POLITECNICO DI TORINO, TURIN, ITALY

Unconstrained Optimization
Numerical Optimization for Large Scale Problems

January 30, 2024

AILLA Sana
s321630
BAYAT Erfan
s328390

Contents

Abstract

In this work, we present and describe the Modified Newton method based on the hessian modification, utilized for the resolution of Large Scale Optimization Problems. The chosen method is employed both with and without finite differences to approximate the gradient and Hessian of selected problems, accompanied by the Steepest Descent with backtracking strategies.

The implemented code for this project can be found in the **GitHub repository**: GitHub Repository.

Keywords*: Modified Newton Method, Hessian modification, Finite Difference, Steepest Descent, Backtracking, Rosenbrock Function*

1 Introduction

In this homework, we will report, compare and comment the result of the implemented methods applied to Rosenbrock function in $\mathbb{R}^2$ (2.1) and 3 optimization problems choosen from the aritcle [**1**]:

- Problem 1 : Chained Rosenbrock Function (2.2)
- Problem 25 : Extended Rosenbrock Function (2.3)
- Problem 75 : Problem 103 (2.4) from the article [**1**]

2 Problem Overview

The general optimization problem addressed in this work is an unconstrained optimization of the form:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \quad \text{where } f:\mathbb{R}^n \to \mathbb{R} \text{ is a numeric function.}$$

and this is using the methods mentionned in (3) where the function f to minimize will be described for each adressed problem in the next sections. In addition, for each problem a starting point is already provided.

2.1 Rosenbrock Function in $\mathbb{R}^2$

Recalling the expression of the Rosenbrock function [**2**] in $\mathbb{R}^2$:

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

Then, the gradient of f is given by:

$$\nabla f(x_1, x_2) = \begin{bmatrix} 400x_1^3 - 400x_1x_2 + 2x_1 - 2 \\ 200(x_2 - x_1^2) \end{bmatrix}$$

And the Hessian matrix $\nabla^2 f$ is given by:

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

For this problem, we tested our methods using two differents starting points $x_0 = (1.2, 1.2)$ and $x_0 = (-1.2, 1)$ in 4.1.

2.2 Problem 1

The Rosenbrock function is a popular test problem in the optimisation literature [**3**] due to its challenging features: its minimum is located at the bottom of a long and narrow parabolic valley, also known as the banana valley and it has many variants. This problem is the chained Rosenbrock function. It is expressed as :

$$F(x) = \sum_{i=2}^{n} f_k(x)$$

where $f_k(x)$ is defined as:

$$f_k(x) = 100 \cdot (x_k - x_{k-1}^2)^2 + (1 - x_{k-1})^2$$

The gradient of the function F is given by three cases:

$$\frac{\partial F}{\partial x_i} = \begin{cases} -400x_1(x_2 - x_1^2) - 2(1 - x_1) & \text{if } i = 1 \\ 200(x_{i-1} - x_{i-2}^2) - 400x_i(x_{i+1} - x_i^2) - 2(1 - x_i) & \text{if } 2 \le i \le n-1 \\ 200(x_n - x_{n-1}^2) & \text{if } i = n \end{cases}$$

Each component of the gradient depends only on the terms x_i, x_{i-1}, and x_{i+1}. Thus, the Hessian matrix will be a **tridiagonal** sparse matrix such that :

$$\begin{cases} \frac{\partial^2 F}{\partial x_i^2} = 1200x_1^2 - 400x_2 + 2 & \text{if } i = 1 \\ \frac{\partial^2 F}{\partial x_i \partial x_{i-1}} = -400x_{i-1} & \text{for } i = 2,3,\ldots,N-1, \\ \frac{\partial^2 F}{\partial x_i \partial x_i} = 202 + 1200x_i^2 - 400x_{i+1} & \text{for } i = 2,3,\ldots,N-1, \\ \frac{\partial^2 F}{\partial x_i \partial x_{i+1}} = -400x_i & \text{for } i = 2,3,\ldots,N-1, \\ \frac{\partial^2 F}{\partial x_i^2} = 200 & \text{if } i = n \end{cases}$$

The starting point of the optimization problem is $x \in \mathbb{R}^n$ given by:

$$x_i = \begin{cases} -1.2 & \text{if } i \equiv 1 \pmod 2 \\ 1.0 & \text{if } i \equiv 0 \pmod 2 \end{cases}$$

We tested this problem for $n = 10^3$, results are reported in (4.2)

2.3 Problem 2

This problem is also a variant of Rosenbrock function called chained rosenbrock such that :

$$F(x) = \frac{1}{2}\sum_{i=2}^{n}(f_k(x))^2$$

where $f_k(x)$ is defined as :

$$f_k(x) = \begin{cases} 10(x_k^2 - x_{k+1}), & \text{if } k \text{ is odd} \\ x_{k-1} - 1, & \text{if } k \text{ is even} \end{cases}$$

The analytical expression of the gradient of F is :

$$\nabla F_k(x) = \begin{cases} 200x_k^3 - 200x_k x_{k+1} + x_k - 1 & \text{if } k \text{ is odd} \\ 100(x_k - x_{k-1}^2) & \text{if } k \text{ is even} \end{cases}$$

Its also clear that each component of the gradient depends only on the terms x_i, x_{i-1}, and x_{i+1}. Thus, the Hessian matrix will be a $(2x2)$ bloc diagonal (a **tridiagonal**), with every $(2x2)$ block given by :

$$100\begin{bmatrix} 6x_{i-1}^2 - 2x_i + 0.01 & -2x_{i-1} \\ -2x_{i-1} & 1 \end{bmatrix} \quad \text{for } i = 2,4,...,n$$

The starting point of the optimization problem is $x \in \mathbb{R}^n$ given by:

$$x_i = \begin{cases} -1.2 & \text{if } i \equiv 1 \pmod 2 \\ 1.0 & \text{if } i \equiv 0 \pmod 2 \end{cases}$$

We tested this problem for $n = 10^3$, results are reported in (4.3)

2.4 Problem 3

In this problem, the function F is also defined as :

$$F(x) = \frac{1}{2}\sum_{i=1}^{n}(f_k(x))^2$$

where $f_k(x)$ is defined as :

$$f_k(x) = \begin{cases} x_k - 1 & \text{if } k = 1 \\ 10(k-1)(x_k - x_{k-1})^2 & \text{if } 1 < i \leq n \end{cases}$$

The gradient of F is defined by :

$$\nabla F_k(x) = \begin{cases} x_k - 1 + 200(x_k - x_{k+1})^3, & \text{if } k = 1 \\ 200k^2(x_k - x_{k+1})^3 - 200(k-1)^2(x_{k-1} - x_k)^3, & \text{if } 1 < k < n \\ -200(k-1)^2(x_{k-1} - x_k)^3, & \text{if } k = n \end{cases}$$

The hessian matrix H of F is a tri-diagonal matrix for which the elements are defined by :
- For diagonal elements

$$\frac{\partial^2 F}{\partial x_k^2} = \begin{cases} 1 + 600(x_1 - x_2)^2, & \text{if } k = 1 \\ 600(k-1)^2(x_{k-1} - x_k^2)^3 + 600k^2(x_k - x_{k+1})^2, & \text{if } 1 < k < n \\ 600(k-1)^2(x_{k-1} - x_k)^2, & \text{if } k = n \end{cases}$$

- For upper and lower diagonal elements :

$$\frac{\partial^2 F}{\partial x_k \partial x_{k+1}} = \frac{\partial^2 F}{\partial x_{k+1} \partial x_k} = 600(k-1)^2(x_{k-1} - x_k)^2, \text{for k = 1, 2, ... ,n - 1}$$

The starting point of the optimization problem is $x \in \mathbb{R}^n$ given by:

$$x_i = \begin{cases} -1.2 & \text{if } 1 \leq i < n \\ -1.0 & \text{if } i = n \end{cases}$$

We tested this problem for $n = 10^3$, results are reported in (4.4)

3 Methods

The main chosen method to adress to problems mentioned in (2) is a variant of the famous Newton method, specifically tailored for iterative optimization. The Newton descent method is an iterative optimization technique that, starting from a given vector $\mathbf{x}_0 \in \mathbb{R}^n$, computes a sequence of vectors $\{\mathbf{x}_k\}_{k\geq 0}$ characterized by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k),$$

where α is the steplength and the descent direction $\nabla f(\mathbf{x}_k)$ is the solution of the linear system

$$H_f(\mathbf{x}_k)\mathbf{p}_k = -\nabla f(\mathbf{x}_k),$$

and ∇f and H_f are the gradient and the Hessian matrix of f, respectively.

3.1 Modified Newton Method

In many cases, the Hessian of F is not guaranteed to be positive definite, and thus, p_k is not guaranteed to be a descent direction. To address this issue, we introduce the Modified Newton method, which incorporates a modification of the Hessian matrix of f [**7**].

At each iteration, we correct the Hessian $H_f(\mathbf{x}_k)$ by adding a scalar matrix $E_k = \tau_k \mathbf{I}$, such that $B_k = H_f(\mathbf{x}_k) + E_k$ is symmetric positive definite (SPD). We then solve the linear system $B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$, where $\mathbf{p}_k$ is the new "Modified Newton direction." The updated iteration is given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$.

The value of τ_k (and thus the value of E_k) is chosen to ensure that the resulting matrix B_k is well-conditioned, sufficiently positive definite ($\lambda_{\min}(B_k) \geq \delta > 0$, where $\delta = \sqrt{\epsilon_m}$), and close enough to $H_f(\mathbf{x}_k)$. Theoretical results have shown that $\tau_k = \max(0, \delta - \lambda_{\min}(H_f(\mathbf{x}_k)))$. However, this solution is not efficient in the case of large-scale problems, where computing the minimum eigenvalue at each iteration may be computationally expensive. In practice, we follow this alternative approach to compute B_k and its Cholesky decomposition that can be useful to solve the linear system.

Algorithm 1 Compute B_k and Adjust τ_k

```
1:  Set β = ‖∇²f(x^k)‖_F
2:  for iteration k do
3:      if diag(H_f(x^k)) > 0 then
4:          τ_0 = 0
5:      else
6:          Set τ_0 = β/2
7:          for j = 0, 1, ... do
8:              Try Cholesky decomposition on B_k: RR^T = B_k
9:              if Cholesky decomposition succeeded then
10:                 τ_j is good enough
11:                 Break
12:             else
13:                 τ_{j+1} = max(2τ_j, β/2)
```

In practice, we chose $\beta = 1 \times 10^{-3}$, and at each iteration, if the decomposition fails, we changed the factor 2 to 10 in line 13: $\tau_{j+1} = \max(10\tau_j, \beta/2)$ [**5**]. This is called Cholesky Hessian modification, and there exist other types of modifications cited here [**4**].

3.2 Modified newton Method with Finite differences

In the second method, we will use the same modified Newton method described in the previous section, but instead of defining the gradient and Hessian of the function as functions, we will approximate them using finite differences. In fact, all chosen problems (2) can be proven to be C^2, thus we can use the following expressions to approximate their gradient and Hessian.

Gradient Approximation

Assuming that $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ is $C^1(\Omega)$, we will approximate the gradient of f at $x_0 \in \Omega$ with the following methods:

Centered Finite Differences (C-FD) Our thorough evaluation showed that the centered difference method achieved more accurate overall behavior compared to forward difference methods. This conclusion is also

supported by state-of-the-art techniques.

$$f_{x_i}(x_0) = \frac{f(x_0 + he_i) - f(x_0 - he_i)}{2h}, \quad \forall i = 1, \ldots, n;$$

Hessian Approximation

Assuming that $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ is $C^2(\Omega)$ (i.e., symmetric Hessian for each $x \in \Omega$), we can approximate the second-order derivatives of f at $x_0 \in \Omega$ with the following expressions:

Hessian's diagonal elements:

$$H_f^{(ii)}(x_0) = f_{x_i x_i}(x_0) = \frac{f(x_0 + he_i) - 2f(x_0) + f(x_0 - he_i)}{h^2}, \quad \forall i = 1, \ldots, n;$$

Hessian's non-diagonal elements:

$$H_f^{(ij)}(x_0) = f_{x_i x_j}(x_0) = \frac{f(x_0 + he_i + he_j) - f(x_0 + he_i) - f(x_0 + he_j) + f(x_0)}{h^2}, \quad \forall i, j = 1, \ldots, n,\ i \neq j.$$

3.3 Tridiagonal Hessian of chosen problems [2]

For example, for problem 25 (Chained Rosenbrock function) [2.2], F has a special structure that leads to a tridiagonal Hessian matrix. Breaking down the terms of $F(x) = \frac{1}{2}\sum_{k=1}^{n} f_k^2(x)$, into

$$f_k(x) = \begin{cases} 10(x_k^2 - x_{k+1}), & \text{if } k \text{ is odd} \\ x_{k-1}, & \text{if } k \text{ is even} \end{cases}$$

Observing each term f_k, we can discern the following:

1. The term $10(x_{k+1} - x_k)$ involves only x_k and x_{k+1}, contributing to both the diagonal and upper diagonal elements of the Hessian matrix.

2. The term $(x_{k-1})^2$ involves only x_{k-1}, contributing exclusively to the lower diagonal elements of the Hessian matrix.

Given that the function is a composite of these terms, the Hessian matrix is characterized by non-zero entries along its diagonal, lower diagonal, and upper diagonal, forming a tridiagonal structure. This implies a tridiagonal Hessian matrix. Extending this analysis, we deduce that the three identified problems share a consistent Hessian matrix structure, marked by tridiagonal "SPARSE" matrices.

Choice of h:

The choice of h is a critical factor in the accuracy and stability of finite difference approximations. In general, a smaller value of h will lead to more accurate approximations, but it will also increase the computational cost. Conversely, a larger value of h will lead to less accurate approximations, but it will also reduce the computational cost. A typical choice for h [**6**] that we used in our implemented approach is:

$$\begin{cases} h_{\text{grad}} = \sqrt{\epsilon_m} \\ h_{\text{hessian}} = \sqrt{h_{\text{grad}}} \end{cases} \quad \text{where } \epsilon_m \approx 10^{-16} \text{ is the machine precision}$$

3.4 Steepest Descent and Backtracking

In this section, we dive into the Steepest Descent method enhanced by the Backtracking strategy. The Steepest Descent method is an iterative optimization technique used to minimize a given function $f : \mathbb{R}^n \to \mathbb{R}$.

Starting from an initial vector $\mathbf{x}_0 \in \mathbb{R}^n$, the method computes a sequence of vectors $\{\mathbf{x}_k\}_{k\geq 0}$ characterized by the update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k),$$

where $\nabla f(\mathbf{x}_k)$ is the gradient of f at $\mathbf{x}_k$ and α represents a step length factor determined through our backtracking method.

At each iteration, we used a Backtracking strategy to find a suitable stepsize α_k, that satisfies :

- Armijo Condition :
$$f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^\top \nabla f(\mathbf{x}_k),$$
- Wolfe condition:
$$f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^\top \nabla f(\mathbf{x}_k),$$
where $c_1 \in (0,1)$

$$\nabla f(\mathbf{x}_{k+1})^\top \nabla f(\mathbf{x}_k) \geq c_2 f(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)) \quad c_2 \in (c_1, 1)$$

where $c_2 \in (c_1, 1)$

The backtracking strategy is used to ensure that the step length is not excessive and that there is a sufficient decrease in the objective function value. This is achieved by iteratively updating the step length until both Armijo and Wolfe conditions are satisfied. The Armijo condition ensures that the objective function value decreases at each step, while the Wolfe condition ensures that the gradient of the objective function is sufficiently reduced.

To find α_k, we start with a factor $\rho \in (0,1)$ and the initial value $\alpha_0 = 1$ is deliberate, specifically tailored for the Newton method we are employing., the algorithm iteratively decreases α_k by multiplying it by ρ until the Armijo and Curvature conditions are satisfied. The updating rule is given by $\alpha_{k+1} = \rho^t \alpha_k$, where t is the number of iterations. It is also important to mention that there is a maximum number of iterations, denoted as $B_{\max}$, which limits the algorithm's iteration process.

4 Results and comments

The reported results in the following sections are for this parameters (unless otherwie noted) :

Parameter	Value	Definition
kmax	1000	The maximum number of iterations
c_1	1e-4	Factor of the Armijo condition
ρ	0.4	Factor less than 1, used to reduce α
bt_{max}	100	Maximum number of steps allowed to update α
tolgrad	1e-6	Tolerance with respect to the norm of the gradient

- For the problems 2.2, 2.3, and 2.4, we will report the results in $\mathbb{R}^3$.
- The reported results encompass the iteration count for convergence (if converging), final gradient norm, objective function value at the minimum, computational time and the Mean Experimental Rate of Convergence (Mean RC).
- **N.B** : Using the preconditioning didn't improve the results, that's why we didn't report its results.

Experimental Rate of Convergence: The number of iterations needed to find a solution is closely related to the rate of convergence, which represents the speed at which the error diminishes as we approach the root. More precisely, we introduce the error in iteration n as $e_n = x_n - x^*$, where x^* is the exact solution. We then define the convergence rate q as

$$\|e_{n+1}\| \approx \|e_n\|^q.$$

The exponent q measures how quickly the error is reduced from one iteration to the next. The larger q is, the faster the error converges to zero, resulting in fewer iterations.

To estimate q, we can compute all the errors e_n using the following equation

$$q = \frac{\ln(\|e_{n+1}\|/\|e_n\|)}{\ln(\|e_n\|/\|e_{n-1}\|)}.$$

We will approximate the rate of convergence of the method using the mean of the rates of convergence over the sequence x_k. When we do not know the exact solution (as in the case of Problem 3), we can surrogate e_n by $\|x_k - x_{k-1}\|$ (add reference to the lectures).

4.1 Rosenbrock in $\mathbb{R}^2$

The table 1 contains the results of runing the two methods using two different starting points.

Starting point	$x_{01} = [1.2, 1.2]$		$x_{02} = [-1.2, 1.]$	
Method	**MNM**	**MNM C-FD**	**MNM**	**MNM C-FD**
Number of iterations	8	9	21	23
Mean RC	1.62	1.31	1.68	1.48
Execution Time (s)	7.10294×10^{-4}	1.08623×10^{-3}	2.17843×10^{-3}	3.28606×10^{-3}
$\|\|\nabla f(x_k)\|\|$	2.12502×10^{-9}	2.18216×10^{-9}	2.16075×10^{-9}	5.46229×10^{-10}
$f(x_k)$	4.41835×10^{-21}	3.81218×10^{-18}	2.45306×10^{-21}	2.38731×10^{-19}

Table 1: Comparison of Modified Newton methods applied to Rosenbrock function in $\mathbb{R}^2$ for different starting points

The Figures 1 and 2 show the objective function values over iterations, while Figures 3 and 4 depict the gradient norm of the implemented methods over iterations.

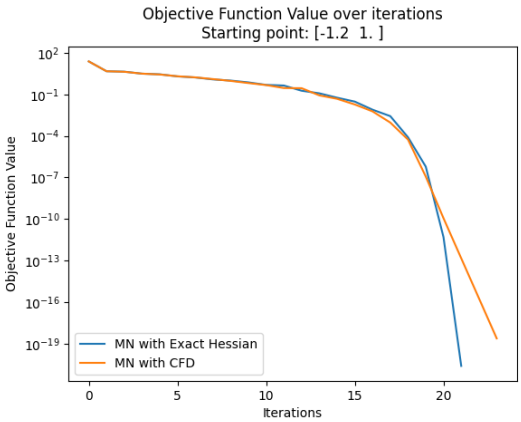

Figure 1: Objective Function Over iterations for x_{01}

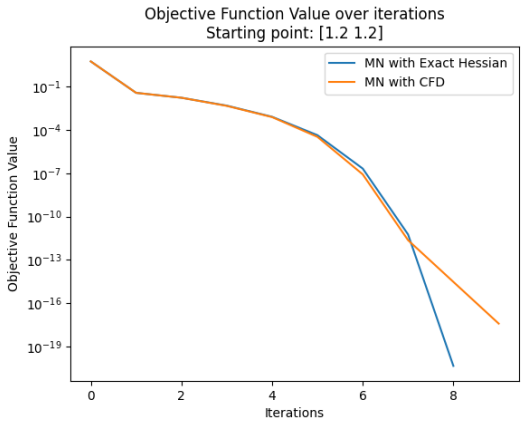

Figure 2: Objective Function Over iterations for x_{02}

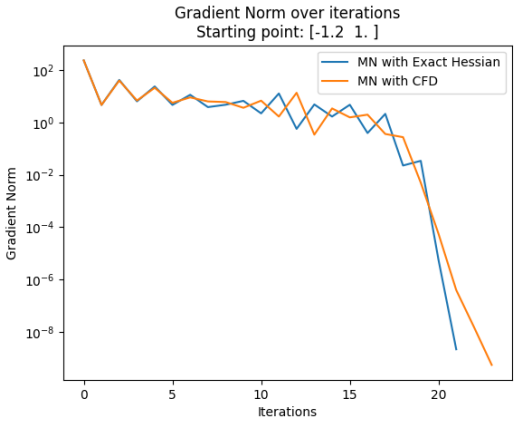

Figure 3: Gradient Norm Over Iterations for x_{01}

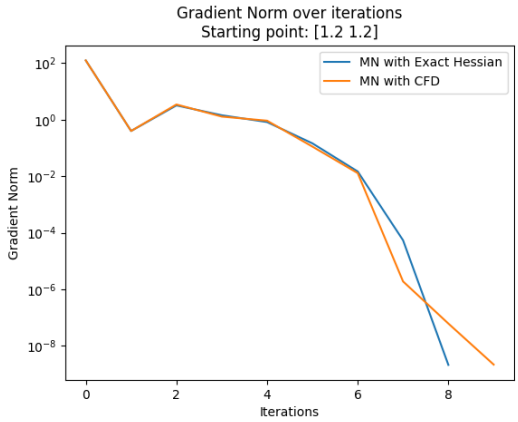

Figure 4: Gradient Norm Over Iterations for x_{02}

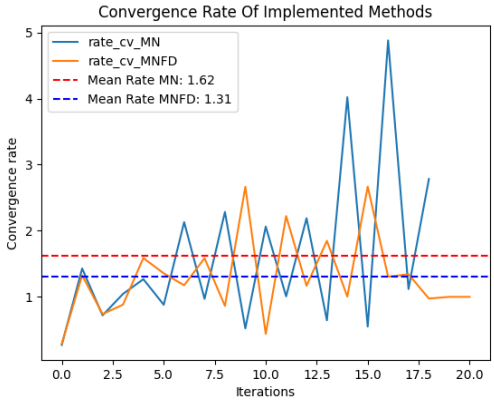

Figure 5: Experimental rate of Convergence for x_{01}

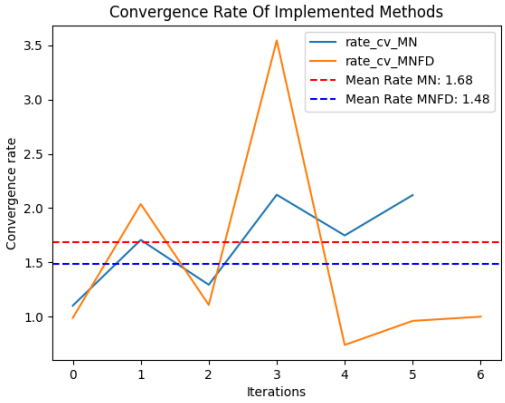

Figure 6: Experimental Rate of Convergence for x_{02}

Finally the Figure 5 and 6 present the Experimental rate of converge at iteration.

Comments : Based on the reported results and figures, we can make the following observations:

- Both methods converged to the optimal solution in a faster way, demonstrating a sufficient decrease in the gradient of the sequence values over iterations and the objective function. This holds true for both initial starting points, with faster convergence observed in both cases of the modified Newton method with exact analytical values (Blue curve).
- The behavior of the implemented method differs for $x_{01} = [-1.2, 1]$ and $x_{02} = [1.2, 1.2]$. Specifically, we observe that the overall behavior changes based on the starting points. The number of iterations required to reach convergence is higher for x_{01}, and the rate of convergence is lower compared to starting from x_{02}. This can be explained by the fact that x_{01} is farther from the optimal solution x^*; in fact, $\|x_{01} - x^*\| = \mathbf{2.2} > \|x_{02} - x^*\| = \mathbf{0.282}$.
- We can also notice that the experimental rate of convergence is slightly higher when starting from x_{02}, which can also help explain the difference in convergence vitesse.

4.2 Problem 1

For this problem, we chose $k_{\max} = 1600$ for the simple reason that, when using it, the implemnted methods converges within 1555 iterations. Therefore, if considering $k_{\max}$ as 1000, we can conclude that the methods are not converging.

	MNM with FD	MNM
Number of iterations	1555	1555
Mean RC	1.09	1.09
Execution Time (s)	1.45037×10^{3}	9.09595×10^{1}
$\|\|\nabla f(x_k)\|\|$	9.32484×10^{-7}	3.98603×10^{-12}
$f(x_k)$	3.98662	3.98662

Table 2: Comparison of Modified Newton methods applied to Problem 1

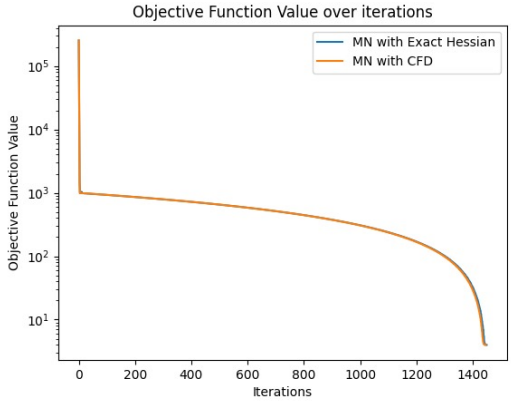

Figure 7: Objective Function Over Iterations over iterations

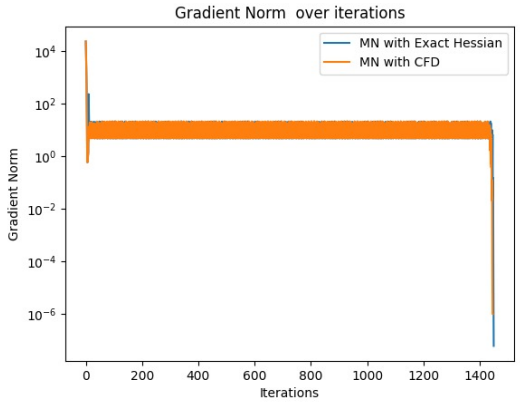

Figure 8: Gradient Norm Over Iterations

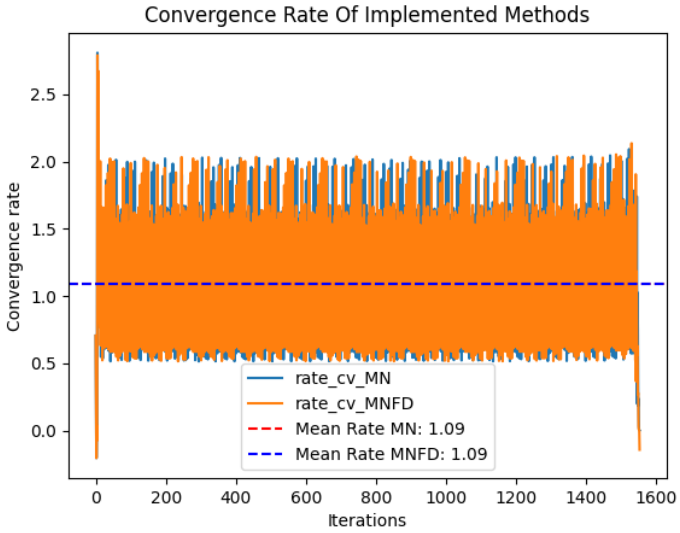

Figure 9: Experimental Rate Of convergence Over iterations

Comments : Based on the reported results and figures, we can make the following observations:

- Both methods converged to the optimal solution, although the method using finite differences required a larger number of iterations 2. Additionally, both methods exhibited approximately the same behavior. The longer computation time of the method using finite differences can be attributed to the additional computations and function calls needed to approximate the Hessian and the gradient at each point in the sequence.

- Based on Figure 7 and 8, we observe a decrease in the objective function and the gradient; however, a zigzag behavior is noticeable in the same region. This phenomenon may be attributed to the descent

direction found and the failure of the Hessian modification using Cholesky decomposition. Ultimately, the method achieves a sufficient decrease and convergence within 1555 iterations.

- It is also noteworthy that the experimental rate of convergence 9 appears to be low, possibly linear (≈ 1). This observation explains the gradual convergence towards the optimal solution.

4.3 Problem 2

In this section, we present the results of applying Modified Newton methods to address Problem 2.

	MNM	**MNM C-FD**
Number of iterations	17	12
Mean RC	2.27	1
Execution Time (s)	9.10022×10^{-1}	5.80218×10^{1}
$\|\|\nabla f(x_k)\|\|$	2.14936×10^{-12}	9.66061×10^{-9}
$f(x_k)$	7.66043×10^{-24}	7.55307×10^{-17}

Table 3: Comparison of Modified Newton methods applied to Problem 2

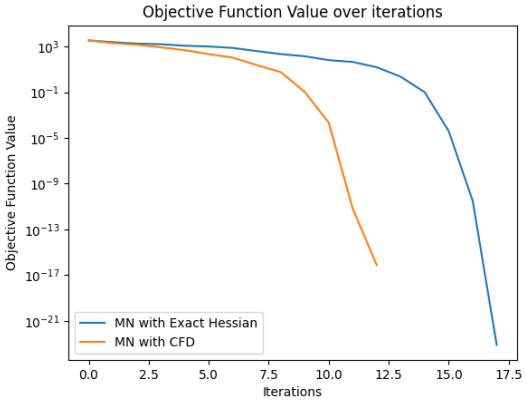

Figure 10: Objective Function Over Iterations

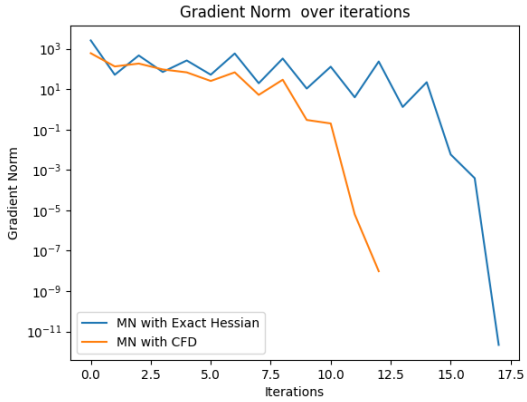

Figure 11: Gradient Norm Over Iterations

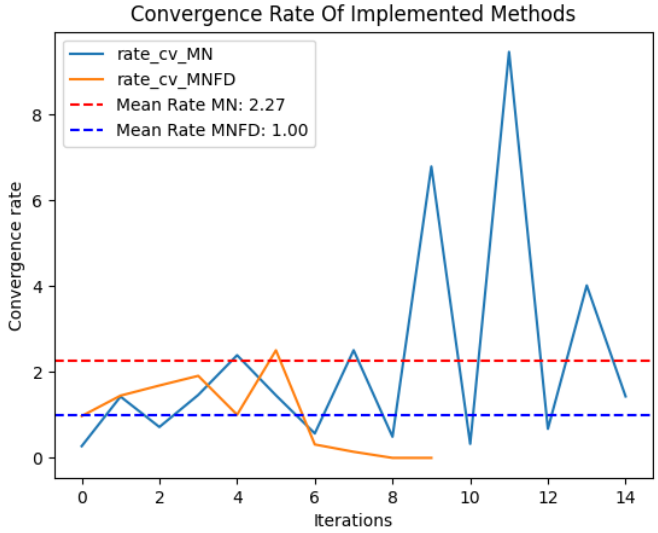

Figure 12: Experimental Rate Of convergence Over iterations

Comments : Based on the reported results and figures, we can make the following observations:

- Both methods successfully and very fately converged to the optimal solution, very feew iterations 4.
- As depicted in Figure 10, a clear decrease in the objective function over iterations is observed for both methods. The zigzag behavior in the same region may be attributed to the descent direction found and potential challenges in Hessian modification using Cholesky. However, both methods ultimately achieve a substantial decrease and converge within the specified number of iterations.
- The experimental rate of convergence for Problem 2 appears to be fast, as illustrated in Figure 12 (covering the first 8 iterations of MNM C-FD). The mean rate of convergence for MNM C-FD is quadratic, indicating a rapid and consistent approach towards the optimal solution. This observation aligns with the relatively quick convergence seen in the objective function and gradient norm plots.

4.4 Problem 3

- For this problem, a particular behavior is observed in the method using the exact analytical expressions of the gradient and the Hessian, while MNM with finite differences shows a notably fast and efficient convergence (13 and 14, 23 iterations vs. 528 iterations for MNM). This is further highlighted in Table 4.
- Due to the lack of information about the exact solution of this problem, the experimental rate of convergence was reported as mentioned in the section. The mean rate of convergence for MNM with finite differences (≈ 1.32) is consistent with the observed convergence behavior, as illustrated in Figure 12. On the other hand, the experimental rate of convergence of MNM with exact formulas exhibits an understandable but unusual behavior, possibly influenced by the surrogation.

	MNM	**MNM C-FD**
Number of iterations	528	23
Mean RC	-1.02	1.32
Execution Time (s)	3.46999×10^{1}	1.48288×10^{2}
$\|\|\nabla f(x_k)\|\|$	6.06026×10^{-6}	8.80361×10^{-6}
$f(x_k)$	1.62526	1.53061×10^{-10}

Table 4: Comparison of Modified Newton methods applied to Problem 3 without preconditioning

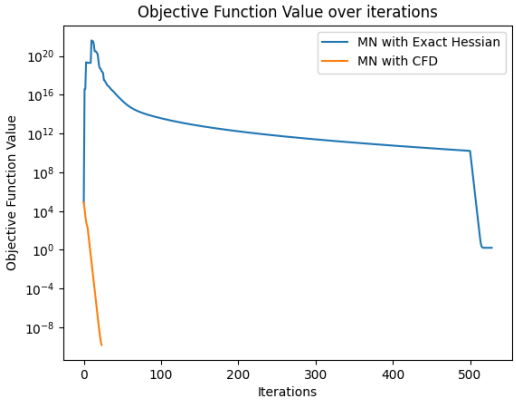

Figure 13: Objective Function Over Iterations

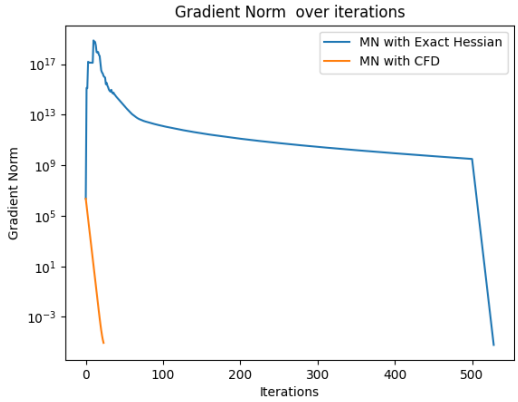

Figure 14: Gradient Norm Over Iterations

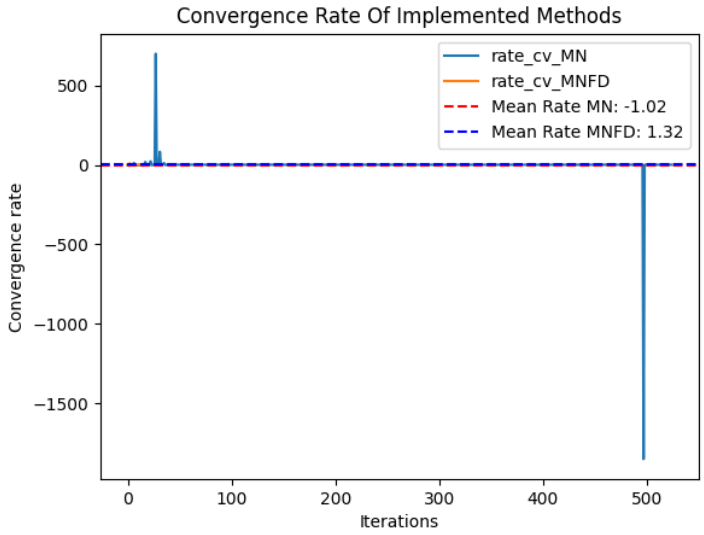

Figure 15: Experimental Rate Of convergence Over iterations

5 Conclusion and further work

In this study, we implemented two distinct unconstrained optimization methods using the **Python programming language** to address three different problems. Recognizing the intricacies of these problems, we introduced a tridiagonal Hessian approximation when employing finite differences to expedite the convergence process.

We achieved convergence of MNM - CFD in all the considered problems, although it required over 1000 iterations to converge for problem 1. Furthermore, even after reaching convergence, the ensemble approach we developed did not prove beneficial across the various problems. This encompassed additional techniques such as Wolfe conditioning, forward finite differences approximation, and preconditioning.

We can assert that the implemented methods rely on superlinear/quadratic local convergence, as evidenced by the computed experimental rate of convergence (except for Problem 3).

For future work, we propose investigating and addressing the challenges faced in achieving convergence for Problem 1 within a more efficient iteration limit. Additionally, exploring alternative ensemble approaches, such as the Hessian modification approach and preconditioning when the matrix is not symmetric positive definite (SPD), could provide valuable insights into overcoming the issues encountered during the failure of the Hessian modification for this problem. These avenues of exploration aim to enhance the overall robustness and efficiency of the optimization methods under consideration.

References

[1] Ladislav Luksan, "Test problems for unconstrained optimization," *Journal of Global Optimization*, vol. 27, no. 2, pp. 217–233, 2003.

[2] Wikipedia contributors, "Rosenbrock function," Online; available at: `https://en.wikipedia.org/wiki/Rosenbrock_function`, 2022.

[3] Howard H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[4] Haw-ren Fang, Dianne P. O'Leary , "Modified Cholesky algorithms: a catalog with new approaches,"

[5] Thomas V. Mikosch, Sidney I. Resnick, Stephen M. Robinson (Eds.), *Springer Series in Operations Research and Financial Engineering*, Publisher: Springer,

[6] Francesco Della Santa, *Numerical Optimization for Large Scale Problems Lecture Notes*,

[7] Pieraccini, S. (2023/2024). *Numerical optimization for large scale problems and stochastic optimization.* Lecture conducted at PoliTo, Turin.