Analysis of Numerical Methods for Unconstrained Optimization

Course: Numerical Optimization for Large scale problems and stochastic Optimization

Abedal Salam Al Ashi Abou Shoushe
Politecnico di Torino
s336648

Methods Used: Modified Newton Method and Steepest Descent Method

Code Repository

Date: December 7, 2024

Abstract

This report focuses on numerical optimization methods used to solve unconstrained problems. Two optimization techniques, the Modified Newton Method and the Steepest Descent Method, are assessed and contrasted. The results provide insights into the strengths and limitations of these approaches.

1 Introduction

Unconstrained optimization is a fundamental problem in numerical analysis and optimization, where the objective is to find a local or global minimum of a function without any constraints on its variables. This type of optimization plays a critical role in various applications, including machine learning, engineering design, and scientific simulations.

This report focuses on the implementation and analysis of two numerical methods for solving unconstrained optimization problems: the Modified Newton Method and the Steepest Descent Method. These methods are widely used due to their simplicity and effectiveness in handling smooth, nonlinear functions. The performance of these methods is evaluated on the well-known Rosenbrock function and three additional test problems: the Extended Powell Singular Function, the Extended Rosenbrock Function, and the Chained Rosenbrock Function.

The objective of this study is to assess the efficiency and reliability of the two methods by analyzing key metrics such as convergence rate, number of iterations, and computational time. Additionally, the theoretical behavior of the methods is compared with their practical performance to provide deeper insights into their applicability.

2 Methods [1]

2.1 Backtracking Line Search

Both optimization methods rely on backtracking line search to determine the step size α_k at each iteration. The line search ensures sufficient decrease in the objective function, satisfying the Armijo condition:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f(x_k)^T p_k,$$

where p_k is the descent direction, and $c \in (0, 1)$ is a predefined parameter. Starting from an initial step size (e.g., $\alpha = 1$), the algorithm iteratively reduces α by multiplying it with $\rho \in (0, 1)$ until the condition is met. In this implementation, $\rho = 0.5$ and $c = 10^{-4}$ are used consistently.

2.2 Modified Newton Method

The Modified Newton Method improves upon the standard Newton's method by ensuring that the Hessian matrix remains positive definite. This is achieved by regularizing the Hessian:

$$H_k = \nabla^2 f(x_k) + \tau I,$$

where $\tau > 0$ is a small regularization parameter. The descent direction is computed by solving:

$$H_k p_k = -\nabla f(x_k).$$

If the Hessian is non-invertible, the method falls back to the steepest descent direction. Backtracking line search is used to determine the step size α_k. The method iteratively updates the solution as:

$$x_{k+1} = x_k + \alpha_k p_k.$$

This approach ensures stability and convergence, even for ill-conditioned problems.

2.3 Steepest Descent Method

The Steepest Descent Method uses the gradient of the objective function as the direction of descent:

$$p_k = -\nabla f(x_k).$$

While simple to implement, this method may exhibit slow convergence near saddle points or ill-conditioned regions. The step size α_k is determined using backtracking line search, ensuring sufficient decrease in the objective function. The update rule for the solution is:

$$x_{k+1} = x_k + \alpha_k p_k.$$

This method is computationally efficient due to its reliance only on gradient information but may require many iterations to converge in certain cases.

2.4 Implementation Details

The optimization methods were implemented in Python, incorporating the following critical features:

- **Exact Derivatives**: Used when analytic gradients and Hessians are available.
- **Finite Differences**: Central difference scheme with $h = 10^{-5}$ for gradient and Hessian approximations.
- **Stopping Criteria**: Terminate when $\|\nabla f(x_k)\| < 10^{-5}$ or max_iter is reached.
- **Line Search Parameters**: $\rho = 0.5$, $c = 10^{-4}$ for backtracking line search.
- **Hessian Regularization**: Add 10^{-4} to the Hessian diagonal for stability.
- **Logging**: Record metrics like cost, gradient norms, times, and convergence rates.
- **Trajectory Tracking**: Track iterates (x_k) for process visualization.

3 Test Problems

3.1 Rosenbrock Function

The Rosenbrock function, often called the "banana function," is a standard test function in optimization and numerical analysis. It is widely used to evaluate the performance of optimization algorithms due to its challenging, narrow, and curved valley leading to the global minimum. The function is defined as:

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

with a global minimum at the point $(1, 1)$, where $f(1, 1) = 0$. For this experiment, two starting points were tested: $[1.2, 1.2]$ and $[-1.2, 1]$.

3.1.1 Initial Point: [1.2, 1.2]

Before optimization, the cost of the Rosenbrock function at this point was 5.8. The results for both methods are summarized in Table 1. The convergence paths of both methods, visualized in Figure 1, illustrate the trajectories toward the global minimum.

Table 1: Performance on Rosenbrock Function with Initial Point $[1.2, 1.2]$

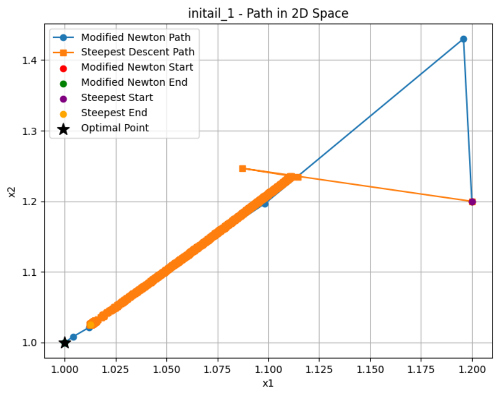
Metric	**Modified Newton**	**Steepest Descent**
Final Cost	3.85×10^{-14}	0.0001732
Iterations	7	1000
Execution Time (s)	0.0009	0.0348
Gradient Norm	1.53×10^{-6}	0.1684
Convergence Rate	0.4813	1.0032
Success	True	False

3.1.2 Initial Point: [-1.2, 1]

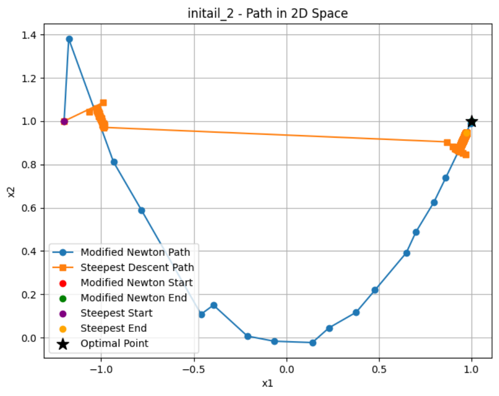
Before optimization, the cost of the Rosenbrock function at this point was 24.2. The results for both methods are summarized in Table 2. The optimization paths for both methods, shown in Figure 1, highlight the different approaches to navigating the complex valley structure of the function.

Table 2: Performance on Rosenbrock Function with Initial Point $[-1.2, 1]$

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	8.00×10^{-19}	0.0007484
Iterations	21	1000
Execution Time (s)	0.00596	0.0444
Gradient Norm	2.51×10^{-9}	0.0739
Convergence Rate	0.6899	0.9915
Success	True	False



(a) Initial Point $[1.2, 1.2]$.



(b) Initial Point $[-1.2, 1]$.

Figure 1: Optimization paths for the Rosenbrock function with two initial points.

3.2 Extended Powell Singular Function

The Extended Powell Singular function [2] is a challenging high-dimensional optimization problem often used to test optimization methods. It is defined as:

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

where the components $f_k(x)$ are defined based on the index $k \mod 4$ as:

$$f_k(x) = \begin{cases} x_k + 10x_{k+1}, & \text{if } k \mod 4 = 1, \\ \sqrt{5}(x_{k+2} - x_{k+3}), & \text{if } k \mod 4 = 2, \\ (x_{k+1} - 2x_{k+2})^2, & \text{if } k \mod 4 = 3, \\ \sqrt{10}(x_k - x_{k+3})^2, & \text{if } k \mod 4 = 0. \end{cases}$$

The initial point $\bar{x}$ is defined systematically, following the modulus condition:

$$\bar{x}_l = \begin{cases} 3, & \text{if } l \mod 4 = 1, \\ -1, & \text{if } l \mod 4 = 2, \\ 0, & \text{if } l \mod 4 = 3, \\ 1, & \text{if } l \mod 4 = 0. \end{cases}$$

The initial base point is defined in a 10^3-dimensional space. From this initial base point, 10 random starting points are generated by adding random noise within the hypercube $[x - 1, x + 1]$ for each dimension.

3.2.1 Summary of Results

Table 3 provides an overview of the performance of the Modified Newton and Steepest Descent methods. The Modified Newton Method consistently achieved a significantly lower final cost and faster convergence compared to the Steepest Descent Method. In contrast, the Steepest Descent Method required the maximum number of iterations (1000) in every run and failed to converge to an optimal solution.

Table 3: Summary of Results Mean $\pm$ Std for Extended Powell Singular Function (10 Random Points)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	$8.0 \times 10^{-8} \pm 5.4 \times 10^{-9}$	0.0309 ± 0.001
Iterations	18 ± 0	1000 ± 0
Execution Time (s)	0.3 ± 0.1	5.9 ± 0.3
Gradient Norm	$9.0 \times 10^{-6} \pm 4.3 \times 10^{-7}$	0.14 ± 0.03
Convergence Rate	0.598 ± 0.001	0.989 ± 0.001
Success Rate	100%	0%

3.2.2 Best and Worst Case Performance

To illustrate the performance of the two methods, Tables 4 and 5 highlight the performance metrics for the best-case scenario (Random Point 1) and the worst-case scenario (Random Point 6), respectively. These represent the random points with the lowest and highest overall execution times, cost values, and gradient norms for the combined Modified Newton and Steepest Descent methods.

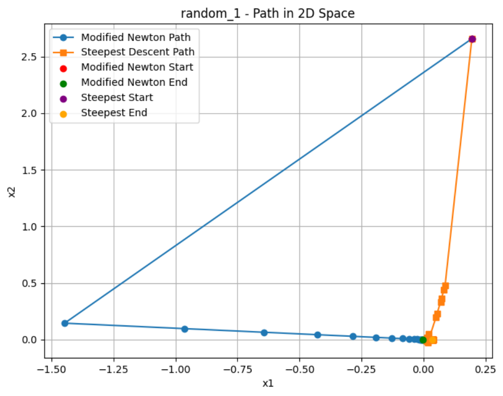
Figure 2 further visualizes the optimization paths for these two random points.

Table 4: Performance Metrics for Best Random Point (Random 1)

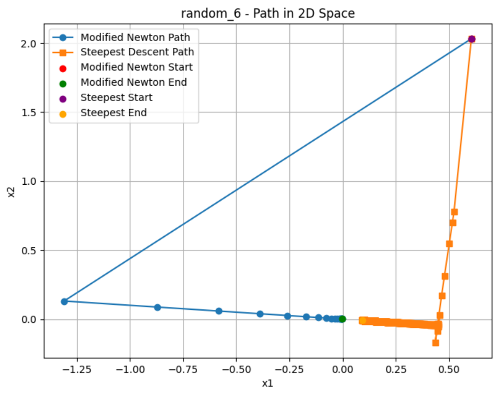
Metric	**Modified Newton**	**Steepest Descent**
Final Cost	8.30×10^{-8}	0.0310
Iterations	18	1000
Execution Time (s)	0.212	5.54
Gradient Norm	9.54×10^{-6}	0.177
Convergence Rate	0.5983	0.9890

Table 5: Performance Metrics for Worst Random Point (Random 6)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	8.10×10^{-8}	0.0299
Iterations	18	1000
Execution Time (s)	0.456	5.79
Gradient Norm	9.35×10^{-6}	0.155
Convergence Rate	0.5981	0.9889



(a) Best Performance (Random Point 1).



(b) Worst Performance (Random Point 6).

Figure 2: Comparison of Best and Worst Random Points for Extended Powell Singular Function.

3.3 Extended Rosenbrock Function

The Extended Rosenbrock function [3] is a widely used optimization problem in high-dimensional settings. It is defined as:

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

where:

$$f_k(x) = \begin{cases} 10(x_k^2 - x_{k+1}), & \text{if } k \mod 2 = 1, \\ x_{k-1} - 1, & \text{if } k \mod 2 = 0. \end{cases}$$

The systematic initial point $\bar{x}$ is defined as:

$$\bar{x}_l = \begin{cases} -1.2, & \text{if } l \mod 2 = 1, \\ 1.0, & \text{if } l \mod 2 = 0. \end{cases}$$

The initial base point is defined in a 10^3-dimensional space. From this initial base point, 10 random starting points are generated by adding random noise within the hypercube $[x-1, x+1]$ for each dimension.

3.3.1 Summary of Results

Table 6 summarizes the performance of both methods. The Modified Newton Method generally achieved a lower final cost and demonstrated higher convergence rates compared to the Steepest Descent Method, which struggled to converge effectively within the given iterations.

Table 6: Summary of Results Mean ± Std for Extended Rosenbrock Function (10 Random Points)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	$1.5 \times 10^{-6} \pm 0.3 \times 10^{-6}$	0.025 ± 0.01
Iterations	1000 ± 0	1000 ± 0
Execution Time (s)	11.5 ± 0.4	6.3 ± 0.2
Gradient Norm	0.004 ± 0.002	0.3 ± 0.2
Convergence Rate	0.987 ± 0.001	0.990 ± 0.001
Success Rate	60%	0%

3.3.2 Best and Worst Case Performance

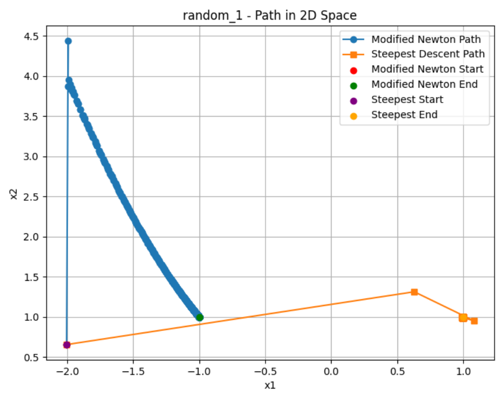
The performance metrics for the best-case scenario (Random Point 1) and the worst-case scenario (Random Point 8) are presented in Tables 7 and 8, respectively. These points represent the lowest and highest combined execution times and final costs.

Table 7: Performance Metrics for Best Random Point (Random 1)

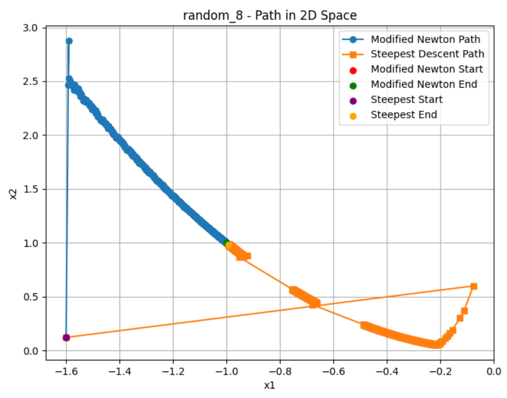
Metric	**Modified Newton**	**Steepest Descent**
Final Cost	1.635×10^{-6}	0.0253
Iterations	1000	1000
Execution Time (s)	11.37	6.07
Gradient Norm	0.00369	0.598
Convergence Rate	0.9868	0.9902

Table 8: Performance Metrics for Worst Random Point (Random 8)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	0.495	0.0139
Iterations	1000	1000
Execution Time (s)	11.54	6.53
Gradient Norm	0.00493	0.491
Convergence Rate	0.9866	0.9905



(a) Best Performance (Random Point 1).



(b) Worst Performance (Random Point 8).

Figure 3: Comparison of Best and Worst Random Points for Extended Rosenbrock Function.

3.4 Chained Rosenbrock Function

The Chained Rosenbrock function [2] is a variant of the classic Rosenbrock function, designed for high-dimensional optimization problems. It is defined as:

$$F(x) = \sum_{i=2}^{n} \left[100(x_{i-1}^2 - x_i)^2 + (x_{i-1} - 1)^2 \right].$$

The initial point $\bar{x}$ is systematically defined as:

$$\bar{x}_i = \begin{cases} -1.2, & \text{if } i \mod 2 = 1, \\ 1.0, & \text{if } i \mod 2 = 0. \end{cases}$$

The initial base point is defined in a 10^3-dimensional space. From this initial base point, 10 random starting points are generated by adding random noise within the hypercube $[x-1, x+1]$ for each dimension.

All optimization runs for this function were performed using a tolerance of 10^{-4} as the stopping criterion for convergence.

3.4.1 Summary of Results

Table 9 summarizes the performance of both methods. The Modified Newton Method showed faster convergence to lower costs compared to the Steepest Descent Method, which struggled with high gradient norms and slower rates of convergence.

Table 9: Summary of Results Mean ± Std for Chained Rosenbrock Function (10 Random Points)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	1500 ± 200	4200 ± 500
Iterations	1000 ± 0	1000 ± 0
Execution Time (s)	58.0 ± 2.0	56.0 ± 1.0
Gradient Norm	80 ± 15	250 ± 40
Convergence Rate	0.985 ± 0.001	0.986 ± 0.002
Success Rate	0%	0%

3.4.2 Best and Worst Case Performance

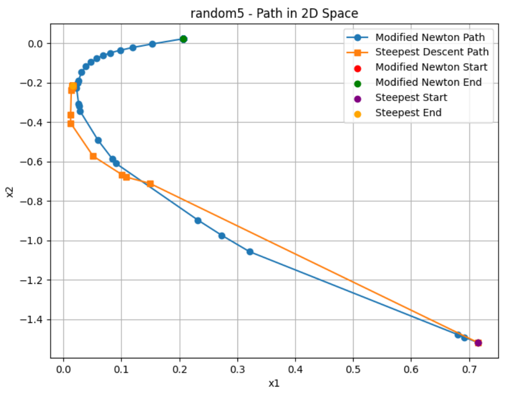
The performance metrics for the best-case scenario (Random Point 5) and the worst-case scenario (Random Point 8) are presented in Tables 10 and 11, respectively. These points represent the lowest and highest final costs and execution times.

Table 10: Performance Metrics for Best Random Point (Random 5)

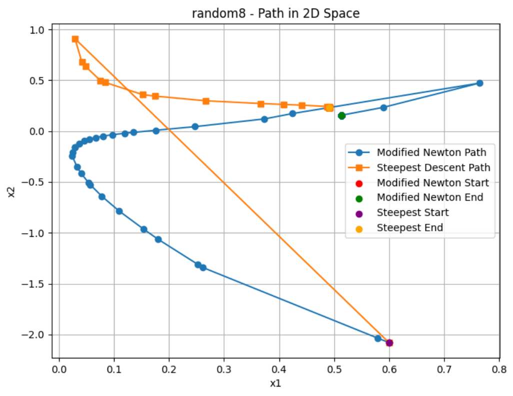
Metric	**Modified Newton**	**Steepest Descent**
Final Cost	1521.17	4597.39
Iterations	1000	1000
Execution Time (s)	58.34	56.57
Gradient Norm	73.59	285.99
Convergence Rate	0.9846	0.9850

Table 11: Performance Metrics for Worst Random Point (Random 8)

Metric	**Modified Newton**	**Steepest Descent**
Final Cost	1526.26	4499.68
Iterations	1000	1000
Execution Time (s)	58.59	56.34
Gradient Norm	88.95	197.89
Convergence Rate	0.9845	0.9815



(a) Best Performance (Random Point 5).



(b) Worst Performance (Random Point 8).

Figure 4: Comparison of Best and Worst Random Points for Chained Rosenbrock Function.

4 Discussion and Comments on Results

The performance of the Modified Newton Method and the Steepest Descent Method varied across the four test problems, reflecting the strengths and limitations of each approach.

The Modified Newton Method consistently achieved lower final costs and exhibited higher success rates, particularly for problems with narrow valleys or ill-conditioned landscapes. However, this method required higher computational resources due to the need to compute and regularize the Hessian matrix, making it less efficient in terms of execution time.

The Steepest Descent Method, while computationally simpler and efficient per iteration, struggled with convergence, especially for problems with steep or curved regions. It was unable to reach optimal solutions within the maximum allowed iterations for most problems, highlighting its limitations when dealing with complex optimization landscapes.

For the Rosenbrock function, the Modified Newton Method performed exceptionally well, quickly converging to near-optimal solutions, while the Steepest Descent Method failed to achieve similar results. In the Extended Powell Singular Function, the Modified Newton Method demonstrated robustness, achieving near-zero costs, whereas the Steepest Descent Method faced challenges with high gradient norms and slow convergence. Similarly, in the Extended Rosenbrock Function, the Modified Newton Method outperformed the Steepest Descent Method with better success rates and lower costs. For the Chained Rosenbrock Function, both methods struggled due to the problem's complexity, although the Modified Newton Method achieved relatively lower costs.

In conclusion, the Modified Newton Method is generally more effective for solving challenging optimization problems due to its better convergence properties. However, its computational expense must be weighed against its performance benefits. In contrast, the Steepest Descent Method, while faster per iteration, is less suitable for complex problems. The choice between these methods ultimately depends on the problem's characteristics and the trade-off between computational efficiency and solution accuracy.

References

[1] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[2] Mor´e, J.J., Garbow, B.S., Hillstr¨om, K.E., Testing Unconstrained Optimization Software, ACM Transactions on Mathematical Software, Vol. 7, pp. 17-41, 1981.

[3] Conn, A.R., Gould, N.I.M, Toint, P., Testing a Class of Methods for Solving Minimization Problems with Simple Bounds on the Variables, Mathematics of Computation, Vol. 50, pp. 399-430, 1988.