Numerical Optimization with Python (2024B)
Programming Assignment 01

Omri Drori 207921719

May 12, 2024

Introduction

In this assignment, we implement line search minimization methods and test them on various objective functions. We visualize their performance and show it here .

1 Project Organization

The project is organized into two main directories: `src` and `tests`. The `src` directory contains the modules `unconstrained_min.py` and `utils.py`. The `tests` directory includes `test_unconstrained_min.py` and `examples.py`.

2 Implementation and Testing

We have implemented various minimization methods including Gradient Descent and Newton's Method with Wolfe conditions for step length search.

2.1 Quadratic Functions

2.1.1 Function 1: Contour Lines are Circles

$$f(x) = x^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x$$

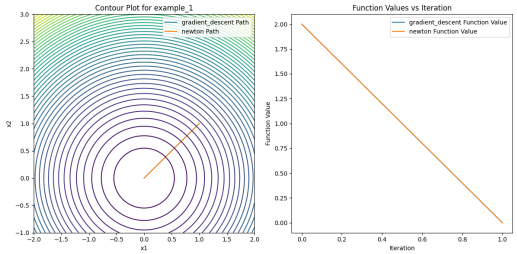


Figure 1: Example 1 Contour Lines are regular Circles

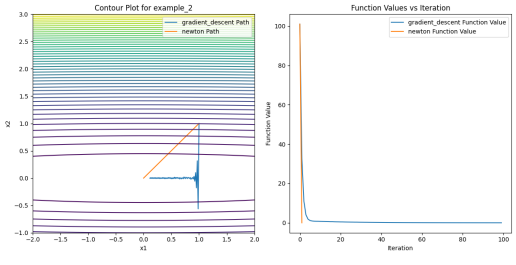
```
Example 1 - Method: gradient_descent, Final x: [0. 0.], Final f: 0.0, Success: True, Iterations: 2
Example 1 - Method: newton, Final x: [0. 0.], Final f: 0.0, Success: True, Iterations: 2
```

Figure 2: details Example 1

Analysis The convergence for this function is both stable and fast due to the isotropic nature of its Hessian matrix, which is the identity matrix. This condition ensures that the gradient is proportional to the displacement from the minimum, allowing both Gradient Descent and Newton's Method to proceed directly towards the global minimum without any directional bias. This ideal scenario showcases the algorithms' efficiency in handling well-conditioned problems.

2.1.2 Function 2: Contour Lines are Axis-Aligned Ellipses

$$f(x) = x^T \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix} x$$



[!h]

Figure 3: Example 2: Contour Lines are Axis-Aligned Ellipses

```
Example 2 - Method: gradient_descent, Final x: [ 0.11450918 -0.00157439], Final f: 0.013360223501345091, Success: False,
 Iterations: 100
Example 2 - Method: newton, Final x: [0. 0.], Final f: 0.0, Success: True, Iterations: 2
```

Figure 4: example 2 details

Analysis For the second function, the Hessian matrix $\begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix}$ shows there's much more curvature in one direction compared to the other. This difference affects how well the optimization algorithms perform. Newton's Method, which uses this Hessian matrix to guide its steps, quickly finds the best path to the minimum because it adjusts for the uneven curvature. In contrast, Gradient Descent does not adjust well; it moves inefficiently and often zigzags slowly towards the minimum, especially in the direction where the curve is less steep. This situation really shows the weaknesses of Gradient Descent when dealing with problems where one direction is much steeper than the other and highlights how powerful Newton's Method can be since it takes the shape of the curve into account.

2.1.3 Function 3: Contour Lines are Rotated Ellipses

$$f(x) = x^T \left(\begin{bmatrix} \frac{\sqrt{3}}{2} & 0.5 \\ -0.5 & \frac{\sqrt{3}}{2} \end{bmatrix}^T \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{2} & 0.5 \\ -0.5 & \frac{\sqrt{3}}{2} \end{bmatrix} \right) x$$

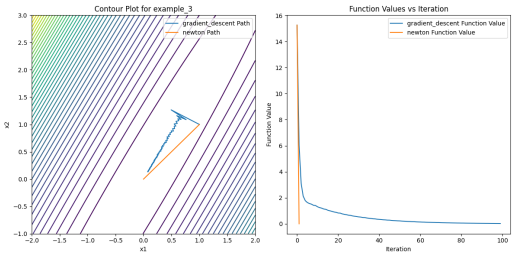


Figure 5: example 3: Contour Lines are Rotated Ellipses

Analysis Looks like the trajectory of both algorithms is rotated with the same rotation as the rotation matrix of the function. It shows rotation transformation doesnt impact the effectivness of the algorithms.

```
Example 3 - Method: gradient_descent, Final x: [0.07508442 0.13440407], Final f: 0.02417132378605058, Success: False,
 Iterations: 100
Example 3 - Method: newton, Final x: [ 1.66533454e-15 -2.22044605e-16], Final f: 2.416668659244244e-28, Success: True,
 Iterations: 2
```

Figure 6: example 3 Details

2.2 Rosenbrock Function

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

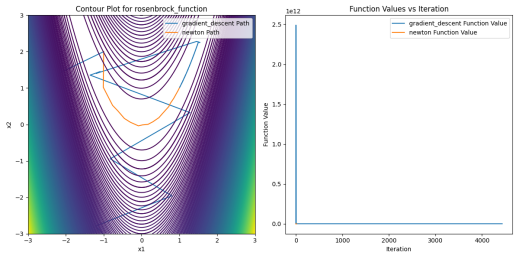


Figure 7: Rosenbrock Function

```
Rosenbrock Function - Method: gradient_descent, Final x: [1.00770212 1.01550939], Final f: 5.95326648477609e-05, Success:
 True, Iterations: 4436
Rosenbrock Function - Method: newton, Final x: [0.9999622  0.99992331], Final f: 1.5495537434004026e-09, Success: True,
Iterations: 21
```

Figure 8: rosenbrock details

Analysis The Rosenbrock function, notorious for its narrow, parabolic valley, poses a significant challenge for Gradient Descent. The first few iterations gradient descent exhibit extremely large values, suggesting that initial steps overshoot significantly, likely due to the steep gradients encountered away from the valley. The coordinates swing from one extreme to another, as seen from (-397, -198) to (349.8388227868109, -197.05939695117638), highlighting instability in the step sizes. Once the trajectory approaches the valley, the convergence becomes painfully slow. The values oscillate and gradually approach the minimum, but the path is fraught with minor improvements and frequent corrections. Once again newton method solves it much faster by taking into account also the curvature not just the steepness.

2.3 Linear Function

$$f(x) = a^T x$$

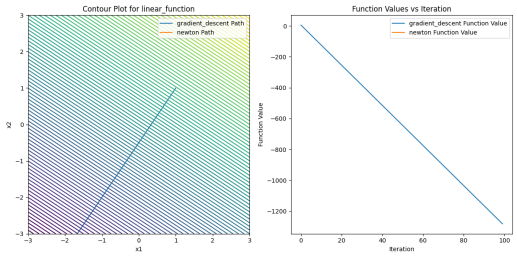


Figure 9: linear function.png

```
Linear Function - Method: gradient_descent, Final x: [-197 -296], Final f: -1282, Success: False, Iterations: 100
Linear Function - Method: newton, Final x: [1 1], Final f: 5, Success: False, Iterations: 1
```

Figure 10: linear function details

Analysis Discussion on the straightforward optimization path due to linearity.

2.4 Boyd's Function

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$

Analysis Complexity introduced by the function's characteristics and its impact on the optimization process.

3 Conclusion

Summarize the findings, the effectiveness of different algorithms, and insights gained from the visualization of algorithm paths.