Numerical Optimization for Large Scale Problems

Report

Assignment on Unconstrained Optimization

Di Battista Simona — 302689
Rostagno Andrea — 349152

June 14, 2025

Contents

1 Introduction **1**
- 1.1 Modified Newton Method . . . 3
  - 1.1.1 Finite differences approximations . . . 4
- 1.2 Nelder–Mead Method . . . 6

2 Rosenbrock Function in Dimension 2 **8**
- 2.1 Problem introduction . . . 8
- 2.2 Experimental results . . . 9

3 Extended Rosenbrock Function **11**
- 3.1 Problem introduction . . . 11
- 3.2 Modified Newton method . . . 12
  - 3.2.1 Modified Newton method with exact derivatives . . . 13
  - 3.2.2 Modified Newton method with approximated derivatives 16
- 3.3 Nelder–Mead method . . . 25

4 Generalized Broyden Tridiagonal Function **29**
- 4.1 Problem introduction . . . 29
- 4.2 Modified Newton method . . . 30
  - 4.2.1 Modified Newton method with exact derivatives . . . 31
  - 4.2.2 Modified Newton method with approximated derivatives 34
- 4.3 Nelder–Mead method . . . 42

5 Banded Trigonometric Function **46**
- 5.1 Problem introduction . . . 46
- 5.2 Modified Newton method . . . 47
  - 5.2.1 Modified Newton method with exact derivatives . . . 48
  - 5.2.2 Modified Newton method with approximated derivatives 52
- 5.3 Nelder–Mead method . . . 59

6 Conclusions **64**

1 Introduction

The goal of this project is to implement and compare two numerical methods for unconstrained optimization: the Modified Newton method and the Nelder-Mead method.

First these algorithms are tested on the standard 2-dimensional Rosenbrock function using two different initial conditions, in order to validate their implementation. Subsequently, they are applied to three benchmark problems selected from the test set for unconstrained optimization proposed in [**?**].

For benchmark problems, in accordance with the assignment instructions, the Nelder-Mead method is tested in low dimensions $n = 10, 26, 50$, while the Modified Newton method is evaluated on $n = 10^3, 10^4, 10^5$. Furthermore, for each test function and each method, a fixed starting point $\bar{x}$ suggested in [**?**] is used, together with 10 uniformly randomly generated starting points, sampled in the hypercube $[\bar{x}_1 - 1, \bar{x}_1 + 1] \times \cdots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$. Performance is assessed in terms of number of successful runs, total iterations, CPU time. Also an experimental rate of convergence is computed, starting from the definition of rate of convergence. First it was defined $e^{(k)} = x^{(k)} - x^\star$, where $x^\star$ is the optimal minimizer; since the value of the optimal minimizer is not known a priori, it is replaced in the formula by the last available approximation, obtaining $e^{(k)} \approx x^{(k)} - x^{(k-1)}$. For k large enough

$$\frac{\left\|e^{(k+1)}\right\|}{\|e^{(k)}\|} \approx \left\|\frac{e^{(k)}}{e^{(k-1)}}\right\|^\rho \Rightarrow \rho \approx \frac{\log\left(\left\|e^{(k+1)}\right\| \Big/ \left\|e^{(k)}\right\|\right)}{\log\left(\|e^{(k)}\| \Big/ \|e^{(k-1)}\|\right)}.$$

Ultimately, the experimental convergence rate is determined by implementing the following formula:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \Big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \Big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

Each starting point is associated with a run, considered successful when the algorithm satisfies a prescribed stopping criterion within a certain maximum number of iterations, depending on the method. These details will be described better in the next sections. Moreover, in the case of Modified Newton method, in addition to exact gradients and hessians, finite differences approximations are employed with different step sizes and types of increment.

The next sections describe the implemented methods and the test problems in detail, followed by a discussion of the experimental results, cost analysis, and final remarks.

1.1 Modified Newton Method

The Modified Newton method is a variant of the classical Newton method. At each iteration it computes a search direction $p^{(k)}$ by solving the linear system

$$H_{mod}^{(k)} p^{(k)} = -\nabla f(x^{(k)}),$$

involving a modified version of the hessian matrix, computed as follow:

$$H_{mod}^{(k)} = H^{(k)} + E^{(k)}.$$

This modification is to ensure the positive definiteness of the hessian, so it is not necessarily computed. The update rule is

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)},$$

where $\alpha^{(k)}$ is determined in the code by a backtracking line search, governed by the Armijo condition

$$f(x^{(k)} + \alpha^{(k)} p^{(k)}) \leq f(x^{(k)}) + c\alpha^{(k)} \nabla f(x^{(k)})^\top p^{(k)}.$$

The algorithm attempts a full step $\alpha^{(k)} = 1$ first, and then reduces it geometrically until the condition is met or a maximum number of trials is reached.

In our implemetation:

1. $E^{(k)} = \tau I$. A modified Cholesky factorization is applied (Algorithm 6.3), that attempts to factor $H^{(k)} \approx LL^\top$ and adds the regularization shift τI, if needed. The procedure starts with $\tau = 0$, and doubles it iteratively until the factorization succeeds, ensuring positive definiteness of the hessian at each step.

2. $\rho = 0.5$ and $c = 10^{-4}$ are default parameters in the Armijo condition.

3. max_backtracking_iter = 10 is the maximum number of trials in the backtracking procedure. This value was chosen to prevent the stepsize alpha from becoming too small, and thus not adding more information than the current approximation of x.

Additionally, when testing problems with structure (such as banded functions), some components of the gradient and direction vectors are padded to preserve compatibility with the structure of the objective function and to avoid dimension mismatch. Finally, the following stopping criteria were considered:

- $\|\nabla f(x^{(k+1)})\|_\infty \leq \texttt{tol}$: the norm of the gradient computed in the new approximation is sufficiently small (below a required tolerance). If this criterion is met, the algorithm has reached a stationary point for the function to be minimized;
- $|f(x^{(k+1)}) - f(x^{(k)})| \leq \texttt{tol} \cdot \max(1, |f(x^{(k)})|)$: the relative functional decrement is sufficiently small (below a required tolerance). Such a criterion is introduced to prevent the algorithm from proceeding in an unnecessarily costly number of iterations, which do not significantly improve the function (as in the case of the Banded Trigonometric problem);
- $f(x^{(k+1)}) \leq \texttt{tol}$: the value of the function computed in the new approximation is sufficiently small (below a required tolerance). This stopping criterion is introduced in the case of functions whose minimum value is 0 (as in the case of Extended Rosenbrock and Generalized Broyden problems).

The algorithm breaks before reaching the maximum number of iterates, as soon as one of the previous conditions –checked in that order– is met. In the study of all benchmark problems, when Modified Newton method is applied, each individual execution, associated with a specific initial point, was considered a success if the algorithm breaks since the gradient is close to zero, that is, if the algorithm reaches a stationary point of the function.

1.1.1 Finite differences approximations

Within the elaborate, where possible, the finite differences method was implemented for the computation of first–and second–order derivatives. In the code, this variant is found in the Modified Newton method, which requires calculation of both the gradient and the hessian of the function to be minimized. Recalling that Nelder-Mead is a "zero-order" method, this part does not involve the use of finite differences.

The gradient approximation is obtained using the centered finite differences formula. In fact, although this is more expensive than forward and backward formulas, it offers a more accurate approximation of the required value. Considering $f : \mathbb{R}^n \to \mathbb{R}$ and $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$, with

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right),$$

using centered finite differences for the approximated gradient, we obtain

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + he_i) - f(x - he_i)}{2h}, \quad i = 1, \ldots, n. \tag{1}$$

Instead, in the hessian matrix $\nabla^2 f \in \mathbb{R}^{n,n}$

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \ldots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix},$$

each entry is obtained using the following calculus:

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \approx \frac{f(x + he_i + he_j) - f(x + he_i) - f(x + he_j) + f(x)}{h^2}. \tag{2}$$

The code implement the finite differences approximations, passing via handles the extra parameters `h` and `type`. In particular

- `h`: is a parameter equal to 10^{-k}, for $k = 2, 4, 6, 8, 10, 12$;
- `type`: is a parameter which indicates the typology of the increment. If type = 1 a variant of the increment scaled componentwise as $h_i = 10^{-k} \cdot |x_i|$ is used (this ensures reasonable accuracy and robustness when computing derivatives in high dimensions), otherwise default increment with $h_i = 10^{-k}$ is employed.

Since application of the Modified Newton method to large benchmak problems ($n = 10^3, 10^4, 10^5$) is required in the assignment, it was not possible to implement the formulas as presented, but it was necessary to make extensive use of each test function's structure and sparsity of each hessian matrix. Alternatively, the time to run the algorithms would have been excessive. In each of the sections regarding individual benchmark problems, it is detailed how these features were exploited in order to obtain reasonable results.

1.2 Nelder–Mead Method

The Nelder–Mead algorithm is a popular derivative–free optimization method designed to minimize a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ without relying on gradient or hessian informations. The algorithm operates by iteratively updating a simplex—a geometric figure composed of $n+1$ vertices in $\mathbb{R}^n$—based on function evaluations at its vertices.

At initialization, the algorithm constructs the simplex by perturbing the initial guess $x^{(0)}$ along the canonical directions with a small fixed step size. At each iteration k, the vertices are sorted according to their function values, in such a way that

$$f(x_1^{(k)}) \leq f(x_2^{(k)}) \leq \cdots \leq f(x_{n+1}^{(k)}).$$

Then the algorithm computes the centroid $\bar{x}$ of the best n vertices (excluding the worst one $x_{n+1}^{(k)}$), and applies the following operations:

- **Reflection:** the worst vertex is reflected through the centroid to produce a new trial point $x_R^{(k)}$. If $f(x_1^{(k)}) \leq f(x_R^{(k)}) < f(x_n^{(k)})$, then $x_R^{(k)}$ is accepted in the new simplex in place of $x_{n+1}^{(k)}$.

- **Expansion:** if $f(x_R^{(k)}) < f(x_1^{(k)})$, the reflected point significantly improves the function, so an expansion is attempted further along the reflection direction producing the point $x_E^{(k)}$. If $f(x_E^{(k)}) < f(x_R^{(k)})$, then $\tilde{x}_{n+1}^{(k+1)} = x_E^{(k)}$, otherwise $\tilde{x}_{n+1}^{(k+1)} = x_R^{(k)}$.

- **Contraction:** if the reflection fails to improve, the algorithm attempts a contraction between the centroid and the worst vertex, producing the point $x_C^{(k)}$.

- **Shrinkage:** if the contraction step is successful, then $\tilde{x}_{n+1}^{(k+1)} = x_C^{(k)}$, otherwise the entire simplex is shriked around the best vertex.

In our implementation default coefficients which are standard in the literature are used: $\rho = 1$ (reflection parameter), $\chi = 2$ (expansion parameter), $\gamma = 0.5$ (contraction parameter), and $\sigma = 0.5$ (shrinkage parameter).

Convergence is declared when one of the following stopping conditions is met:

- $|f(x_{\max}) - f(x_{\min})| \leq \texttt{tol}$: the maximum difference in function values across the simplex is sufficiently small (below a required tolerance);

- $\max_{i=2,\ldots,n+1} \|x^{(i)} - x^{(1)}\|_\infty \leq$ `tol`: the maximum distance between simplex points is sufficiently small (below a required tolerance).

2 Rosenbrock Function in Dimension 2

2.1 Problem introduction

The Rosenbrock function is a well-known test case for unconstrained optimization. Its 2-dimensional version is defined as

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

and features a curved valley with a global minimum $f(x^\star) = 0$ at $x^\star = (1, 1)$. Due to the narrow and curved shape of the valley, the problem is nontrivial for many first-order methods.

We tested both the Modified Newton and Nelder–Mead algorithms using the two standard initial guesses required by the assignment:

$$x_A^{(0)} = (1.2,\ 1.2), \qquad x_B^{(0)} = (-1.2,\ 1.0).$$

3D Visualization of the function:

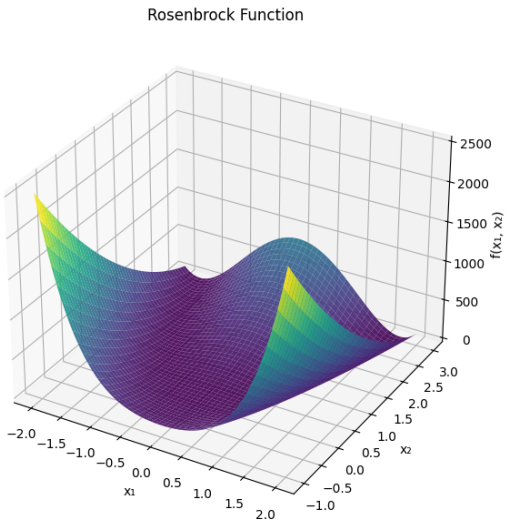


Figure 1: Surface plot of the Rosenbrock function over the domain $[-2, 2] \times [-1, 3]$. The function exhibits a narrow curved valley with a global minimum at $(1, 1)$, where $f(x) = 0$. This geometry makes it a standard benchmark for testing unconstrained optimization algorithms.

2.2 Experimental results

This subsection discusses the results obtained for the minimization of the 2-dimensional Rosenbrock function, comparing the application of the Modified Newton method and the Nelder–Mead method.

- **Modified Newton Method**:

Starting point	Function minimum	Function minimizer	Number of iterations
$x_A^{(0)}$	0.000000	$(1.000050, 1.000083)$	6
$x_B^{(0)}$	0.000000	$(0.999995, 0.999990)$	21

Table 1: The table highlights the convergence results associated with the application of the Modified Newton method. The analysis takes into account the starting point, the value of the minimum and the value of the minimizer reached, and the number of iterations that were required to achieve these results.

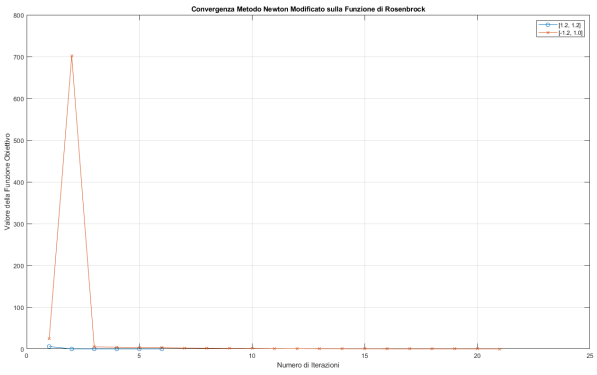


Figure 2: Convergence behaviour of the Modified Newton method on the Rosenbrock function starting from $x_A^{(0)} = [1.2, 1.2]$ and $x_B^{(0)} = [-1.2, 1.0]$.

- **Nelder–Mead Method**:

Starting point	Function minimum	Function minimizer	Number of iterations
$x_A^{(0)}$	0.000000	$((0.999741, 0.999441)$	58
$x_B^{(0)}$	0.053018	$((1.222612, 1.488895)$	29

Table 2: The table highlights the convergence results associated with the application of the Nelder–Mead method. The analysis takes into account the starting point, the value of the minimum and the value of the minimizer reached, and the number of iterations that were required to achieve these results.

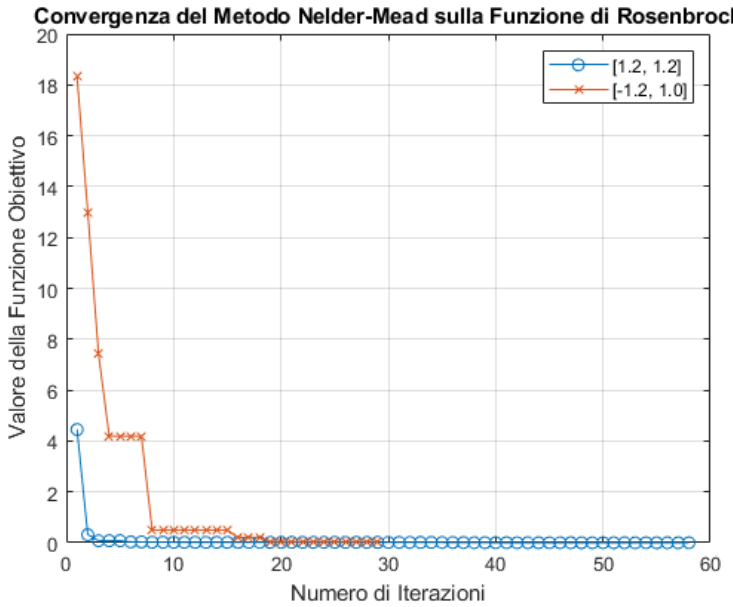


Figure 3: Convergence behaviour of the Nelder–Mead method on the Rosenbrock function starting from $x_A^{(0)} = [1.2, 1.2]$ and $x_B^{(0)} = [-1.2, 1.0]$.

Although both methods converge starting from $x_A^{(0)}$, the Modified Newton method reaches the solution significantly faster (6 iterations vs. 58). Starting from the more challenging initial point $x_B^{(0)}$, the Modified Newton method converges reliably, while Nelder–Mead gets stuck in a suboptimal region, with higher final function value and fewer iterations. This fact confirms the advantage of second–order informations for curved valleys.

3 Extended Rosenbrock Function

3.1 Problem introduction

The Extended Rosenbrock function is an high-dimensional generalization of the classical Rosenbrock function. For even dimensions n, it is defined as follow:

$$F(x) = \frac{1}{2}\sum_{k=1}^{n} f_k^2(x), \qquad f_k(x) = \begin{cases} 10(x_k^2 - x_{k+1}), & \text{if } k \bmod 2 = 1 \\ x_{k-1} - 1, & \text{if } k \bmod 2 = 0 \end{cases}.$$

This function is non–convex and features a narrow curved valley that makes optimization challenging in high dimensions. Its global minimum is $f(x^\star) = 0$ and its minimizer $x^\star$ such that $x_k^\star = 1$, $\forall i = 1, \ldots, n$. It is frequently used as a benchmark problem for large–scale unconstrained optimization algorithms, due to its scalability and pathological curvature.

In this chapter we are going to analyze the behaviour of both the Modified Newton and the Nelder–Mead methods, when they are applied to minimize the Extended Rosenbrock function. The initial points suggested in the benchmark library are $\bar{x} \in \mathbb{R}^n$ such that

$$\bar{x}_k = \begin{cases} -1.2 & \text{if} \quad k \bmod 2 = 1 \\ 1.0 & \text{if} \quad k \bmod 2 = 0 \end{cases}, \tag{3}$$

together with another 10 random initial points sampled uniformly in the hypercube $[\bar{x}_1 - 1, \bar{x}_1 + 1] \times \cdots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$, starting from $\bar{x}$.

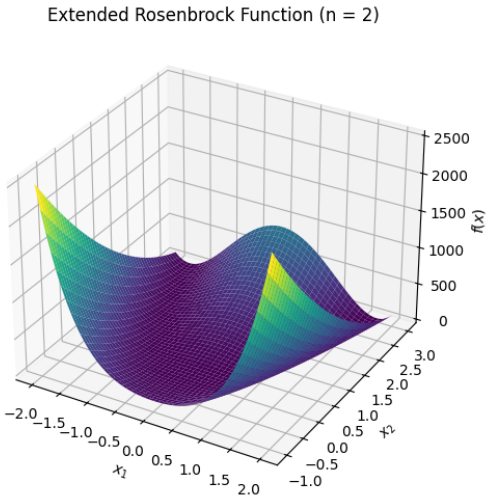


Figure 4: 3D visualization of the Extended Rosenbrock function in dimension $n = 2$. The global minimum lies at $(1, 1)$, and the function exhibits a curved valley that becomes increasingly difficult to navigate in higher dimensions.

3.2 Modified Newton method

In this subsection are shown the results obtained from the minimization of the Extended Rosenbrock function using the Modified Newton method. The study includes both exact derivatives and derivatives computed by finite–differences approximations. Due to its tridiagonal structure, the hessian is stored and manipulated in sparse format; this choice drastically reduces memory requirements and computational cost (in both matrix factorization and Newton direction computation), allowing efficiently handle large–scale problems. Before presenting detailed outcomes, a general experimental setup is given:

- `n` $= 10^3, 10^4, 10^5$;
- `max_iter` $= 5000$;
- `tol` $= 10^{-6}$.

For each run, the following were tracked:

- number of iterations to convergence;
- CPU time;

- number of successful runs;
- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported. Furthermore, the code is designed to work also with finite–differences case, and this variant is implemented thanks to the extra parameters

- `h`: a parameter equal to 10^{-k}, for $k = 2, 4, 6, 8, 10, 12$;
- `type`: a parameter which indicates if the increment is scaled componentwise as $h_i = 10^{-k} \cdot |x_i|$ or if it is a default increment such that $h_i = 10^{-k}$.

For this part, the several features were computed for each stepsize h.

3.2.1 Modified Newton method with exact derivatives

The Extended Rosenbrock function admits analytical expressions for both gradient and hessian in component–wise form. These are derived by exploiting the structure of the function and are used directly in the implementation for performance and accuracy. Specifically:

- each gradient's component is computed as:

$$\frac{\partial F}{\partial x_k}(x) = \begin{cases} 200(x_k^3 - x_k x_{k+1}) + (x_k - 1), & \text{if } k \bmod 2 = 1 \\ -100(x_{k-1}^2 - x_k), & \text{if } k \bmod 2 = 0 \end{cases};$$

- each entry of the hessian is given by:

$$\frac{\partial^2 F}{\partial x_k \partial x_j}(x) = \begin{cases} 200(3x_k^2 - x_{k+1}) + 1, & \text{if } j = k,\ k \bmod 2 = 1 \\ 100, & \text{if } j = k,\ k \bmod 2 = 0 \\ -200x_k, & \text{if } |k - j| = 1,\ k \bmod 2 = 1 \\ 0 & \text{otherwise} \end{cases}.$$

Dimension	Starting point	Iter	Time (s)	ρ	Success
10^3	$\bar{x}$	20	0.00	2.14	1/1
10^3	Avg (10 pts)	25.3	0.01	2.02	10/10
10^4	$\bar{x}$	20	0.02	2.14	1/1
10^4	Avg (10 pts)	25.4	0.05	1.96	10/10
10^5	$\bar{x}$	20	0.34	2.14	1/1
10^5	Avg (10 pts)	26.0	0.74	1.37	10/10

Table 3: The table highlights the convergence results associated with the application of the Modified Newton method with exact derivatives. The analysis takes into account the starting point (for the 10 random points an average behaviour is reported), the number of iterations that were required to achieve the result, the CPU time (s) and the rate of convergence ρ.

Experimental results. All runs shown in table (3) were successful according to the stopping criterion related to the gradient ($\|\nabla f(x^{(k)})\|_\infty \leq \texttt{tol}$), and reached the known minimum $f^\star = 0$ up to machine precision. The number of iterations remains nearly constant across dimensions both for $\bar{x}$ and for the 10 random starting points, highlighting the scalability of Newton's method when combined with sparse matrix storage. CPU time increases with n, as expected due to the cost of Cholesky factorization on sparse tridiagonal matrices. The experimental convergence rate ρ remains close to quadratic ($\rho \approx 2$) for moderate dimensions, while it slightly drops for $n = 10^5$, possibly due to cumulative rounding errors.

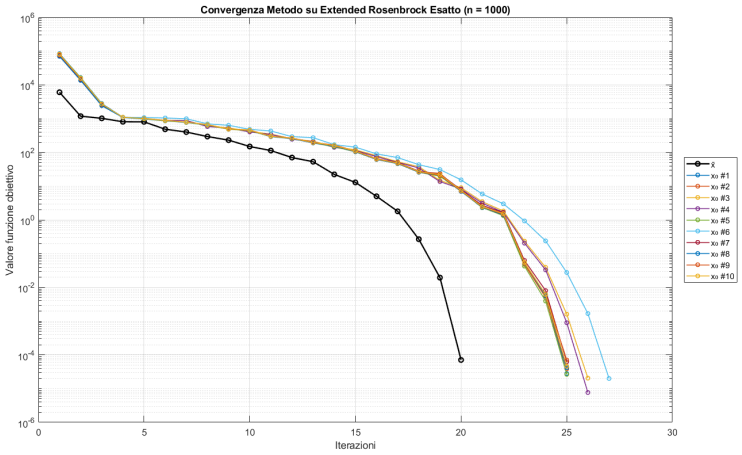


Figure 5: Convergence of the Modified Newton method with exact derivatives on the Extended Rosenbrock function for $n = 1000$. Each curve corresponds to a different initial point. The method converges quadratically with stable behaviour across all tests.

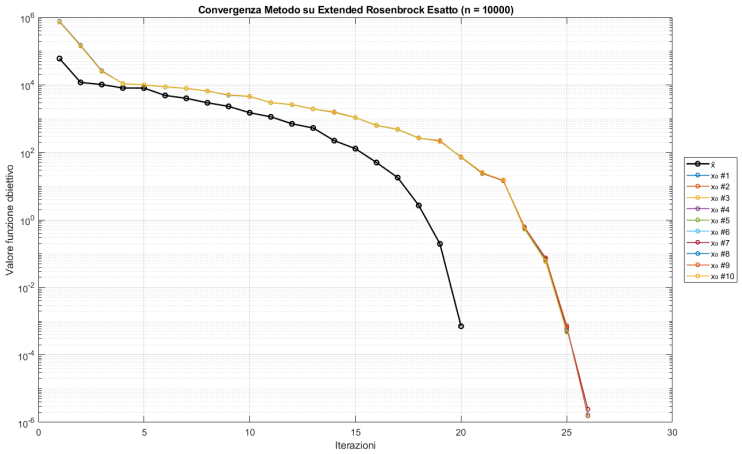


Figure 6: Convergence of the Modified Newton method on the Extended Rosenbrock function for $n = 10\,000$. The method exhibits consistent quadratic convergence also in higher dimension.

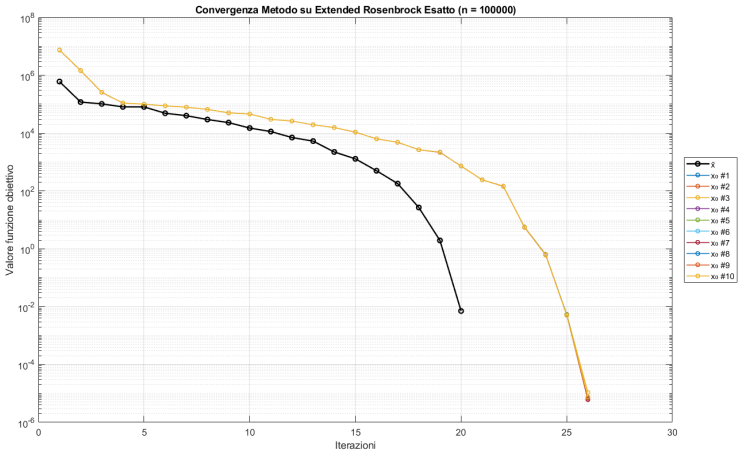


Figure 7: Convergence of the Modified Newton method on the Extended Rosenbrock function for $n = 100\,000$. Despite the high dimensionality, the convergence remains stable with similar iteration counts.

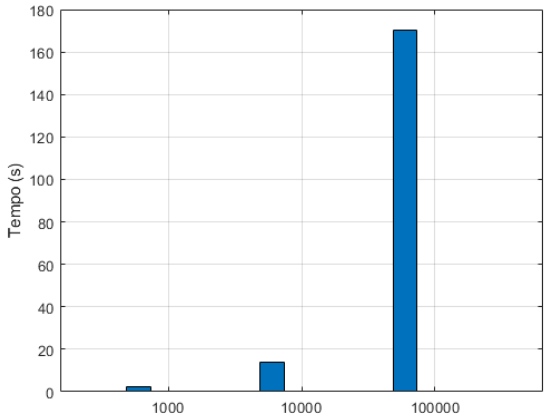


Figure 8: Execution time (in seconds) of the Modified Newton method with exact derivatives for $n = 10^3$, 10^4 and 10^5. The runtime grows approximately linearly with the problem size, confirming the efficiency of sparse matrix operations.

3.2.2 Modified Newton method with approximated derivatives

When exact gradients and hessians are not available, a common alternative is to approximate them using finite differences method. In our implementation,

we manually built the approximations in component-wise form, exploiting the alternating structure of the Extended Rosenbrock function.

Gradient approximation. As we have seen in the introduction part, the gradient vector is computed using the centered finite differences formula (1). Due to the structure of the function, only a small number of f_k survive when computing the numerator in the formula. In particular, fixed k as the component whose derivative is being approximated:

$$\frac{\partial F}{\partial x_k}(x) \approx \begin{cases} \frac{\frac{1}{2}[f_k^2(x+he_k)+f_{k+1}^2(x+he_k)]-\frac{1}{2}[f_k^2(x-he_k)+f_{k+1}^2(x-he_k)]}{2h} & \text{if } k \bmod 2 = 1 \\ \frac{\frac{1}{2}f_{k-1}^2(x+he_k)-\frac{1}{2}f_{k-1}^2(x-he_k)}{2h} & \text{if } k \bmod 2 = 0 \end{cases}. \tag{4}$$

Thus, the finite-difference approximation of the gradient was implemented as:

$$\frac{\partial F}{\partial x_k}(x) \approx \begin{cases} 600x_k^2 - 100x_k x_{k+1} + \frac{1}{2}h + 350x_k^3 + 300hx_k^2, & \text{if } k \bmod 2 = 1 \\ -100x_{k-1}^2 + 100x_k, & \text{if } k \bmod 2 = 0 \end{cases}.$$

Hessian approximation. First, as seen in the previous section, we recall that the hessian matrix of the Extended Rosenbrock function has a tridiagonal structure. Then, appling the same proceeding as before, we can observe that function evaluations needed to compute diagonal entries of $\nabla^2 F(x)$, differ only by the terms

$$\begin{cases} \frac{1}{2}[f_k^2(x) + f_{k+1}^2(x)] & \text{if} \quad k \bmod 2 = 1 \\ \frac{1}{2}f_{k-1}^2(x) & \text{if} \quad k \bmod 2 = 0 \end{cases},$$

while function evaluations needed to compute non-diagonal entries differ only by the terms

$$\begin{cases} \frac{1}{2}[f_k^2(x) + f_{k+1}^2(x)] & \text{if} \quad k \bmod 2 = 1 \\ 0 & \text{if} \quad k \bmod 2 = 0 \end{cases}.$$

So, the approximated component-wise second derivatives can be coded as:

$$\frac{\partial^2 F}{\partial x_k \partial x_j}(x) \approx \begin{cases} 1200hx_k - 200x_{k+1} + 700h^2 + 600x_k^2, & \text{if } j = k,\ k \bmod 2 = 1 \\ 100, & \text{if } j = k,\ k \bmod 2 = 0 \\ -100hx_k - 200x_k, & \text{if } |k-j| = 1,\ k \bmod 2 = 1 \\ 0, & \text{otherwise} \end{cases}$$

Experimental results. In finite differences–case, two figures were generated per dimension: one for fixed and one for scaled increment. As others, each plot contains:

- a black curve for the reference point $\bar{x}$;
- ten colored curves, one for each random starting point;
- log–scaled $f(x_k)$ values against iterations.

In general, the algorithm lead to convergence within reasonable iteration counts and CPU time, despite the considerable increase in size; scaled increments tend to be slightly more robust and stable when compared with default increments, for all dimensions. In case of $n = 1\,000$ all increments succeed consistently with almost full accuracy, but this does not happen as the size of the problem increases. In fact, both for $n = 10\,000$ and $n = 100\,000$, the algorithm fails to converge in all trials, in correspondence with the largest fixed and the largest scaled increments referred to $h = 10^{-2}$. Moreover, for $n = 1\,000$, quadratic convergence is observed for $h \leq 10^{-6}$ (with $\rho \approx 2$) both for $\bar{x}$ and the other 10 random points, while it slightly drops ($\rho \approx 1$) starting from the 10 random points in large dimensions. The following tables contain more detailed outcomes.

- $\boldsymbol{n = 1\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	118	0.03	0.8856	1/1
10^{-2}	Avg (10 pts)	122.5	0.038	0.8856	10/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	110	0.03	0.8794	1/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	113.6	0.03	0.8794	10/10
10^{-4}	$\bar{x}$	21	0.01	1.1640	1/1
10^{-4}	Avg (10 pts)	26	0.01	1.3972	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	21	0.01	1.1476	1/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	26	0.01	1.4166	10/10
10^{-6}	$\bar{x}$	20	0.01	2.4144	1/1
10^{-6}	Avg (10 pts)	25.2	0.01	2.1933	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	20	0.01	2.4464	1/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	25.2	0.01	2.2105	10/10
10^{-8}	$\bar{x}$	20	0.01	2.1463	1/1
10^{-8}	Avg (10 pts)	25.2	0.01	2.2284	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	20	0.01	2.1468	1/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	25.2	0.01	2.2284	10/10
10^{-10}	$\bar{x}$	20	0.01	2.1432	1/1
10^{-10}	Avg (10 pts)	25.2	0.01	2.2278	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	20	0.01	2.1432	1/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	25.2	0.01	2.2278	10/10
10^{-12}	$\bar{x}$	20	0.01	2.1432	1/1
10^{-12}	Avg (10 pts)	25.2	0.01	2.2278	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	20	0.01	2.1432	1/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	25.2	0.01	2.2278	10/10

Table 4: Finite difference results using different typology of increments ($n = 10^3$).

Figure 9: Convergence of Modified Newton method on Extended Rosenbrock function in dimension $n = 10^3$ with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

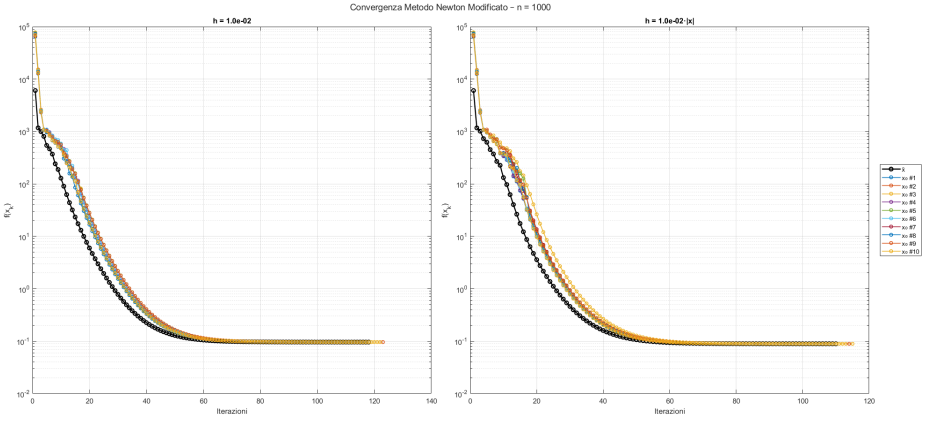
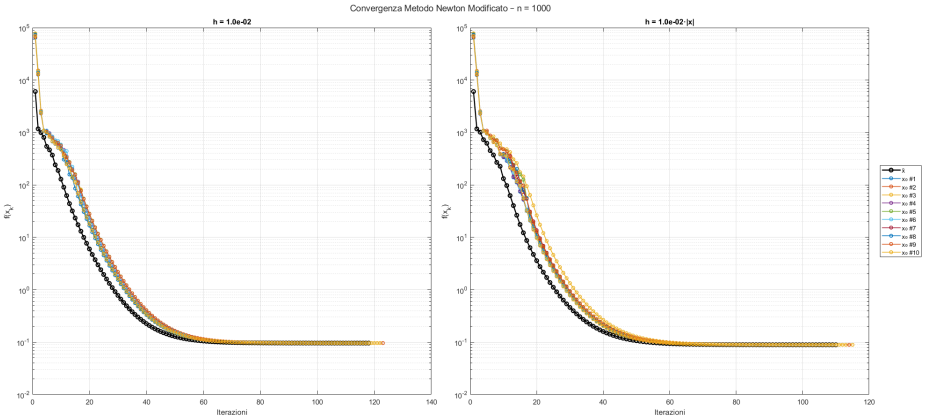


Figure 10: Convergence of Modified Newton method on Extended Rosenbrock function in dimension $n = 10^3$ with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).



- $\boldsymbol{n = 10\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	118	0.23	0.8856	0/10
10^{-2}	Avg (10 pts)	125.3	2.66	0.8856	0/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	110	0.21	0.8794	0/10
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	114.3	2.44	0.8794	0/10
10^{-4}	$\bar{x}$	21	0.04	1.1640	10/10
10^{-4}	Avg (10 pts)	27.0	0.92	1.2708	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	21	0.04	1.1476	10/10
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	27.0	0.91	1.2298	10/10
10^{-6}	$\bar{x}$	20	0.04	2.4144	10/10
10^{-6}	Avg (10 pts)	25.6	0.89	1.3572	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	20	0.04	2.4464	10/10
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	25.6	0.89	1.3581	10/10
10^{-8}	$\bar{x}$	20	0.04	2.1463	10/10
10^{-8}	Avg (10 pts)	25.6	0.90	1.3626	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	20	0.04	2.1468	10/10
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	25.6	0.90	1.3626	10/10
10^{-10}	$\bar{x}$	20	0.04	2.1432	10/10
10^{-10}	Avg (10 pts)	25.6	0.90	1.3626	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	20	0.04	2.1432	10/10
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	25.6	0.90	1.3626	10/10
10^{-12}	$\bar{x}$	20	0.04	2.1432	10/10
10^{-12}	Avg (10 pts)	25.6	0.90	1.3626	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	20	0.04	2.1432	10/10
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	25.6	0.90	1.3626	10/10

Table 5: Finite difference results for using different typology of increments ($n = 10^4$).

Figure 11: Convergence of Modified Newton method on Extended Rosenbrock function in dimension $n = 10^4$ with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

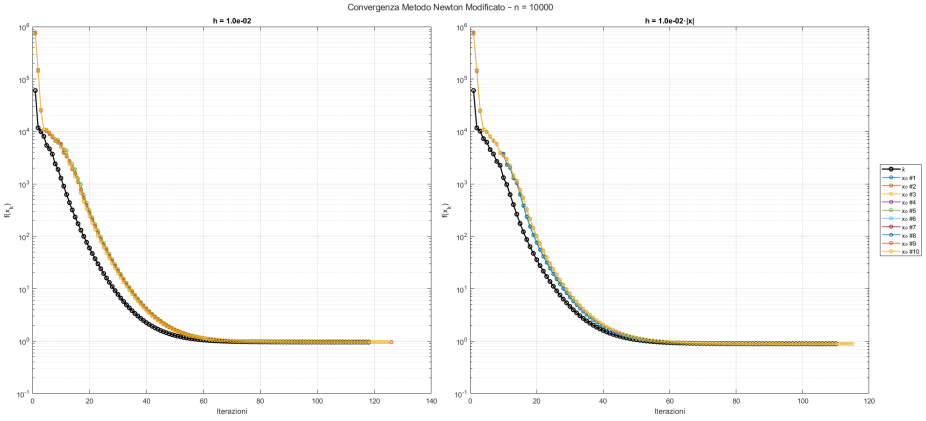
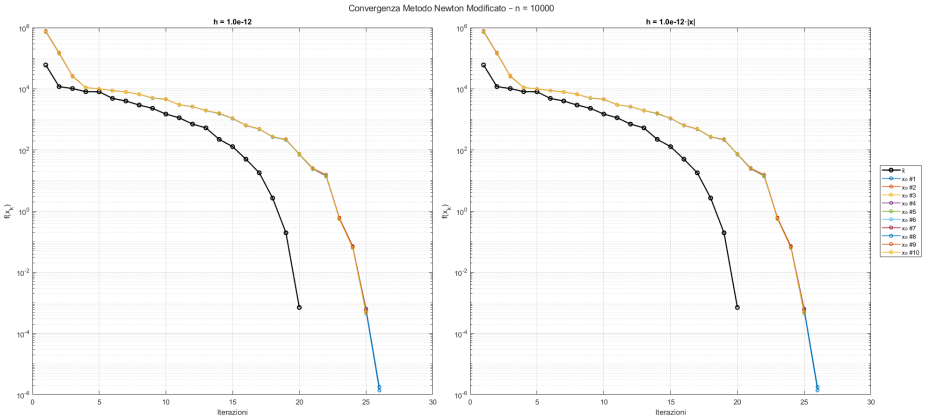


Figure 12: Convergence of Modified Newton method on Extended Rosenbrock function in dimension $n = 10^4$ with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).



- $\boldsymbol{n = 100\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	118	2.66	0.8856	0/10
10^{-2}	Avg (10 pts)	126.0	3.26	0.8856	0/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	110	2.41	0.8794	0/10
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	114.2	2.94	0.8794	0/10
10^{-4}	$\bar{x}$	22	0.49	0.9324	10/10
10^{-4}	Avg (10 pts)	27.0	0.92	1.2738	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	22	0.49	0.9453	10/10
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	27.0	0.91	1.2309	10/10
10^{-6}	$\bar{x}$	20	0.44	2.4144	10/10
10^{-6}	Avg (10 pts)	26.0	0.89	1.3622	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	20	0.44	2.4464	10/10
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	26.0	0.89	1.3620	10/10
10^{-8}	$\bar{x}$	20	0.44	2.1463	10/10
10^{-8}	Avg (10 pts)	26.0	0.90	1.3633	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	20	0.44	2.1468	10/10
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	26.0	0.90	1.3633	10/10
10^{-10}	$\bar{x}$	20	0.45	2.1432	10/10
10^{-10}	Avg (10 pts)	26.0	0.90	1.3632	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	20	0.44	2.1432	10/10
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	26.0	0.90	1.3632	10/10
10^{-12}	$\bar{x}$	20	0.44	2.1432	10/10
10^{-12}	Avg (10 pts)	26.0	0.89	1.3632	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	20	0.44	2.1432	10/10
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	26.0	0.89	1.3632	10/10

Table 6: Finite difference results using different increments ($n = 10^5$).

Figure 13: Convergence of Modified Newton method on Extended Rosenbrock function ($n = 10^5$) with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

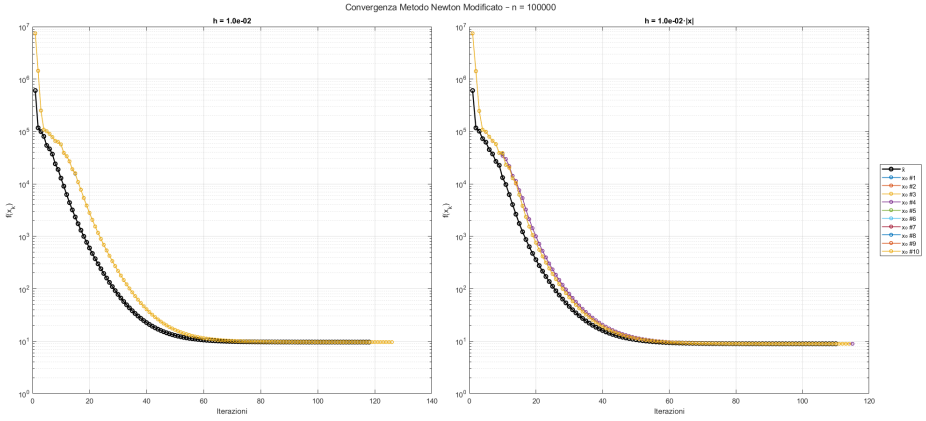
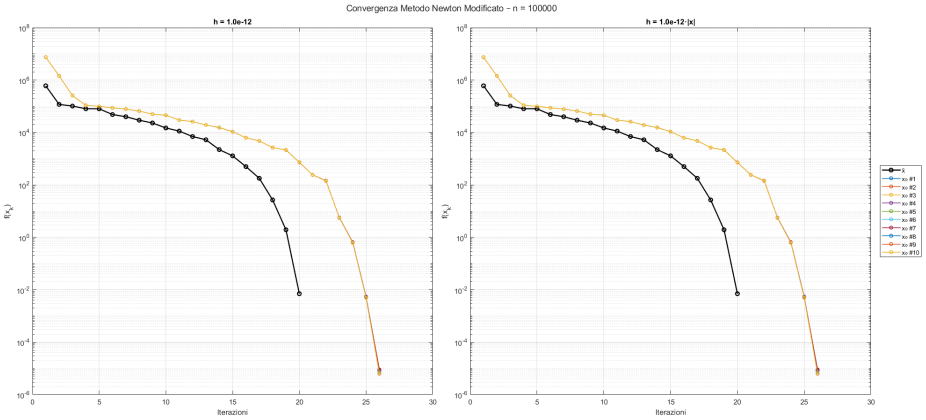


Figure 14: Convergence of Modified Newton method on Extended Rosenbrock function ($n = 10^5$) with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).



3.3 Nelder–Mead method

In this subsection are shown the results obtained from the minimization of the Extended Rosenbrock function using the Nelder–Mead method. Before presenting the results, a general experimental setup is given:

- `n` $= 10, 26, 50$;
- `max_iter` $= 80.000$;
- `tol` $= 10^{-6}$.

Moreover, for each run were recorded:

- number of iterations to convergence;
- CPU time;
- final objective value found f_{min};
- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported.

Experimental results. The algorithm successfully runs on all dimensions, but convergence was never achieved within tolerance. The number of iterations and function values clearly indicate that Nelder–Mead is not effective on this problem; incidentally, in all cases the final function values remained far from zero, indicating that Nelder–Mead struggled with this function even in low dimensions. The high curvature and narrow valleys of the Extended Rosenbrock function severely impact the simplex–based exploration. Furthermore, while runtime remains reasonable, the algorithm not only requires thousands of iterations to make progress but also does not reach values close to zero. This confirms the known limitations of Nelder–Mead in high–dimensional landscapes.

Dimension	Starting point	$f_{\min}$	Iter	Time (s)	ρ
10	$\bar{x}$	4.755805	1807	0.04	0.9082
	Avg (10 pts)	8.248833	1221	0.016	0.0500
26	$\bar{x}$	21.632097	13592	0.20	-0.8190
	Avg (10 pts)	31.887709	9669	0.142	-6.6754
50	$\bar{x}$	36.483857	49703	1.15	-97.5929
	Avg (10 pts)	121.868407	42439	0.953	-1.3963

Table 7: Results of Nelder–Mead on Extended Rosenbrock function.

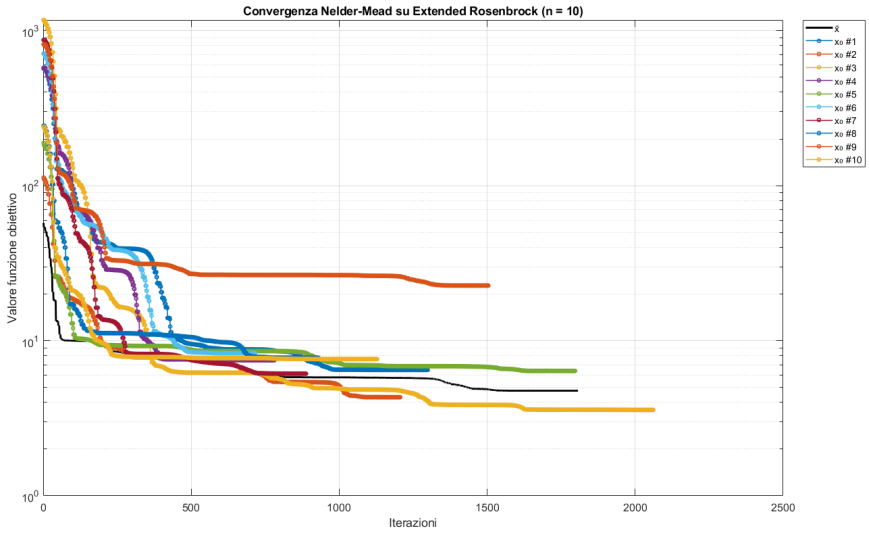


Figure 15: Convergence of Nelder–Mead on Extended Rosenbrock function ($n = 10$) from reference and 10 random initial points. The objective decreases but stagnates above the global minimum.

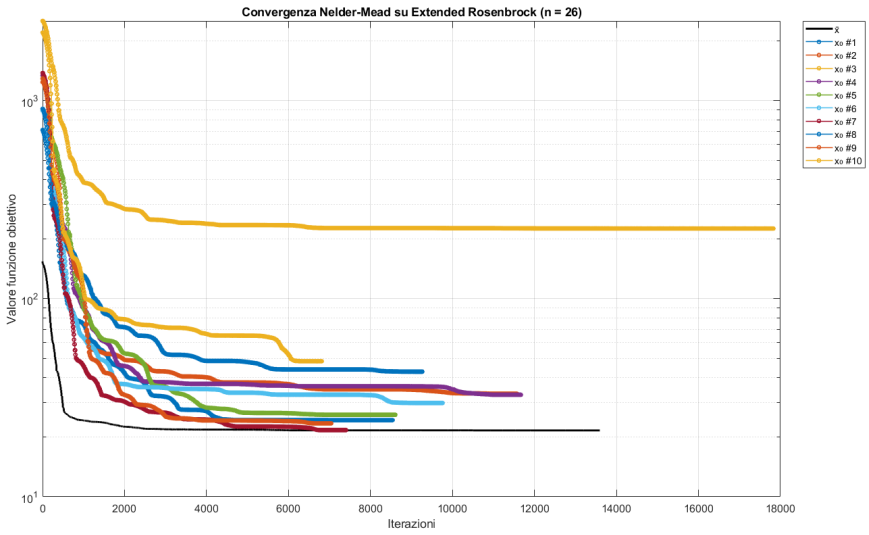


Figure 16: Convergence of Nelder–Mead on Extended Rosenbrock function ($n = 26$). Progress slows down and most trajectories fail to improve after early iterations.

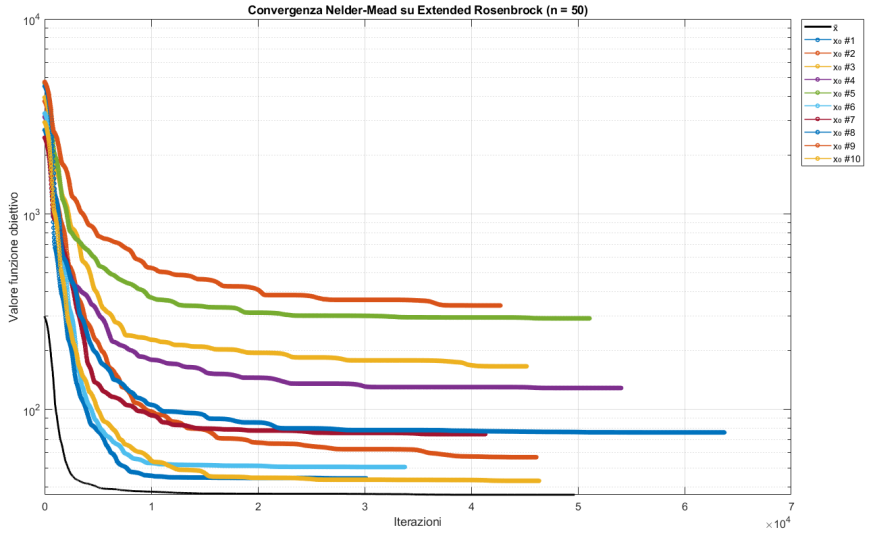


Figure 17: Convergence of Nelder–Mead on Extended Rosenbrock function ($n = 50$). None of the trials reach satisfactory objective values, confirming the method's limits in high dimensions.

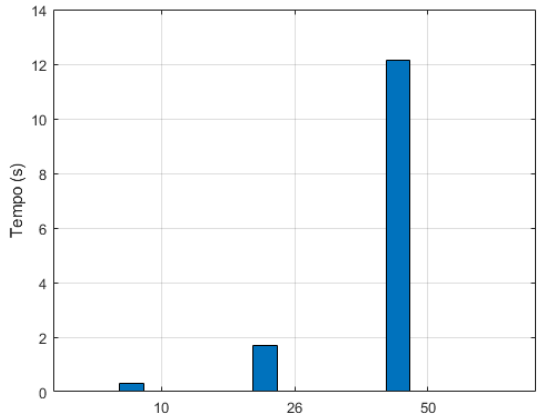


Figure 18: Execution time of Nelder–Mead on Extended Rosenbrock for increasing dimensions $n = 10$, 26, and 50. The cost grows sharply due to the increased simplex size.

4 Generalized Broyden Tridiagonal Function

4.1 Problem introduction

The Generalized Broyden Tridiagonal function is a classical benchmark used in unconstrained optimization, known for its nonlinear and tridiagonal structure. Let $x \in \mathbb{R}^n$,

$$F(x) = \frac{1}{2}\sum_{k=1}^{n} f_k^2(x) \qquad f_k(x) = (3 - 2x_k)x_k + 1 - x_{k-1} - x_{k+1}, \quad (5)$$

with boundary conditions $x_0 = x_{n+1} = 0$. $F(x)$ combines a quadratic component in x_i, two linear coupling terms with the neighbors x_{i-1} and x_{i+1}, and adds a constant shift. The global minimum of the function is $F(x^\star) = 0$ is achieved when all terms inside the squared expression are zero. Thus, the global minimizer $x^* \in \mathbb{R}^n$ is such that

$$x_i^* = 1, \quad \forall i = 1, \ldots, n.$$

In this chapter we are going to analyze the behaviour of both the Modified Newton and the Nelder–Mead methods, when they are applied to minimize the Generalized Broyden Tridigonal function. The initial points suggested in the benchmark library are $\bar{x} \in \mathbb{R}^n$ such that

$$\bar{x}_k = -1.0 \quad \text{for} \quad k = 1, 2, \ldots n,$$

together with another 10 random initial points sampled uniformly in the hypercube $[\bar{x}_1 - 1, \bar{x}_1 + 1] \times \cdots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$, starting from $\bar{x}$.

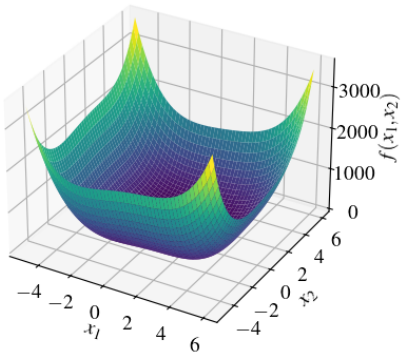


Figure 19: 3D visualization of the Generalized Broyden Tridiagonal function for $n = 2$.

4.2 Modified Newton method

In this subsection are shown the results obtained from the minimization of the Generalized Broyden Tridiagonal function using the Modified Newton method; the study includes both exact derivatives and derivatives computed by finite–differences approximations. Due to its pentadiagonal structure, the hessian is stored and manipulated in sparse format; this choice drastically reduces memory requirements and computational cost (in both matrix factorization and Newton direction computation), allowing efficiently handle large–scale problems. Before presenting detailed outcomes, a general experimental setup is given:

- `n` $= 10^3, 10^4, 10^5$;
- `max_iter` $= 5000$;
- `tol` $= 10^{-6}$.

For each run, the following were tracked:

- number of iterations to convergence;
- CPU time;
- number of successful runs;
- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \Big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \Big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported. Furthermore, the code is designed to work also with finite–differences case, and this variant is implemented thanks to the extra parameters

- `h`: a parameter equal to 10^{-k}, for $k = 2, 4, 6, 8, 10, 12$;
- `type`: a parameter which indicates if the increment is scaled componentwise as $h_i = 10^{-k} \cdot |x_i|$ or if it is a default increment such that $h_i = 10^{-k}$.

For this part, the several features were computed for each stepsize h.

4.2.1 Modified Newton method with exact derivatives

In this section we test the Modified Newton method on the Generalized Broyden Tridiagonal function using exact gradient and hessian. Both were derived analytically and implemented exploiting the banded pattern of the function. Specifically:

- each gradient's component is computed as:

$$\frac{\partial F}{\partial x_k}(x) = \begin{cases} (3-4x_1)f_1(x) - f_2(x), & k = 1 \\ (3-4x_k)f_k(x) - f_{k+1}(x) - f_{k-1}(x), & 1 < k < n \\ (3-4x_n)f_n(x) - f_{n-1}(x), & k = n \end{cases};$$

- each entry of the hessian is given by:

$$\frac{\partial^2 F}{\partial x_k \partial x_j}(x) = \begin{cases} (3-4x_1)^2 - 4f_1(x) + 1, & k = j = 1 \\ (3-4x_k)^2 - 4f_k(x) + 2, & 1 < k < n,\ k = j \\ (3-4x_n)^2 - 4f_n(x) + 1, & k = j = n \\ 4x_k + 4x_{k+1} - 6, & |k-j| = 1 \\ 1, & |k-j| = 2 \\ 0 & \text{otherwise} \end{cases}.$$

Dim	**Init.**	$f_{\min}$	**Iter**	**Time (s)**	ρ
10^3	$\bar{x}$	0.000000	5	0.02	1.93
	Avg (10 pts)	$< 10^{-6}$	8.1	0.02	1.63
10^4	$\bar{x}$	0.000000	5	0.17	1.93
	Avg (10 pts)	$< 10^{-6}$	8.0	0.17	1.64
10^5	$\bar{x}$	0.000000	5	2.01	1.93
	Avg (10 pts)	$< 10^{-6}$	8.0	2.08	1.62

Table 8: Results of Modified Newton method on Generalized Broyden function using exact derivatives.

Experimental results. As shown in table (4.2.1), the Modified Newton method with exact derivatives proves to be extremely effective on the Generalized Broyden tridiagonal function. In all tested dimensions, convergence is achieved in very few iterations (5 for the reference starting point, around 8

on average from random initializations). Function values reach machine precision accuracy, and the convergence rate ρ consistently approaches the ideal quadratic rate of 2. Importantly, the computational time remains reasonable even for large dimensions, thanks to the efficient sparse implementation of the hessian matrix. These results confirm that when second-order information is available and well–structured, Newton–type methods offer both precision and speed — outperforming derivative-free approaches by a large margin in terms of both accuracy and reliability.

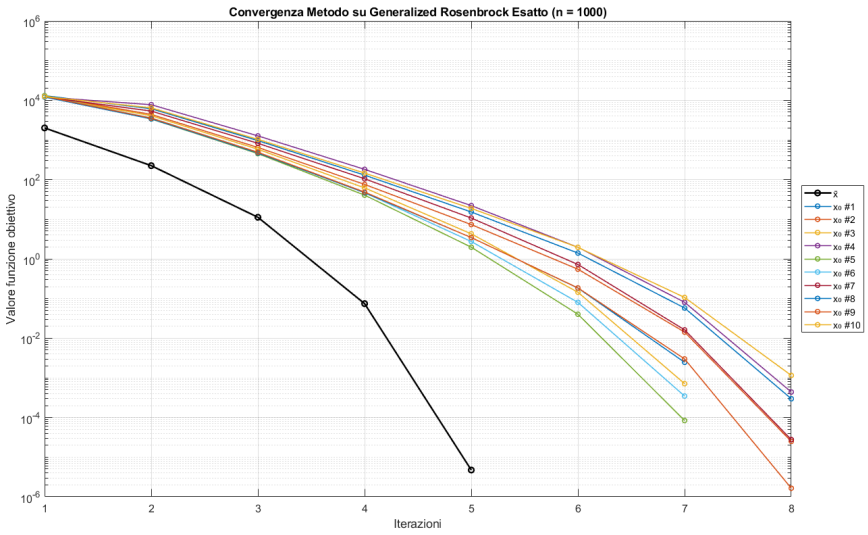


Figure 20: Convergence of the Modified Newton method on Generalized Broyden function with $n = 1000$ using exact gradient and Hessian. The method converges in few iterations with a consistent superlinear rate.

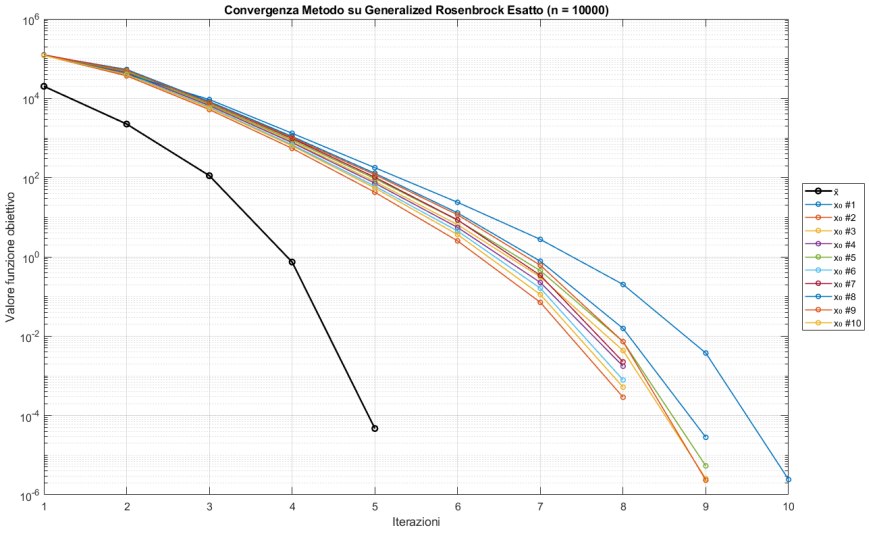


Figure 21: Convergence of the Modified Newton method on Generalized Broyden function with $n = 10000$. The convergence behavior remains stable across all random initializations.

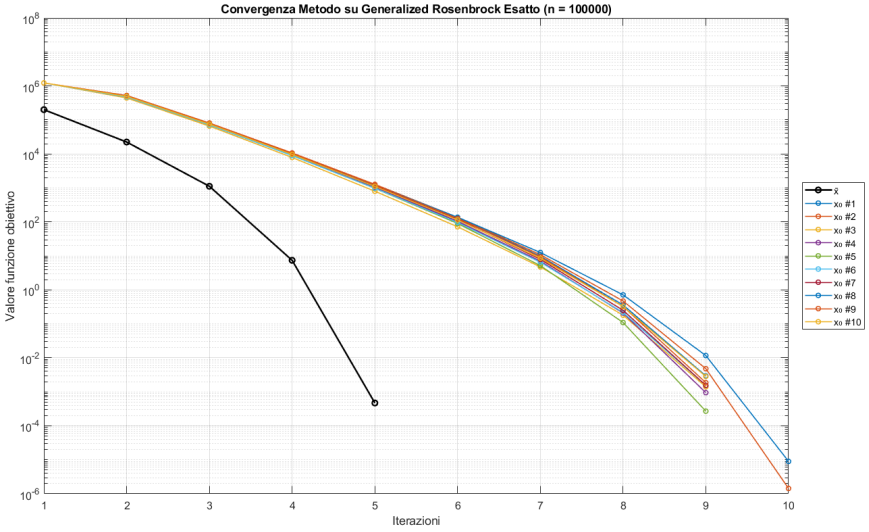


Figure 22: Convergence of the Modified Newton method on Generalized Broyden function with $n = 100000$. Even in high dimensions, the algorithm remains robust and fast.

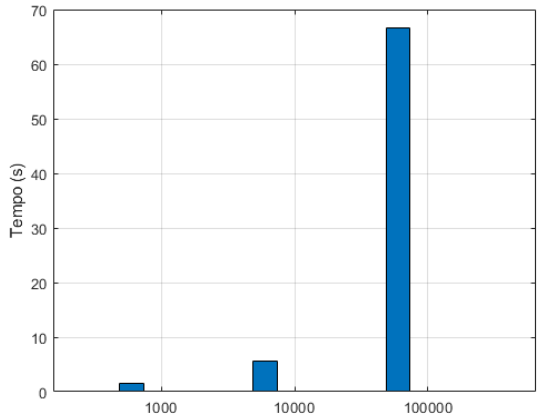


Figure 23: Execution time of the Modified Newton method with exact derivatives on Generalized Broyden function for increasing dimensions. The computation scales efficiently due to sparse Hessian structure.

4.2.2 Modified Newton method with approximated derivatives

In this experiment, we use the Modified Newton method on the Generalized Broyden Tridiagonal function with gradient and hessian computed via finite differences. This approach avoids symbolic derivation, at the cost of numerical approximations and additional function evaluations. The approximated gradient and Hessian components were expanded manually and implemented directly to avoid cancellation and redundancy. The sparsity pattern of the exact hessian was preserved to limit computational cost.

Gradient approximation. Due to the structure of the function, only a small number of f_k survive when computing the numerator in the formula (1). In particular, fixed k as the component whose derivative is being approximated, and defining:

$$g_k(x) = \frac{1}{2}[f_k^2(x) + f_{k+1}^2(x) + f_{k-1}^2(x)],$$

then

$$\frac{\partial F}{\partial x_k}(x) \approx \frac{g_k(x + he_k) - g_k(x - he_k)}{2h}. \tag{6}$$

It is not reported its explicit expression, as it is corpus.

Hessian approximation. Appling the same proceeding as before, we can observe that function evaluations needed to compute diagonal entries of $\nabla^2 F(x)$, differ only by the terms

$$\frac{1}{2}[f_k^2(x) + f_{k+1}^2(x) + f_{k-1}^2(x)].$$

Function evaluations needed to compute non-diagonal entries, on the first codiagonal, differ only by the terms

$$\frac{1}{2}[f_k^2(x) + f_{k+1}^2(x)],$$

while function evaluations needed to compute the ones on the second codiagonal, differ by the terms

$$\frac{1}{2}f_{k-1}^2(x).$$

Specifically, the explicit expressions, derived by expanding these differences, are given in the code.

Experimental results. In finite differences–case, two figures were generated per dimension: one for fixed and one for scaled increment. As others, each plot contains:

- a black curve for the reference point $\bar{x}$;
- ten colored curves, one for each random starting point;
- log–scaled $f(x_k)$ values against iterations.

The finite–differences approach is remarkably stable on this problem. All tested step sizes yield convergence in under 30 iterations. Success rates remain at 100% for all increments, confirming the robustness of the method even in the presence of approximation noise. However, when compared to exact derivatives, the convergence rate ρ is consistently lower, and runtime increases due to the extra evaluations required. For large–scale problems, this gap grows further due to the cost of computing full gradients and hessians numerically. Overall, this shows that while finite differences can provide reliable optimization results, they are computationally expensive and less efficient than analytical derivatives, when available. Below are the results, divided by problem size.

- $\boldsymbol{n = 1\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	11	0.04	0.0216	1/1
10^{-2}	Avg (10 pts)	8.6	0.01	0.1495	10/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	11	0.01	1.0263	1/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	8.2	0.00	0.9274	10/10
10^{-4}	$\bar{x}$	10	0.00	1.6064	1/1
10^{-4}	Avg (10 pts)	7.7	0.00	1.6201	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	10	0.00	1.6122	1/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	7.7	0.00	1.6200	10/10
10^{-6}	$\bar{x}$	10	0.00	1.6112	1/1
10^{-6}	Avg (10 pts)	7.7	0.00	1.6200	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	10	0.00	1.6113	1/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	7.7	0.00	1.6200	10/10
10^{-8}	$\bar{x}$	10	0.00	1.6113	1/1
10^{-8}	Avg (10 pts)	7.7	0.00	1.6200	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	10	0.00	1.6113	1/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	7.7	0.00	1.6200	10/10
10^{-10}	$\bar{x}$	10	0.00	1.6113	1/1
10^{-10}	Avg (10 pts)	7.7	0.00	1.6200	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	10	0.00	1.6113	1/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	7.7	0.00	1.6200	10/10
10^{-12}	$\bar{x}$	10	0.00	1.6113	1/1
10^{-12}	Avg (10 pts)	7.7	0.00	1.6200	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	10	0.00	1.6113	1/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	7.7	0.00	1.6200	10/10

Table 9: Finite differences results for $n = 1000$ using different increments h.

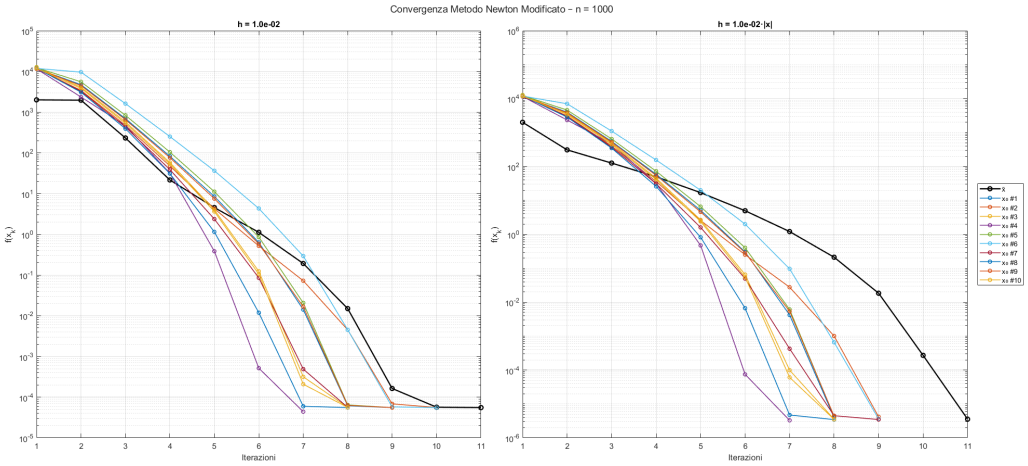


Figure 24: Convergence of Modified Newton method on Generalized Broyden function ($n = 1000$) with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

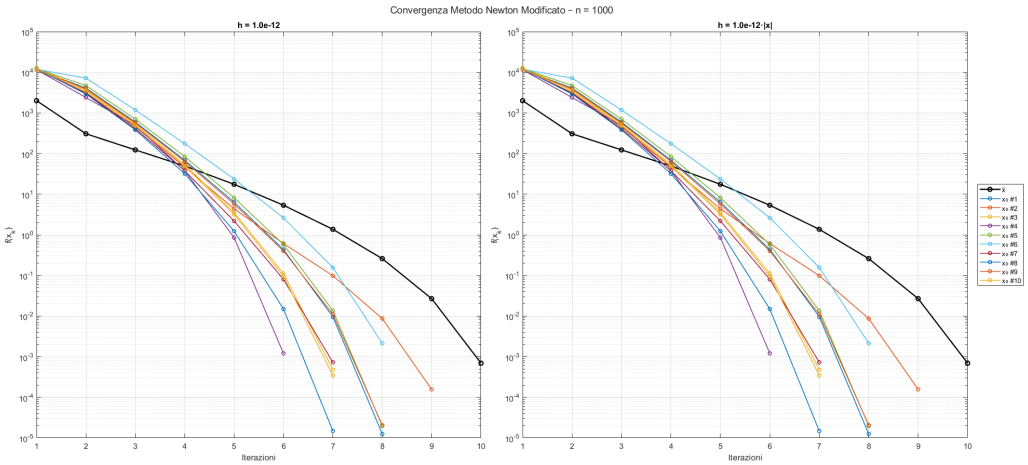


Figure 25: Convergence of Modified Newton method on Generalized Broyden function ($n = 1000$) with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

- $\boldsymbol{n = 10\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	12	0.04	0.0014	1/1
10^{-2}	Avg (10 pts)	9.4	0.04	0.0629	10/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	11	0.03	0.5256	1/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	9.4	0.04	0.4660	10/10
10^{-4}	$\bar{x}$	10	0.03	1.6064	1/1
10^{-4}	Avg (10 pts)	8.5	0.03	1.5916	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	10	0.03	1.6122	1/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	8.5	0.03	1.5916	10/10
10^{-6}	$\bar{x}$	10	0.03	1.6112	1/1
10^{-6}	Avg (10 pts)	8.5	0.03	1.5916	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	10	0.03	1.6113	1/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	8.5	0.03	1.5916	10/10
10^{-8}	$\bar{x}$	10	0.03	1.6113	1/1
10^{-8}	Avg (10 pts)	8.5	0.03	1.5916	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	10	0.03	1.6113	1/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	8.5	0.03	1.5916	10/10
10^{-10}	$\bar{x}$	10	0.03	1.6113	1/1
10^{-10}	Avg (10 pts)	8.5	0.03	1.5916	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	10	0.03	1.6113	1/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	8.5	0.03	1.5916	10/10
10^{-12}	$\bar{x}$	10	0.03	1.6113	1/1
10^{-12}	Avg (10 pts)	8.5	0.03	1.5916	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	10	0.03	1.6113	1/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	8.5	0.03	1.5916	10/10

Table 10: Finite difference results for $n = 10\,000$ using different increments h.

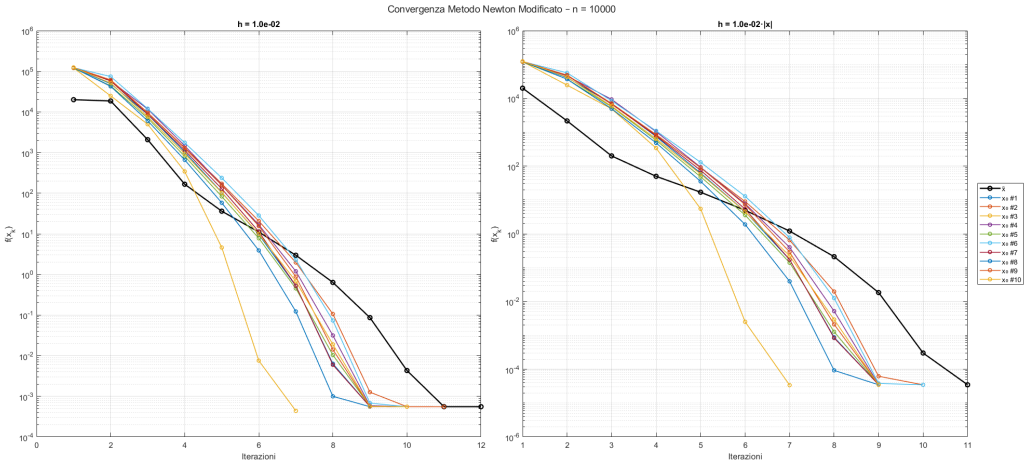


Figure 26: Convergence of Modified Newton method on Generalized Broyden function ($n = 10000$) with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

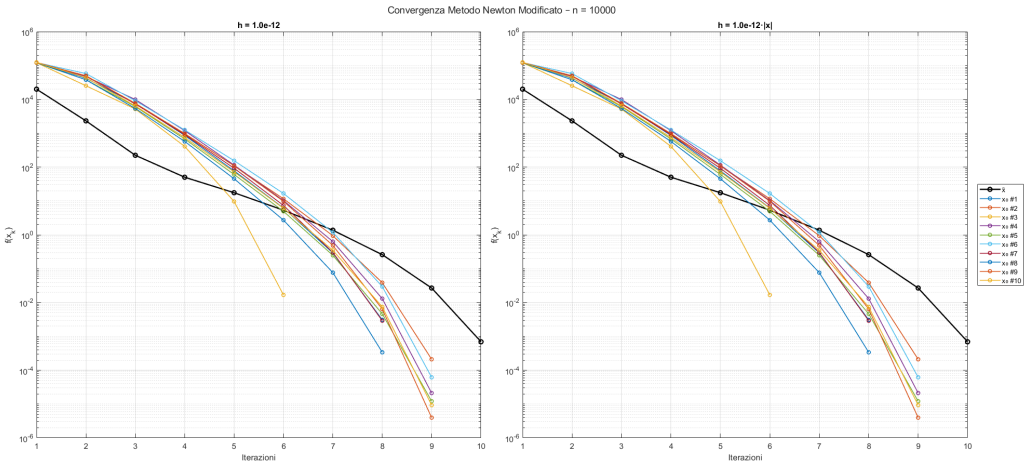


Figure 27: Convergence of Modified Newton method on Generalized Broyden function ($n = 10000$) with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

- $\boldsymbol{n = 100\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	14	0.51	0.0031	1/1
10^{-2}	Avg (10 pts)	10.9	0.56	0.0424	10/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	11	0.47	0.1670	1/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	9.6	0.49	0.1538	10/10
10^{-4}	$\bar{x}$	10	0.36	1.6064	1/1
10^{-4}	Avg (10 pts)	8.7	0.43	1.6182	10/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	10	0.34	1.6122	1/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	8.7	0.43	1.6173	10/10
10^{-6}	$\bar{x}$	10	0.35	1.6112	1/1
10^{-6}	Avg (10 pts)	8.7	0.43	1.6171	10/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	10	0.35	1.6113	1/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	8.7	0.43	1.6170	10/10
10^{-8}	$\bar{x}$	10	0.35	1.6113	1/1
10^{-8}	Avg (10 pts)	8.7	0.43	1.6170	10/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	10	0.35	1.6113	1/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	8.7	0.43	1.6170	10/10
10^{-10}	$\bar{x}$	10	0.35	1.6113	1/1
10^{-10}	Avg (10 pts)	8.7	0.43	1.6170	10/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	10	0.35	1.6113	1/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	8.7	0.43	1.6170	10/10
10^{-12}	$\bar{x}$	10	0.36	1.6113	1/1
10^{-12}	Avg (10 pts)	8.7	0.43	1.6170	10/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	10	0.35	1.6113	1/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	8.7	0.43	1.6170	10/10

Table 11: Finite difference results for $n = 100\,000$ using different increments h.

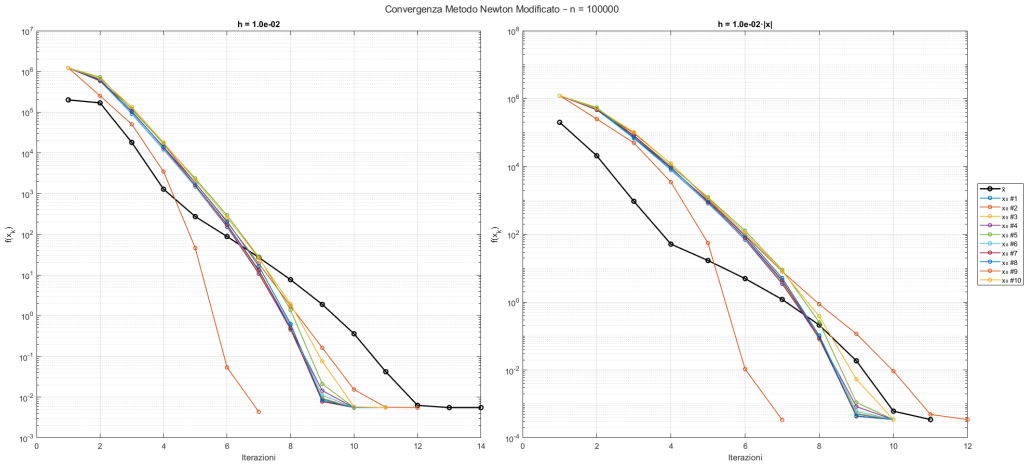


Figure 28: Convergence of Modified Newton method on Generalized Broyden function ($n = 100000$) with fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

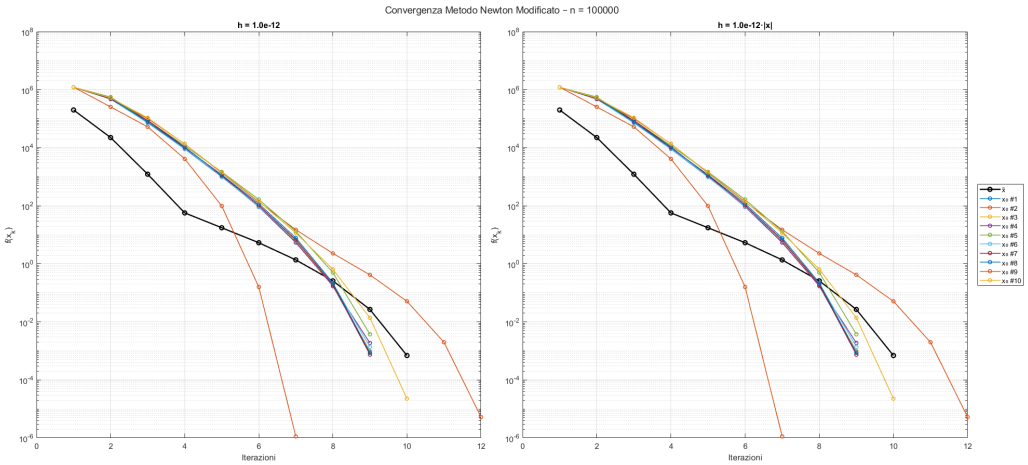


Figure 29: Convergence of Modified Newton method on Generalized Broyden function ($n = 100000$) with fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

4.3 Nelder–Mead method

In this subsection are shown the results obtained from the minimization of the Generalized Broyden function using the Nelder–Mead method. Before presenting the results, a general experimental setup is given:

- `n` $= 10, 26, 50$;
- `max_iter` $= 80.000$;
- `tol` $= 10^{-6}$.

Moreover, for each run have recorded:

- number of iterations to convergence;
- CPU time;
- final objective value found f_{min};
- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported.

Dimension	**Starting point**	$f_{\min}$	**Iter**	**Time (s)**	ρ
10	$\bar{x}$	0.036184	1510	0.03	-1.2743
	Avg (10 pts)	4.974638	1467	0.018	0.3595
26	$\bar{x}$	0.187379	7040	0.11	-0.7587
	Avg (10 pts)	45.657701	11562	0.169	-3.0863
50	$\bar{x}$	0.206557	36027	0.83	-0.6742
	Avg (10 pts)	92.211543	52928	1.218	-2.2175

Table 12: Results of Nelder–Mead on Generalized Broyden tridiagonal function.

Experimental results. The Nelder–Mead method performs reasonably well in very low dimensions, such as $n = 10$, where it achieves successful convergence in most runs and a moderate number of iterations. However, as the problem dimension increases, the method becomes significantly less effective. For $n = 25$ and $n = 50$, none of the runs achieved a function value below the target threshold, and both the number of iterations and runtime increase considerably. The convergence rate ρ also becomes unstable, with values indicating erratic or oscillatory behavior. These results highlight the limitations of Nelder–Mead in structured, high-dimensional, non-separable optimization problems, especially when compared to Newton-type methods equipped with derivative information.

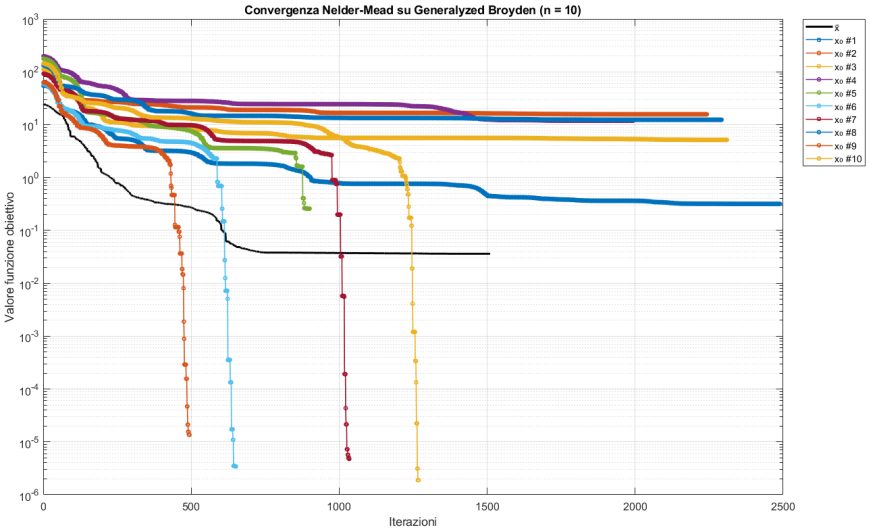


Figure 30: Convergence of Nelder-Mead method on Generalized Broyden function ($n = 10$) for the reference point $\bar{x}$ (black) and 10 randomly generated starting points.

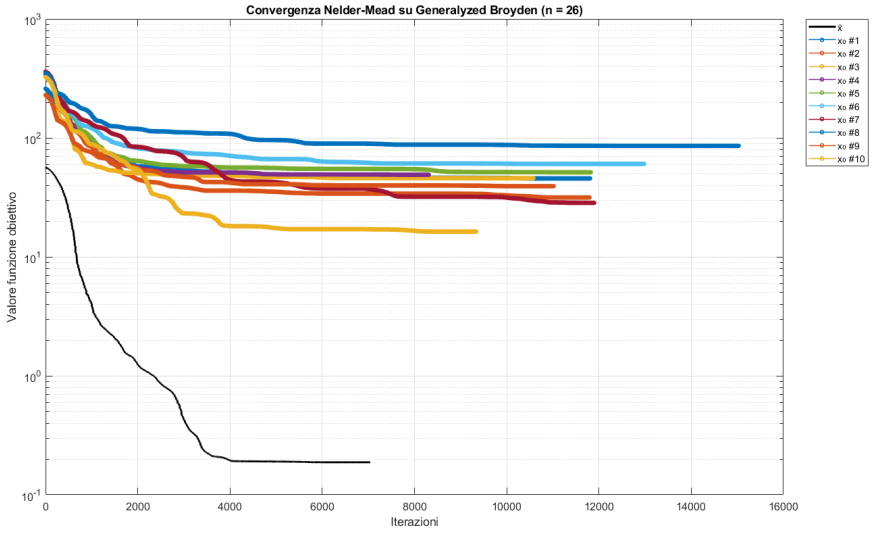


Figure 31: Convergence of Nelder-Mead method on Generalized Broyden function ($n = 26$) for the reference point $\bar{x}$ (black) and 10 randomly generated starting points.

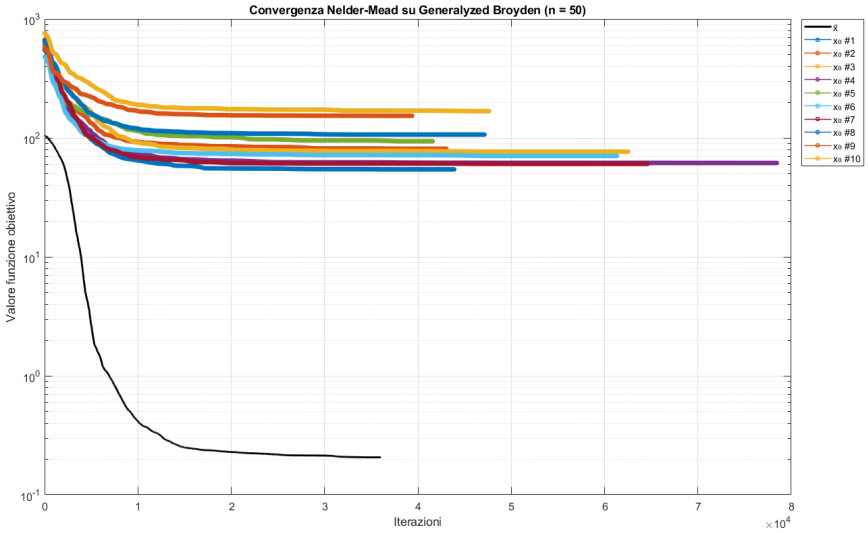


Figure 32: Convergence of Nelder-Mead method on Generalized Broyden function ($n = 50$) for the reference point $\bar{x}$ (black) and 10 randomly generated starting points.

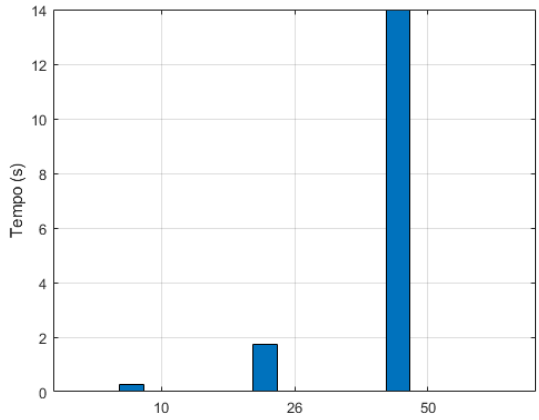


Figure 33: Computational time (in seconds) for Nelder-Mead method applied to the Generalized Broyden function for increasing dimensions.

5 Banded Trigonometric Function

5.1 Problem introduction

The Banded Trigonometric function is a structured nonlinear objective function characterized by trigonometric dependencies among three consecutive variables. It is defined as:

$$f(x) = \sum_{i=1}^{n} i \left[(1 - \cos(x_i)) + \sin(x_{i-1}) - \sin(x_{i+1}) \right],$$

with boundary conditions $x_0 = x_{n+1} = 0$, where $x \in \mathbb{R}^n$. This objective function arises in problems where local periodic variations influence adjacent terms. Its structure creates a banded dependency pattern, meaning each component interacts only with its two immediate neighbors. Because of the involvement of the sine and cosine trigonometric functions, the Banded Trigonometric function has a single minimum but not a single minimizer, as can be seen in the plot reported in figure (34). In this chapter we are going to analyze the behaviour of both the Modified Newton and the Nelder–Mead methods, when they are applied to minimize the Banded Trigonometric function. The initial points suggested in the benchmark library are $\bar{x} \in \mathbb{R}^n$ such that

$$\bar{x}_k = 1.0 \quad \text{for} \quad k = 1, 2, \ldots, n,$$

together with another 10 random initial points sampled uniformly in the hypercube $[\bar{x}_1 - 1, \bar{x}_1 + 1] \times \cdots \times [\bar{x}_n - 1, \bar{x}_n + 1] \subset \mathbb{R}^n$, starting from $\bar{x}$.

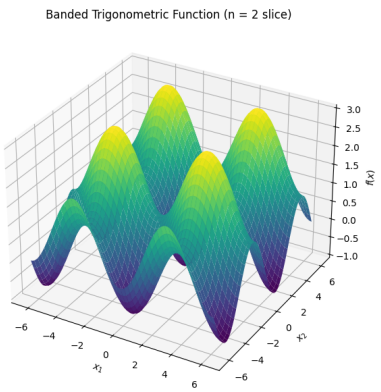


Figure 34: 3D visualization of the Banded Trigonometric function for $n = 2$.

5.2 Modified Newton method

In this subsection are shown the results obtained from the minimization of the Banded Trigonometric function using the Modified Newton method; the study includes both exact derivatives and derivatives computed by finite-differences approximations. Due to its diagonal structure, the hessian is stored and manipulated in sparse format; this choice drastically reduces memory requirements and computational cost (in both matrix factorization and Newton direction computation), allowing efficiently handle large–scale problems.

For this particular function the resulting hessian matrices are often ill-conditioned, especially in higher dimensions. To address this, the Modified Newton method were designed to apply preconditioning through the MATLAB `pcg` solver (Preconditioned Conjugate Gradient). This choice enables efficient direction computation even when Cholesky factorization becomes unreliable or fails, particularly for large-scale sparse systems. The algorithm applies a modified Cholesky approach (Algorithm 6.3 in) to shift the hessian until a positive definite approximation is obtained. Then, an incomplete Cholesky factorization is used to precondition the matrix, ensuring robust convergence even in the presence of large condition numbers.

Before presenting detailed outcomes, a general experimental setup is given:

- `n` $= 10^3, 10^4, 10^5$;
- `max_iter` $= 5000$;
- `tol` $= 10^{-6}$;
- `max_iter_pcg` $= 50$;
- `tol_pcg` $= 10^{-6}$;
- `preconditioner_pcg` $= LL^T$, where the factor L is obtained with the incomplete Cholesky factorization of the coefficient matrix.

For each run, the following were tracked:

- number of iterations to convergence;
- CPU time;
- number of successful runs;

- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported. It should be emphasized that among the data shown within the tables, rho values approximately equal to or greater than three can be identified. These results are not a correct approximation of the required rate of convergence, as this is at most quadratic in the case of Modified Newton method; These results could be traced to replacing the optimal value of the minimizer with its last available approximation.

Furthermore, the code is designed to work also with finite–differences case, and this variant is implemented thanks to the extra parameters

- `h`: a parameter equal to 10^{-k}, for $k = 2, 4, 6, 8, 10, 12$;
- `type`: a parameter which indicates if the increment is scaled componentwise as $h_i = 10^{-k} \cdot |x_i|$ or if it is a default increment such that $h_i = 10^{-k}$.

For this part, the several features were computed for each stepsize h.

5.2.1 Modified Newton method with exact derivatives

In this section the Banded Trigonometric function is minimized using the Modified Newton method with exact gradient and hessian formulas. Both were derived analytically and implemented exploiting the banded pattern of the function. Specifically:

- each gradient's component is constructed as:

$$\frac{\partial F}{\partial x_k} = \begin{cases} \sin(x_1) + 2\cos(x_1), & i = 1 \\ k\sin(x_k) + 2\cos(x_k), & 2 \leq k \leq n-1 \\ n\sin(x_n) - (n-1)\cos(x_n), & k = n \end{cases};$$

- each entry of the hessian matrix is computed as:

$$\frac{\partial^2 F}{\partial x_k \partial x_j} = \begin{cases} \cos(x_1) - 2\sin(x_1), & i = 1 \\ k\cos(x_k) - 2\sin(x_k), & 2 \leq k \leq n-1 \\ n\cos(x_n) + (n-1)\sin(x_n), & i = n \\ 0, & \text{otherwise} \end{cases}.$$

Experimental Results. As shown in table (5.2.1), the Modified Newton method with exact derivatives fails in terms of minimizing the Banded Trigonometric function, which turns out to be a particularly challenging function to optimize, if compared to the ones in the other sections. In all dimensions tested and for all initialization points, not only is convergence not satisfied, but the f_{min} values achieved are far from the actual global minimum. Despite preconditioning the hessian matrix during the search for the descent direction, the algorithm fails to reach a value that is reasonably close to the minimum of the function. Below are tables and figures explaining the situation in detail.

Dim	**Init.**	**Iter**	**Time (s)**	$f_{\min}$	ρ	**Successes**
10^3	$\bar{x}$	6	0.04	-427.40	2.52	1/1
	Avg (10 pts)	16.4	0.02	-565.12	2.62	10/10
10^4	$\bar{x}$	7	0.04	-4159.93	2.22	1/1
	Avg (10 pts)	20.4	0.22	-2835.42	2.25	10/10
10^5	$\bar{x}$	6	0.36	-41443.76	4.11	1/1
	Avg (10 pts)	29.7	4.78	-37052.49	1.78	10/10

Table 13: Results of Modified Newton method on Banded Trigonometric function using exact derivatives.

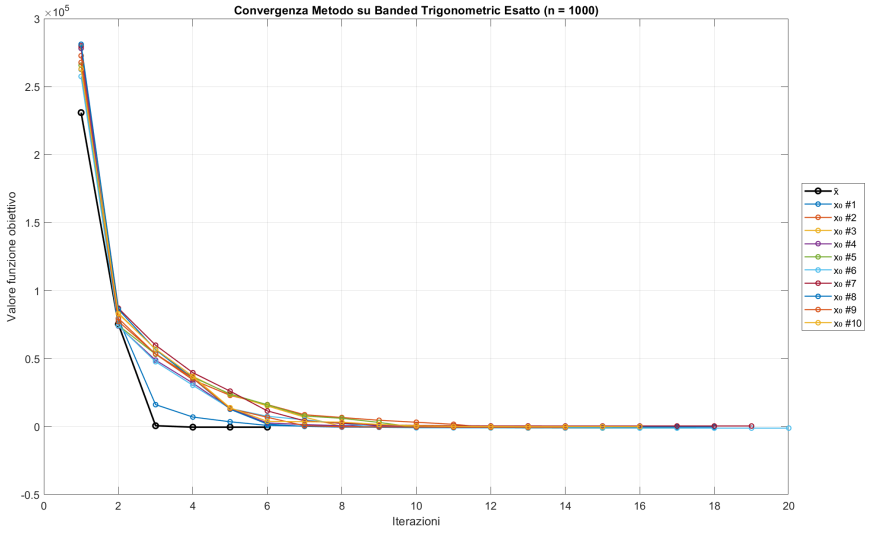


Figure 35: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 1000$) using exact derivatives.

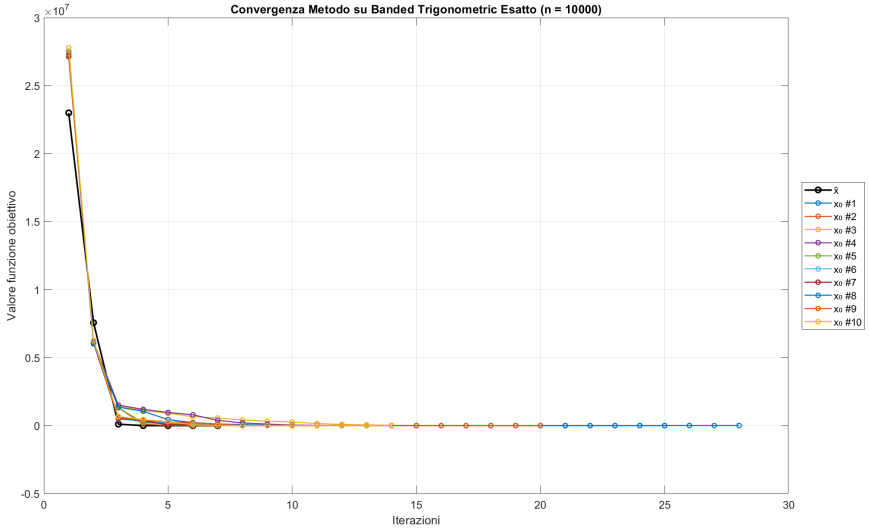


Figure 36: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 10000$) using exact derivatives.

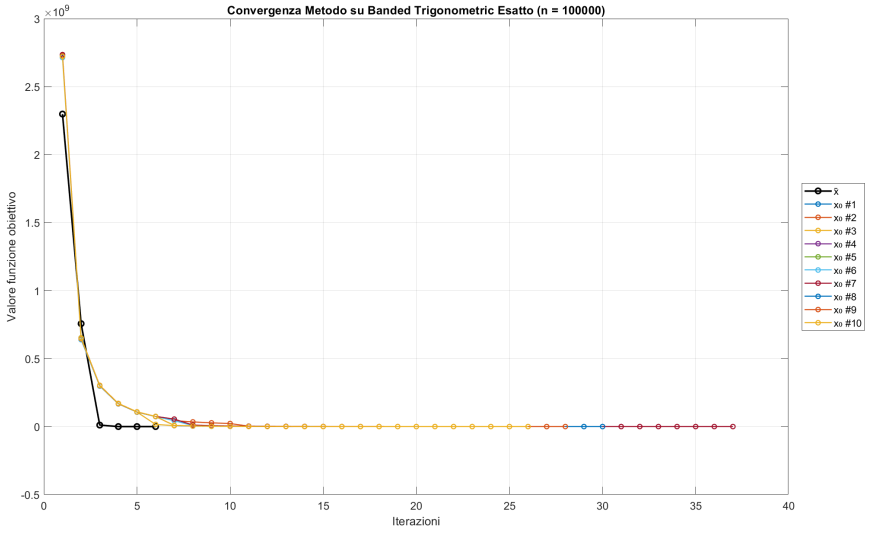


Figure 37: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 100000$) using exact derivatives.

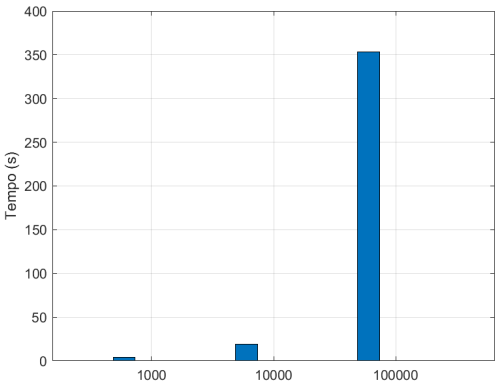


Figure 38: Computational time required by the Modified Newton method on the Banded Trigonometric function with exact derivatives for each problem size.

5.2.2 Modified Newton method with approximated derivatives

In this experiment, we use the Modified Newton method on the Banded Trigonometric function with gradient and hessian computed via finite differences. The approximated gradient and hessian components were expanded manually and implemented directly to avoid cancellation and redundancy. The sparsity pattern of the exact hessian was preserved to limit computational cost. The expressions used for approximating the gradient and the hessian are reported below, following the same structure adopted in previous sections.

Gradient approximation. Due to the structure of the function, only a small number of addends to the summation survive when computing the numerator in the formula (1). These terms are identified by the following function:

$$g(x) = \sum_{i=k-1}^{k+1} i \left[(1 - \cos(x_i)) + \sin(x_{i-1}) - \sin(x_{i+1}) \right].$$

Then, using subtraction and addition formulas for sine and cosine, and fixed k as the component whose derivative is being approximated:

$$\frac{\partial F}{\partial x_k} \approx \frac{g(x + he_k) - g(x + he_k)}{2h} = \begin{cases} \frac{4 \cos x_k \sin h + 2k \sin x_k \sin h}{2h}, & 1 \leq k < n \\ \frac{2n \sin x_n \sin h - 2(n-1) \cos x_k \sin h}{2h}, & k = n \end{cases}. \tag{7}$$

Hessian approximation. Also in the hessian computation, the definition of the function $g(x)$ can be used to describe the terms in the numerator of (2). Rearranging the terms and using a Taylor expansion for the terms involving the cosine of h, is obtained:

$$\frac{\partial^2 F}{\partial x_k \partial x_j} \approx \begin{cases} (k \cos x_k - 2 \sin x_k) - h(k \sin x_k + 2 \cos x_k), & 1 \leq k = j < n \\ (n \cos x_n - (n-1) \sin x_n) - h(n \sin x_n - (n-1) \cos x_n), & k = j = n \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

Experimental Results. Also in this cas the Modified Newton method in unable to correctly optimize the Banded Trigonometric function. In all dimensions tested and for all starting points (suggested or randomly generated), convergence is not met, and the algorithm stops after a few iterations regardless of whether the increment used in calculating finite differences is fixed or scaled by component. In this case it makes no sense to talk about the rate of convergence of the method implemented, in fact in some cases it cannot even be calculated (NaN). More precise results are captured in the following tables and figures, divided by problem size.

- $\boldsymbol{n = 1\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	6	0.33	8.1635	1/1
10^{-2}	Avg (10 pts)	15.9	0.03	5.44	7/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/10
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	519	0.39	3.50	7/10
10^{-4}	$\bar{x}$	7	0.03	1.8294	0/1
10^{-4}	Avg (10 pts)	18.1	0.03	5.35	9/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	16.3	0.02	6.00	8/10
10^{-6}	$\bar{x}$	7	0.01	1.1623	1/1
10^{-6}	Avg (10 pts)	15.8	0.02	5.39	6/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	15.5	0.02	5.20	7/10
10^{-8}	$\bar{x}$	7	0.01	6.4828	1/1
10^{-8}	Avg (10 pts)	14.9	0.02	6.45	6/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	15	0.02	7.63	6/10
10^{-10}	$\bar{x}$	7	0.01	6.4832	1/1
10^{-10}	Avg (10 pts)	15.1	0.02	7.15	6/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	15.1	0.03	7.28	7/10
10^{-12}	$\bar{x}$	7	0.01	6.4832	1/1
10^{-12}	Avg (10 pts)	15.1	0.02	7.01	7/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	1	0.00	NaN	0/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	15.1	0.03	7.01	7/10

Table 14: Finite difference results for $n = 1\,000$ using different increments h and strategies.

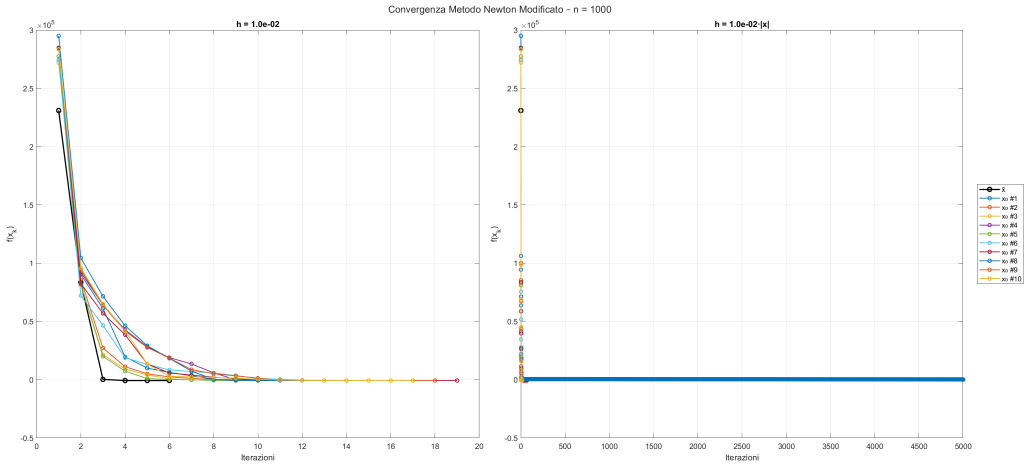


Figure 39: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 1000$) using fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

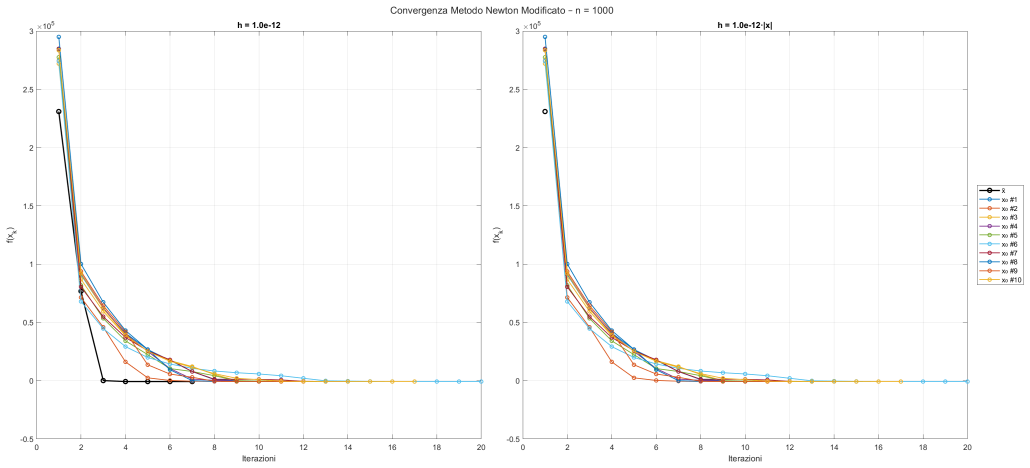


Figure 40: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 1000$) using fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

- $\boldsymbol{n = 10\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	6	0.05	-7087.89	1/1
10^{-2}	Avg (10 pts)	21.6	0.25	3.65	9/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	31.9	0.27	2.84	7/10
10^{-4}	$\bar{x}$	7	0.06	1.83	0/1
10^{-4}	Avg (10 pts)	20.8	0.24	6.73	8/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	20	0.23	5.13	8/10
10^{-6}	$\bar{x}$	7	0.06	1.16	1/1
10^{-6}	Avg (10 pts)	20.5	0.23	4.48	8/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	19.3	0.22	1.11	7/10
10^{-8}	$\bar{x}$	7	0.05	3.4828	1/1
10^{-8}	Avg (10 pts)	20.2	0.23	6.38	9/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	19.8	0.22	6.30	8/10
10^{-10}	$\bar{x}$	7	0.05	6.4832	1/1
10^{-10}	Avg (10 pts)	20.2	0.22	4.87	9/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	20.5	0.23	4.48	9/10
10^{-12}	$\bar{x}$	7	0.06	6.4832	1/1
10^{-12}	Avg (10 pts)	20.1	0.22	4.98	9/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	1	0.01	NaN	0/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	20.2	0.22	4.69	9/10

Table 15: Finite difference results for $n = 10\,000$ using different increments h and strategies.

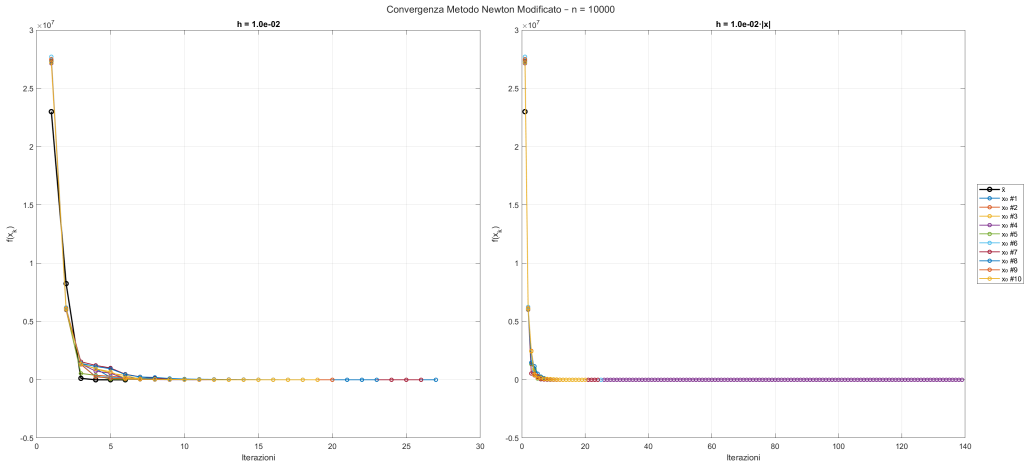


Figure 41: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 10000$) using fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

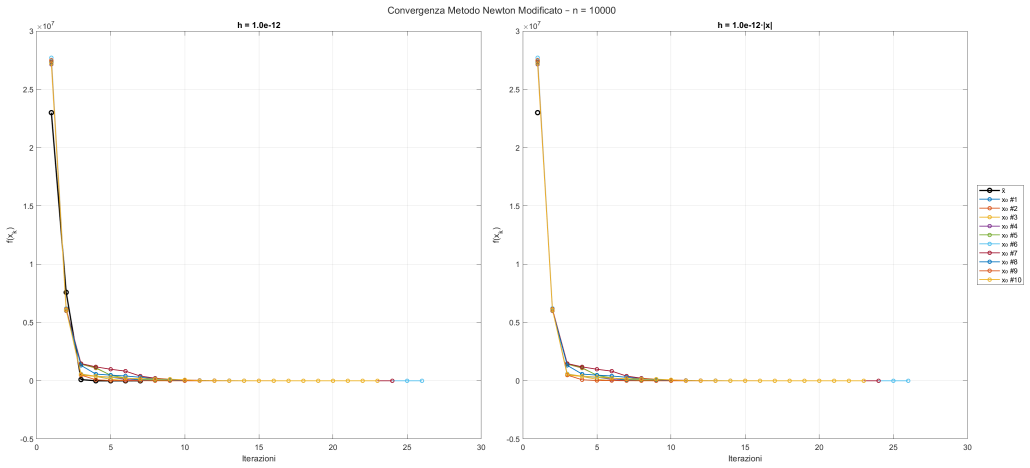


Figure 42: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 10000$) using fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

- $\boldsymbol{n = 100\,000}$

Increment	**Init.**	**Iter**	**Time (s)**	ρ	**Successes**
10^{-2}	$\bar{x}$	6	0.37	8.1604	1/1
10^{-2}	Avg (10 pts)	29.3	4.59	6.36	8/10
$10^{-2} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-2} \cdot \lvert x \rvert$	Avg (10 pts)	41.3	7.00	4.22	8/10
10^{-4}	$\bar{x}$	7	0.50	1.8294	0/1
10^{-4}	Avg (10 pts)	28.4	4.61	4.51	8/10
$10^{-4} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-4} \cdot \lvert x \rvert$	Avg (10 pts)	29	4.59	5.31	8/10
10^{-6}	$\bar{x}$	7	0.50	1.1623	1/1
10^{-6}	Avg (10 pts)	28.9	4.92	6.26	8/10
$10^{-6} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-6} \cdot \lvert x \rvert$	Avg (10 pts)	28.6	4.72	8.31	7/10
10^{-8}	$\bar{x}$	7	0.58	6.4828	1/1
10^{-8}	Avg (10 pts)	29.6	4.70	4.01	8/10
$10^{-8} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-8} \cdot \lvert x \rvert$	Avg (10 pts)	30.8	5.19	6.75	8/10
10^{-10}	$\bar{x}$	7	0.53	6.4832	1/1
10^{-10}	Avg (10 pts)	28	4.43	5.83	9/10
$10^{-10} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-10} \cdot \lvert x \rvert$	Avg (10 pts)	31.1	4.95	2.63	9/10
10^{-12}	$\bar{x}$	7	0.51	6.4832	1/1
10^{-12}	Avg (10 pts)	29.8	4.83	0.88	8/10
$10^{-12} \cdot \lvert x \rvert$	$\bar{x}$	1	0.09	NaN	0/1
$10^{-12} \cdot \lvert x \rvert$	Avg (10 pts)	30.9	4.86	4.18	8/10

Table 16: Finite difference results for $n = 100\,000$ using different increments h and strategies.

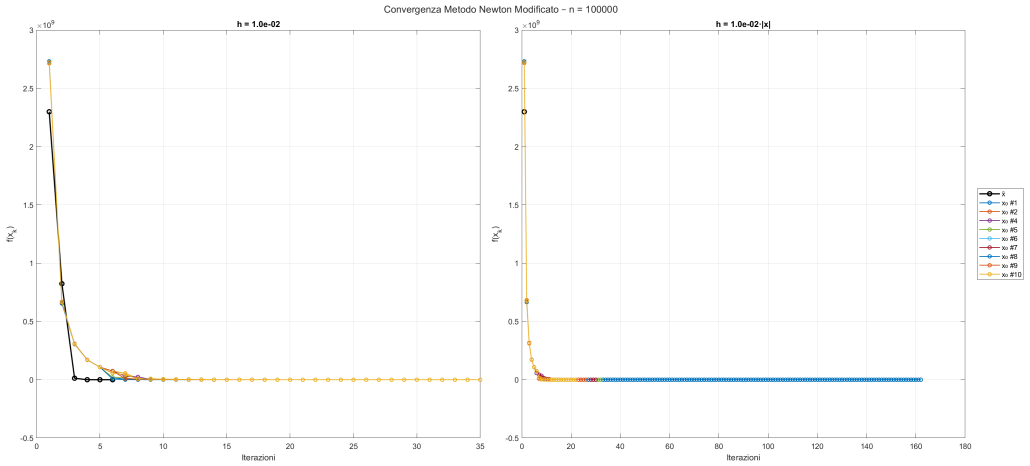


Figure 43: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 100000$) using fixed increment $h = 10^{-2}$ (left) and scaled increment $h = 10^{-2} \cdot |x|$ (right).

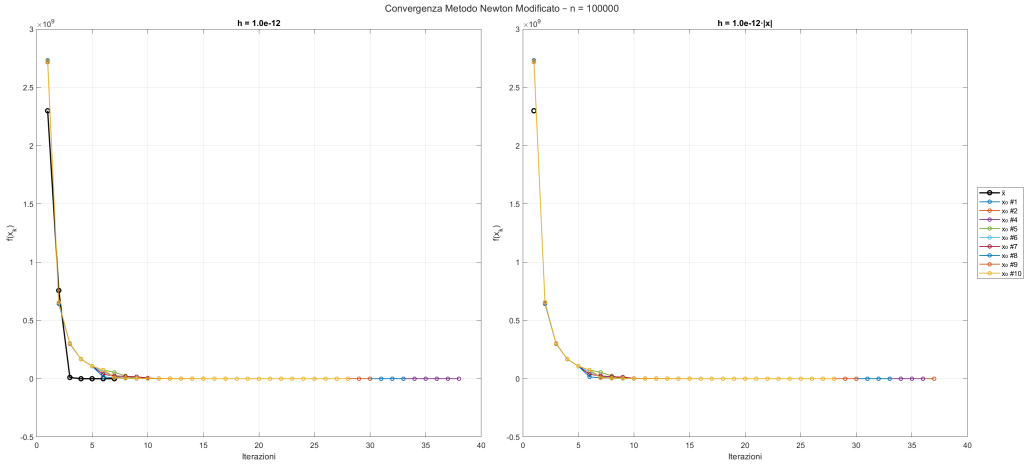


Figure 44: Convergence of the Modified Newton method on the Banded Trigonometric function ($n = 100000$) using fixed increment $h = 10^{-12}$ (left) and scaled increment $h = 10^{-12} \cdot |x|$ (right).

5.3 Nelder–Mead method

In this subsection are shown the results obtained from the minimization of the Banded Trigonometric function using the Nelder–Mead method. This method does not exploit derivative information and therefore struggles more with ill-conditioned landscapes and high-dimensional domains. As expected, the performance worsens as the dimension increases.

Before presenting the results, a general experimental setup is given:

- `n` $= 10, 26, 50$;
- `max_iter` $= 100.000$
- `tol` $= 10^{-6}$.

Moreover, for each run have been recorded:

- number of iterations to convergence;
- CPU time;
- final objective value found f_{min};
- experimental rate of convergence ρ:

$$\rho \approx \frac{\log\left(\|x^{(k+1)} - x^{(k)}\| \big/ \|x^{(k)} - x^{(k-1)}\|\right)}{\log\left(\|x^{(k)} - x^{(k-1)}\| \big/ \|x^{(k-1)} - x^{(k-2)}\|\right)}.$$

In correspondence with the randomly generated points, an average behavior of each of the previous catogories is reported.

Experimental results. The experimental results for the Banded Trigonometric function confirm the limitations of the Nelder–Mead method in high-dimensional and structured nonlinear problems. For $n = 10$, the method achieves very low computational times and iteration counts, but only 2 out of 10 runs with random initializations are successful, and function values remain distant from the global minimum in most cases. As the dimension increases to $n = 26$ and $n = 50$, the number of iterations and total computational time increase drastically, with several runs hitting the iteration cap (notably, two runs at $n = 26$ required 80000 iterations and over 50 seconds). No successful runs are observed for these higher dimensions, and

the convergence rates ρ become highly erratic, alternating between extreme positive and negative values, indicating a lack of stability and robustness. Overall, the method fails to provide accurate or reliable solutions in medium and high dimensions, reinforcing the need for derivative-based or more advanced global optimization methods when tackling large-scale, structured, and highly nonlinear problems.

Dimension	**Starting point**	$f_{\min}$	**Iter**	**Time (s)**	ρ
10	$\bar{x}$	-2.180187	187	0.01	0.5000
	Avg (10 pts)	-9.743851	1297	0.019	0.5390
26	$\bar{x}$	-6.731719	1376	0.04	-7.0805
	Avg (10 pts)	2.086111	77855	6.393	4.1805
50	$\bar{x}$	-5.995540	4955	0.28	-1.8208
	Avg (10 pts)	-2.092789	65323	0.429	0.1752

Table 17: Results of Nelder–Mead on Banded Trigonometric function.

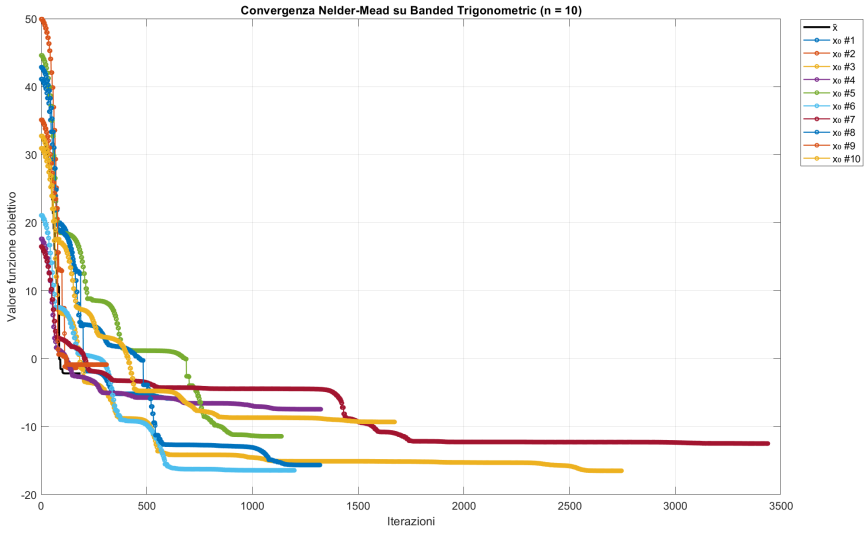


Figure 45: Convergence of Nelder-Mead on the Banded Trigonometric function with $n = 10$.

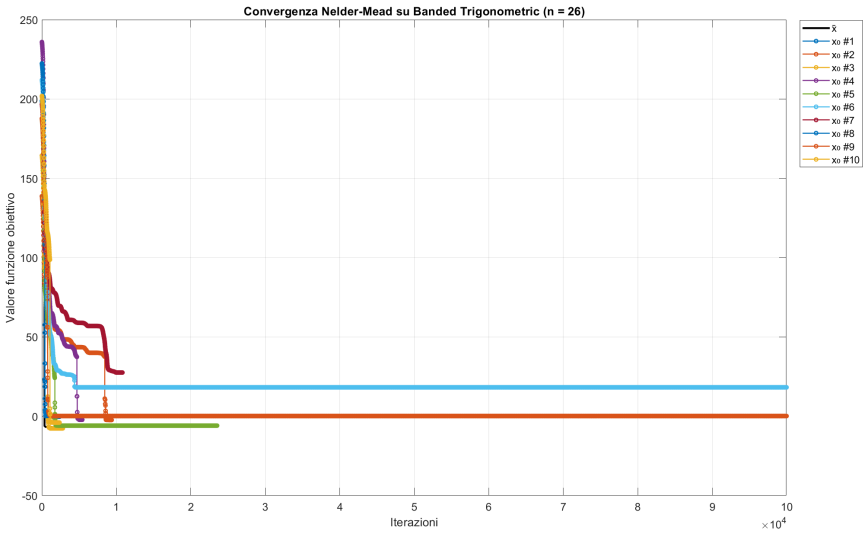


Figure 46: Convergence of Nelder-Mead on the Banded Trigonometric function with $n = 26$.

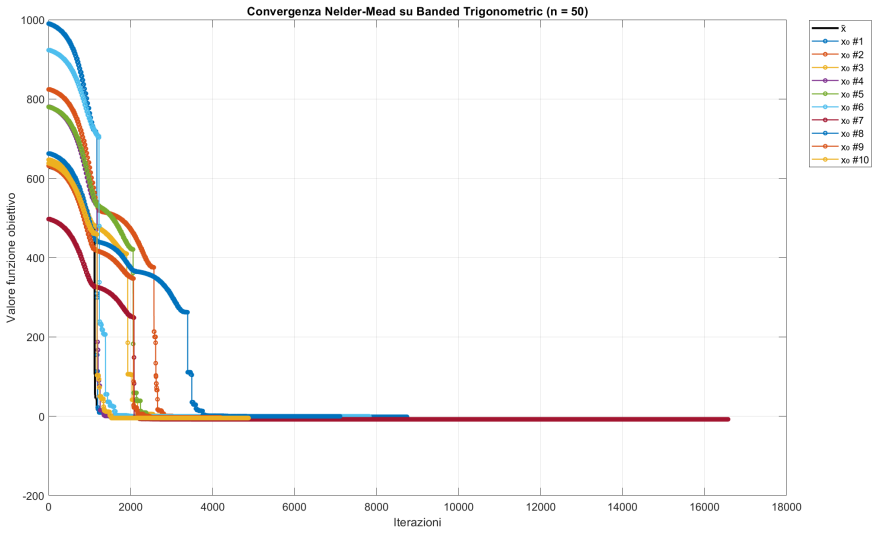


Figure 47: Convergence of Nelder-Mead on the Banded Trigonometric function with $n = 50$.

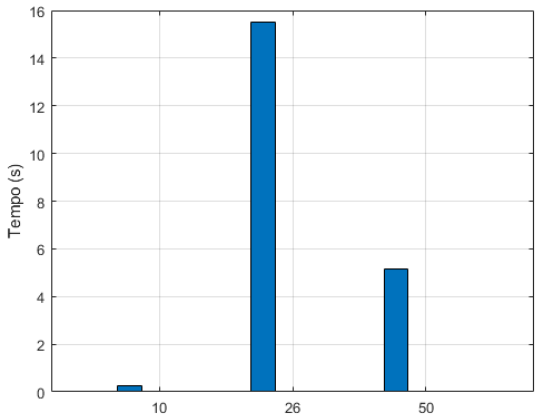


Figure 48: Computation time of Nelder-Mead on the Banded Trigonometric function for $n \in \{10, 26, 50\}$.

6 Conclusions

In this project, we implemented and compared the performance of two numerical optimization methods—Modified Newton and Nelder–Mead—on three benchmark unconstrained problems:

1. Extended Rosenbrock function,
2. Generalized Broyden Tridiagonal function,
3. Banded Trigonometric function.

The Modified Newton method consistently delivered superior results across all tested functions and dimensions. When exact derivatives were available, the method achieved rapid convergence—often in fewer than 10 iterations—and scaled efficiently even in high-dimensional settings (e.g. $n = 10^5$), thanks to sparse linear algebra and preconditioning techniques. Finite difference approximations, although more computationally expensive and slightly less accurate, proved to be reliable when properly tuned, particularly with smaller step sizes (e.g. $h \leq 10^{-6}$).

In contrast, the Nelder–Mead method showed acceptable performance only in low-dimensional cases. While easy to implement and free of derivative requirements, its convergence became erratic and inefficient as the dimension increased. The algorithm often failed to reach low objective values, especially in problems with ill-conditioned or structured landscapes, such as the Banded Trigonometric function.

Overall, this study highlights the importance of leveraging second-order information and exploiting problem structure—such as sparsity or tridiagonality—for scalable and reliable optimization. Derivative-free methods may still be useful in small-scale or noisy settings, but for large, structured problems, Newton-type algorithms remain the preferred choice.