x)$ is defined in the domain $\mathbb{R}^n_{\neq 0}$ so we can conclude that f is twice continuously differentiable, then we can apply the following theorem:

Theorem 3.3 *Suppose that f is twice continuously differentiable and that the iterates generated by the BFGS algorithm converge to a minimizer x^* at which [[5], Assumption 6.2] holds. Suppose also that $\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty$ holds. Then x_k converges to x^* at a superlinear rate.*

The full proof can be found on [[5], p. $158 - 160$].
The [[5], Assumption 6.2], as we said in chapter 2.2, is not certain that holds for our problem safely.
We conclude that the algorithm could converge globally with our function at a superlinear rate.

3.3 Complexity

The complexity of the Algorithm 2 at each iteration is $O(n^2)$, indeed the main operation is the update of the approximation of the Hessian matrix which is $O(n^2)$ because we perform the matrix multiplication between a matrix and a vector. In addition, we have to consider the matrix additions and scalar multiplications which have $O(n^2)$ as well.
Lastly, we have to add the cost of $f(x)$, $\nabla f(x)$, and the line search which we have already computed in section 2.4.

4 Experiments

In this section, we will look at the performance results of the algorithms for our objective function.
We have generated some random matrices, to test the performance of our algorithms, with different sizes and densities, in which each element belongs to an interval, e.g:

- 5 matrices of size 1500×500 and density about equal to 1.0
- 5 matrices of size 1500×50 and density about equal to 1.0
- 5 matrices of size 1000×250 and density about equal to 1.0
- 5 sparse matrices with sizes of 1000×250 and density about equal to 0.1.
- 5 sparse matrices with sizes of 1000×100 and density about equal to 0.1.
- 5 sparse matrices with sizes of 1000×25 and density about equal to 0.1.
- a badly ill-conditioned matrix using a Hilbert matrix with a size of 1000×1000.

For our experiments, we have considered some algorithms' parameters such as the maximum number of iterations and the accuracy in the stopping criteria, epsilon.
In detail, we have tuned:

- maximum number of iterations equal to 500
- stopping criteria epsilon equal to $1e-6$ and equal to $1e-4$
- minimum step size equal to $1e-16$.
- starting vector is chosen randomly using the same vector in both algorithms.

To evaluate the performances of our algorithms, we have computed the relative errors between the values of norms computed by numpy library, x, and the norm approximated by our algorithms, $\tilde{x}$:

$$err_r := \frac{||\tilde{x} - x||}{||x||} \tag{16}$$

Below, the plots show the performance comparing the trends between SGD and BFGS using the geometric mean between the different matrices with the same properties as we said above. We use the geometric mean because it is more accurate and effective than the arithmetic one and it is defined as

$$err_{mean} = (\prod_{i=1}^{n} err_r)^{(1/n)}$$

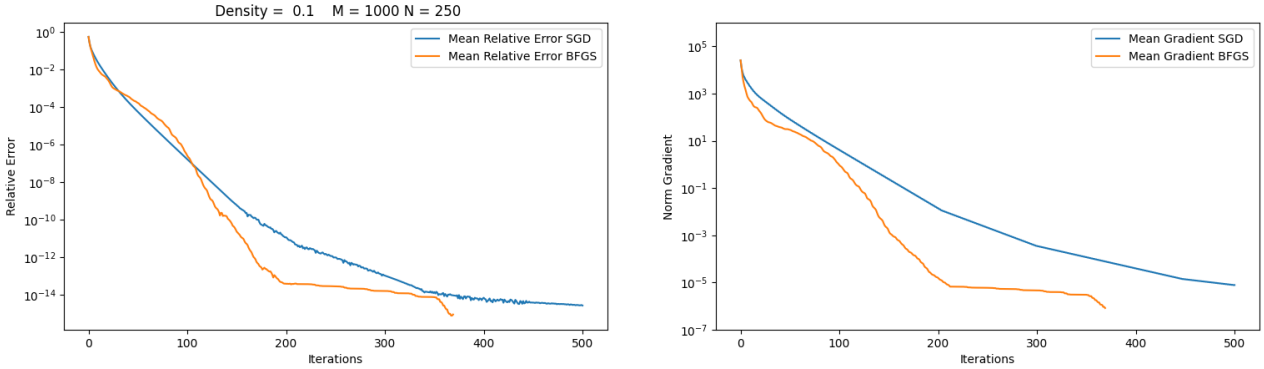

Figure 1: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-6**.

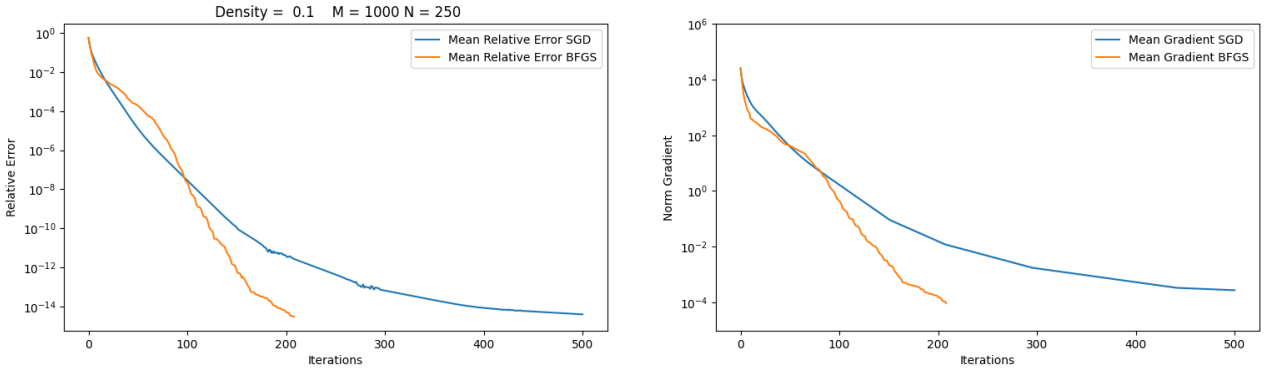

Figure 2: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

In both Figure 1 and 2 we can notice that both algorithms tend to be more unstable with stopping criteria equal to $1e{-}6$, especially in the last iterations, compared to one with $1e{-}4$. Using $\epsilon = 1e - 6$ the gradient norm continues to go down but without improvements in the function.

So we focus our experiments using $\epsilon = 1e - 4$, in Figure 3 and 4 we can notice that both algorithms have different behaviors depending on the size of matrices, indeed we have more fluctuations in larger matrices than in small ones.

In general, we can notice the BFGS algorithm approximates the solution better than SGD where it doesn't often reach the optimal compared to BFGS. In addition, we can see

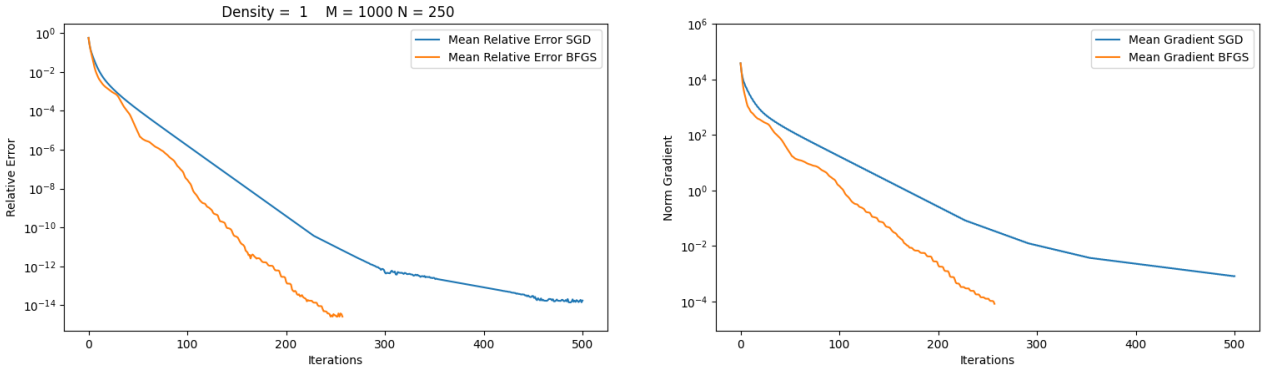

Figure 3: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

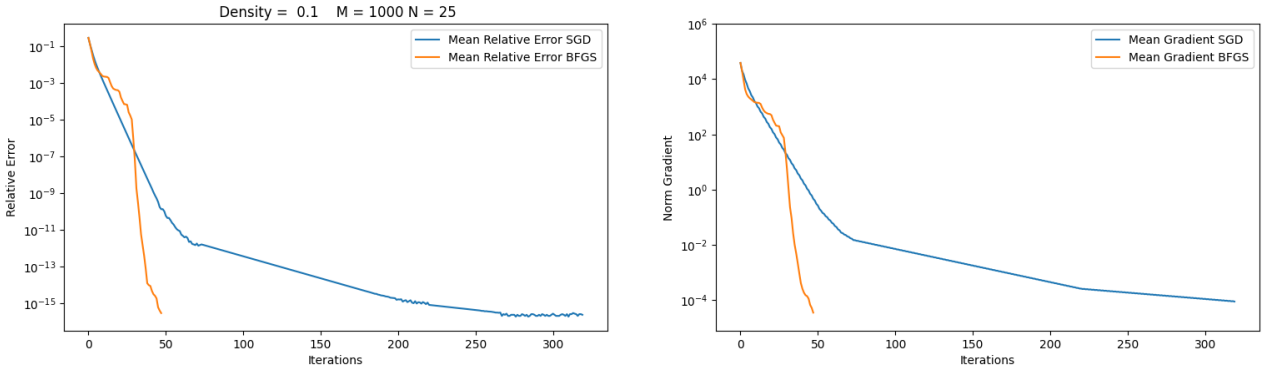

Figure 4: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

the SGD runs more iterations than BFGS in sparse matrices with a density equal to 0.1.

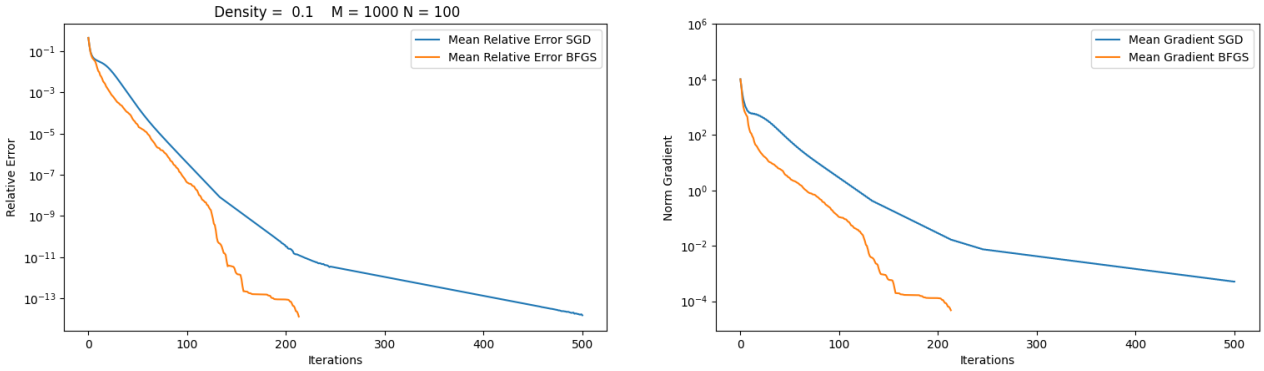

Figure 5: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

Figure 6 shows the results of the Hilbert matrix and both algorithms achieved a good estimation reaching the optimum. In addition, we can see that BFGS finds the optimal solution with one more iteration than SGD.

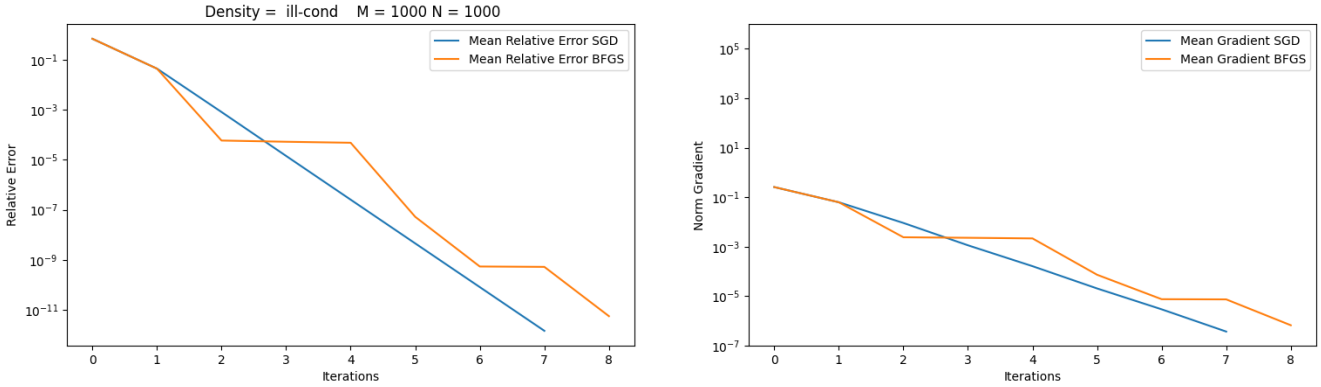

Figure 6: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

In general, both algorithms approximate the solution quite well, especially the BFGS, which did not perform so many iterations regarding the SGD.
The time execution of the two algorithms is shown in Table 2, generally, we can notice the SGD performs very slightly faster than BFGS because the BFGS has to do several approximations of the Hessian matrix during its running.

But for the sake of completeness, we test the algorithms using another stopping criteria to see their behavior. We control if the relative gap 16 is close to zero which means the algorithms are unable to reduce the function, with this new condition the algorithms reach the optimal with a relative gap less and equal to $1e-15$. In Figures 8 and 9 we can see the trend of algorithms is more stable than previous results.

Matrix size	Density	Average Relative Error	
		SGD	BFGS
1000×250	1.0	1.765×10^{-14}	2.629×10^{-15}
1000×250	0.1	3.848×10^{-15}	2.958×10^{-15}
1000×100	0.1	1.523×10^{-14}	1.311×10^{-14}
1000×25	0.1	2.333×10^{-16}	2.885×10^{-16}
Hilbert		1.475×10^{-12}	5.676×10^{-12}

Table 1: Comparing average relative errors on matrices with the same size and different densities with stopping criteria equal to 1e-4.

Matrix	Density	Time Execution (seconds)	
		SGD	BFGS
1000×250	1.0	2.030×10^{-1}	5.606×10^{-1}
1000×250	0.1	7.781×10^{-2}	2.431×10^{-1}
1000×100	0.1	6.611×10^{-2}	7.595×10^{-2}
1000×25	0.1	2.622×10^{-2}	9.733×10^{-3}
Hilbert		5.115×10^{-2}	5.335×10^{-2}

Table 2: Comparing average running times on matrices with the same size and different densities and the Hilbert matrix using a stopping criteria equal to (**1e-4**).

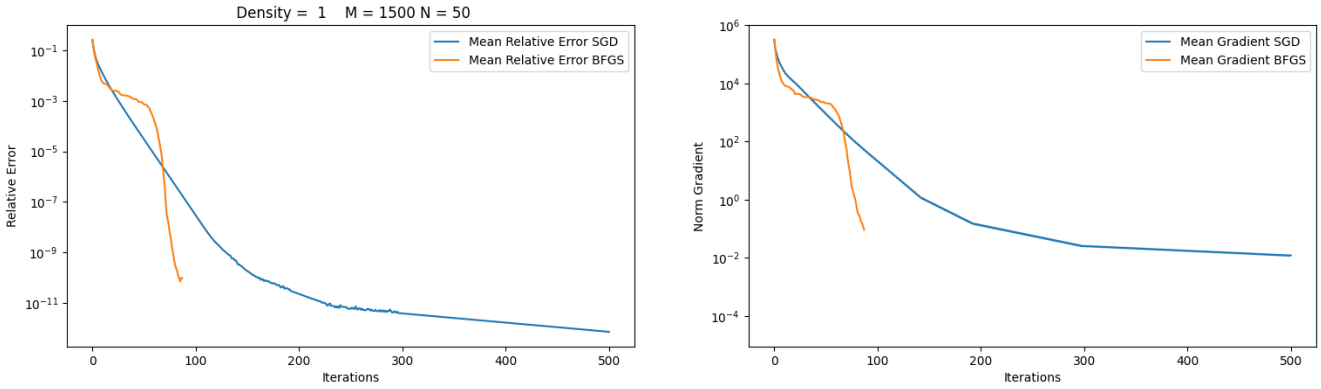

Figure 7: Relative errors plot (**left**) and gradients plot (**right**) with the stopping criteria equal to **1e-4**.

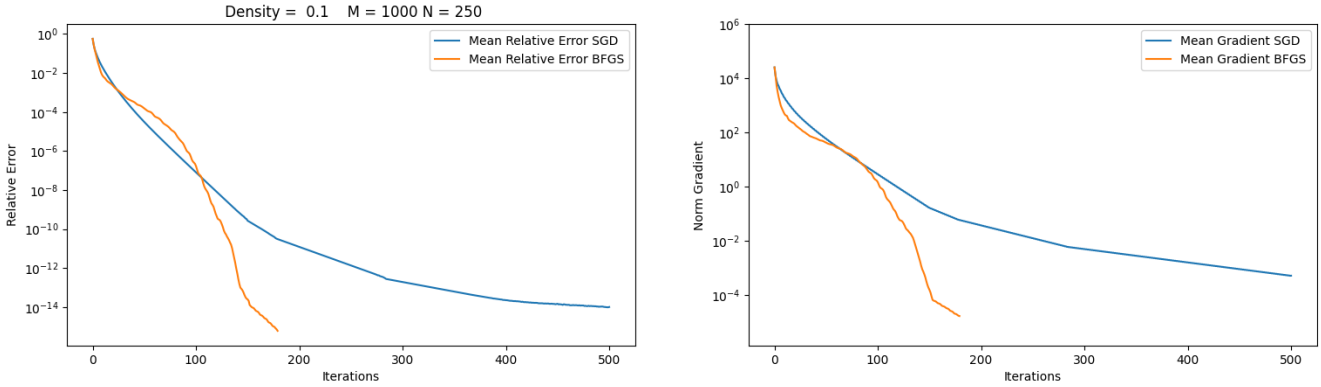

Figure 8: Relative errors plot (**left**) and gradients plot (**right**) using $\frac{||\tilde{x}-x||}{||x||} \leq 1e-15$ as stopping criteria.

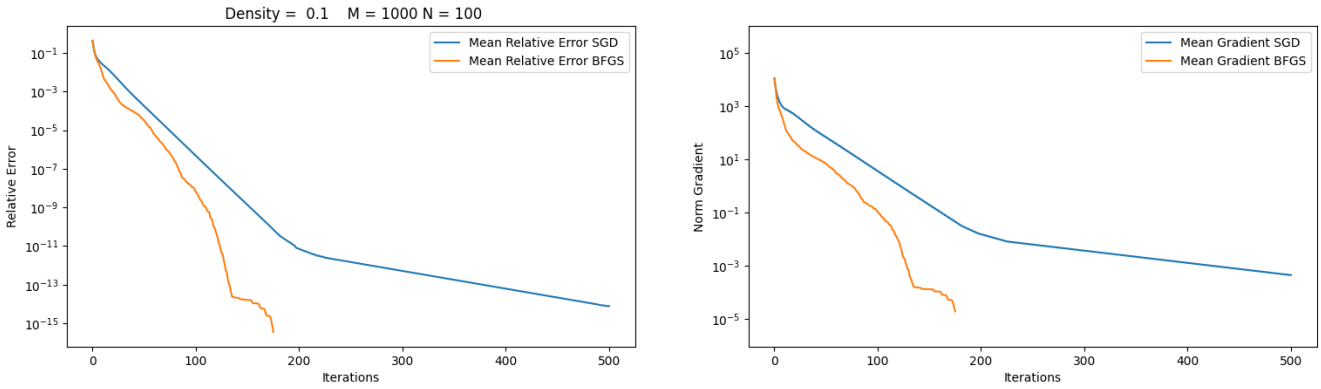

Figure 9: Relative errors plot (**left**) and gradients plot (**right**) using $\frac{||\tilde{x}-x||}{||x||} \leq 1e-15$ as stopping criteria.

5 Conclusions

As we have seen, the SGD algorithm is, in general, simple to implement and efficient from a computational point of view but can converge slowly. BFGS generally converges faster than SGD but can be computationally more expensive due to Hessian matrix approximations.

The results obtained do not differ from what we expected. Based on our results we can conclude that BFGS approximates the solution better and performs fewer iterations, however, it requires more execution time than SGD for larger-sized matrices.

References

[1] D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific optimization and computation series. Athena Scientific, 2003.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[3] J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

[4] D.-H. Li and M. Fukushima. On the global convergence of the bfgs method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11(4):1054–1064, 2001.

[5] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2e edition, 2006.

[6] M. Powell, E. Atomic Energy Research Establishment (Harwell, E. C. S. Atomic Energy Research Establishment (Harwell, and S. Division. *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*. CSS (Series). UKAEA, Harwell Atomic Energy Research Establishment, 1975.

[7] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, 1997.