Optimizing Complex Functions: A Steepest Descent Analysis

Marouane IZMAR, Marco Plazzogna

Professor: Grad Sorin Mihai

Optimisation project 2023/2024

January 2, 2024

Contents

0.1 Introduction

This report focuses on the implementation and analysis of the steepest descent method applied to two well-known test functions in optimization: Rosenbrock's and Himmelblau's functions. The objective is to assess the algorithm's effectiveness in finding the minima of these functions under various starting points and step sizes.

The problem under consideration is the minimization of a real-valued function of two variables. Mathematically, it can be stated as:

$$\inf_{\mathbf{x}\in\mathbb{R}^2} f(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = [x_1, x_2]^T$ is a two-dimensional vector in the real space $\mathbb{R}^2$.

Rosenbrock's function is defined as:

$$f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \tag{2}$$

Himmelblau's function is defined by:

$$g : \mathbb{R}^2 \to \mathbb{R}, \quad g(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{3}$$

0.2 Mathematical Analysis

0.2.1 Rosenbrock's function

The gradient of a function provides the direction of the steepest ascent. For Rosenbrock's function, the gradient is a vector of partial derivatives with respect to x_1 and x_2, given by:

$$\frac{\partial f}{\partial x_1} = -2(1 - x_1) - 400x_1(x_2 - x_1^2) \tag{4}$$

$$\frac{\partial f}{\partial x_2} = 200(x_2 - x_1^2) \tag{5}$$

The Hessian matrix, which is the square matrix of second-order partial derivatives, provides information about the curvature of the function. For Rosenbrock's function, the Hessian is:

$$H = \begin{bmatrix} -400(x_2 - 3x_1^2) + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \tag{6}$$

The global minimum of a function occurs where its gradient is zero. Setting the gradient of Rosenbrock's function to zero, we find:

$$-2(1 - x_1) - 400x_1(x_2 - x_1^2) = 0 \tag{7}$$

$$200(x_2 - x_1^2) = 0 \tag{8}$$

Solving these equations, we find that the global minimum occurs at $(x_1, x_2) = (1, 1)$.

Furthermore, evaluating the Hessian at the global minimum, we get:

$$H_{\min} = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix} \tag{9}$$

The positive definiteness of this Hessian matrix at $(1, 1)$ confirms that this point is indeed a minimum.

0.2.2 Himmelblau's function

The gradient of Himmelblau's function is:

$$\nabla g(x_1, x_2) = \begin{bmatrix} 2(2x_1(x_1^2 + x_2 - 11) + x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11 + 2x_2(x_1 + x_2^2 - 7)) \end{bmatrix} \tag{10}$$

Himmelblau's function has four local minima, all with a function value of zero, located approximately at the following points:

- $(3, 2)$

- $(-2.805118, 3.131312)$
- $(-3.779310, -3.283186)$
- $(3.584428, -1.848126)$

0.3 Steepest Descent Method

The Steepest Descent method, also known as the Gradient Descent method, is an iterative optimization algorithm used to find the local minimum of a function.

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, the Steepest Descent method iteratively updates a candidate solution $\mathbf{x} \in \mathbb{R}^n$ in the direction of the steepest descent, which is the opposite of the gradient of the function at that point. The update rule at each iteration k is given by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) \tag{11}$$

Here, $\nabla f(\mathbf{x}_k)$ is the gradient of f at $\mathbf{x}_k$, and α_k is the step size, which determines the magnitude of the descent step. The step size can be constant or determined adaptively.

The algorithm iterates until a convergence criterion is met, typically when the magnitude of the gradient $\|\nabla f(\mathbf{x}_k)\|$ falls below a specified threshold, indicating that a local minimum has been reached, or when a maximum number of iterations is exceeded.

In this report, we apply the Steepest Descent method to two well-known test functions in optimization: Rosenbrock's and Himmelblau's functions. We analyze the algorithm's performance in terms of convergence to the global minimum and the number of iterations required.

0.4 Results and Analysis

In this section, we present the results obtained from implementing the steepest descent method for minimizing Rosenbrock's and Himmelblau's functions. The analysis focuses on outcomes with different starting points and step sizes, accompanied by a discussion on the algorithm's behavior and performance metrics.

0.4.1 Rosenbrock's Function

- **Starting Point (0, 0):**
  - *Constant Step Size (0.001):* The algorithm reached near the point (0.674, 0.453) after 1000 iterations, suggesting a path towards the minimum but

not fully converging to the global minimum at (1, 1). In fact, if we stay within 5000 iterations, the convergence of the algorithm is (0.956, 0.914), nearer to the already known minimum (1, 1).

  - *Variable Step Size:* The algorithm reached near the point (0.909, 0.826) after 1000 iterations, suggesting a path towards the minimum but not fully converging to the global minimum at (1, 1). In fact, if we stay within 5000 iterations, the convergence of the algorithm is (0.998, 0.995), nearer to the already known minimum (1, 1).

- **Starting Point $(\pi + 1, \pi - 1)$:**

  - *Constant Step Size (0.001):* Failed to converge probably because of the step size. In fact, if we lower the step size (0.0001 instead of 0.001) we have a convergence. Since, the step size is lower, the algorithm may be slower. Therefore, we used 5000 iterations to get (1.519, 2.308) and, in order to see that it converges towards the minimum, with 50000, we get (1.166, 1.361) which is closer to (1, 1).
  - *Variable Step Size:* The algorithm reached near the point (1.888, 3.564) after 1000 iterations, suggesting a path towards the minimum but not fully converging to the global minimum at (1, 1). In fact, if we stay within 5000 iterations, the convergence of the algorithm is (1.577, 2.490), nearer to the already known minimum (1, 1).

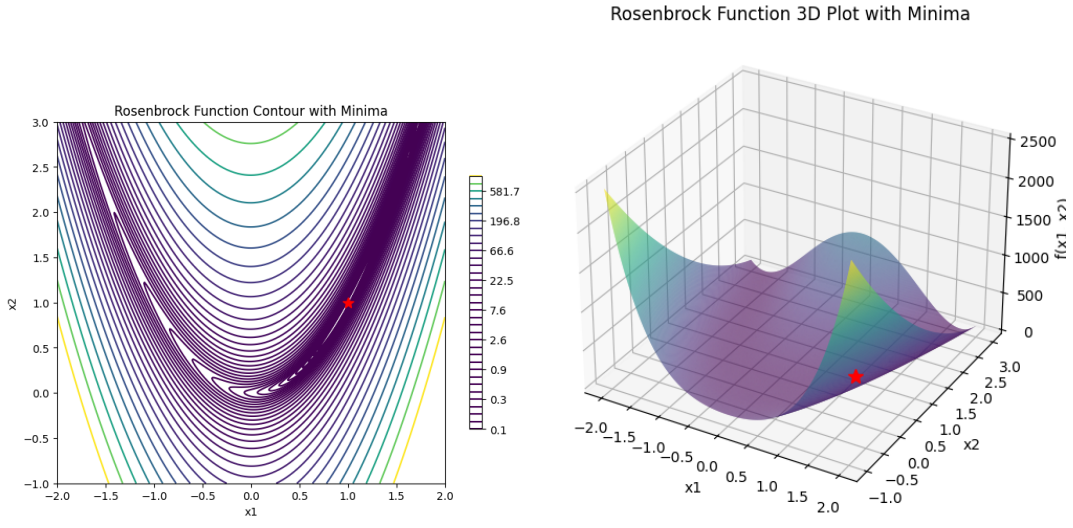

Figure 1: Contour and 3D plot of Rosenbrock's function

0.4.2 Himmelblau's Function

- **Starting Point (0, 0):**

- *Constant Step Size (0.001):* Successfully converged to a local minimum near (3, 2) within 973 iterations.
- *Variable Step Size:* Successfully converged to a local minimum near (3, 2) within 1000 iterations.

- **Starting Point $(\pi + 1, \pi - 1)$:**
  - *Constant Step Size (0.001):* Successfully converged to a local minimum near (3, 2) within 863 iterations.
  - *Variable Step Size:* Successfully converged to a local minimum near (3, 2) within 1000 iterations.

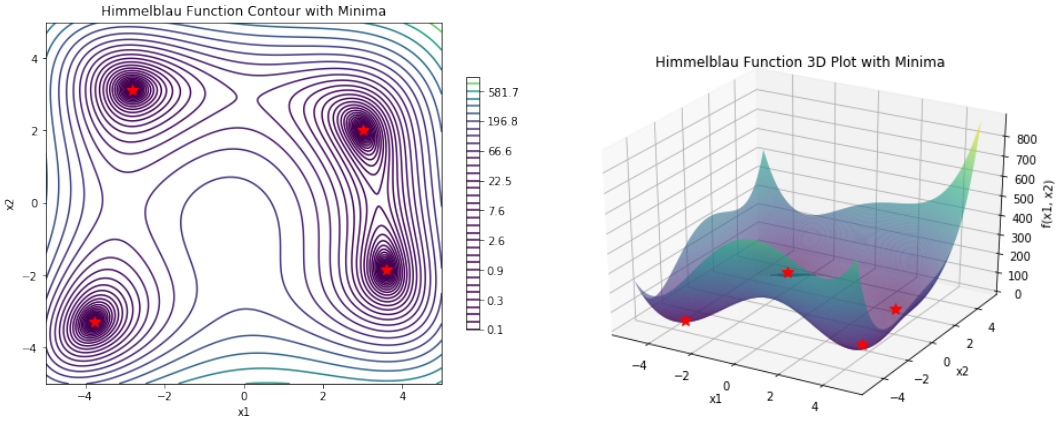

Figure 2: Contour and 3D plot of Himmelblau's function

The following table summarizes the results obtained from the steepest descent method applied to Rosenbrock's and Himmelblau's functions with various starting points and step sizes (we did not include further specific results that were previously discussed singularly):

Function	**Start Pt.**	**Step Size**	**Variable**	**Min. Pt.**	**Iterations**	**CPU Time (s)**
Rosenbrock	(0, 0)	0.001	No	(0.674, 0.453)	1000	0.0214
Rosenbrock	(0, 0)	ARMIJO	Yes	(0.909, 0.826)	1000	0.1377
Rosenbrock	(π+1, π-1)	0.001	No	NaN	1000	0.0163
Rosenbrock	(π+1, π-1)	ARMIJO	Yes	(1.888, 3.564)	1000	0.1582
Himmelblau	(0, 0)	0.001	No	(3.00, 2.00)	973	0.0175
Himmelblau	(0, 0)	ARMIJO	Yes	(3.00, 2.00)	1000	0.0924
Himmelblau	(π+1, π-1)	0.001	No	(3.00, 2.00)	863	0.0163
Himmelblau	(π+1, π-1)	ARMIJO	Yes	(3.00, 2.00)	1000	0.0923

Table 1: Summary of the results with a maximum of 1000 iterations

0.5 Conclusion and Interpretation

0.5.1 Key Observations

- The **Rosenbrock's function**, with its narrow and steep valley, proved to be a challenging landscape for the algorithm. In most cases, the method did not converge to the global minimum at (1,1) within 1000 iterations, but if we use a higher number of iterations, we can see that the algorithm is slowly going in the right direction. This highlights the algorithm's sensitivity to both the choice of starting point and step size, particularly in complex landscapes. We want to focus also on the presence of a problematic context for the fixed step size starting from the point (π+1, π-1) where the algorithm did not converge at first because of the value chosen for the step size; with a slight correction though, we managed to make it work properly.

- The **Himmelblau's function**, is characterised by multiple local minima. We found that with steep descend algorithm, the convergence is always towards the point (3, 2). It may be important to note that choosing a different starting point could lead the algorithm to different local minima.

- The **Variable Step Size** generally did not perform as well as a constant step size in terms of CPU time. This might be attributed to the step size computations that could slow down the overall algorithm. We could underline that, typically, the convergence is faster with a fixed step size.

- The **CPU time** across different tests indicated the algorithm's computational efficiency, although the number of iterations to convergence varied significantly.

0.5.2 Conclusion

In conclusion, the steepest descent method, while straightforward and computationally efficient, has its limitations, particularly in dealing with functions that exhibit steep valleys or multiple local minima. This study underscores the importance of understanding the underlying function landscape and tailoring the optimization approach accordingly.

For further results, like with the ones with a higher number of iterations or with a different fixed step size, we suggest to have a look at the code.