02610 Optimization and Data Fitting: Homework Assignment 1

Adrian Lopez Pirvu (s232101)
Rodrigo Bonmati Salazar (s233334)
DTU Compute
Technical University of Denmark

Contents

List of Figures

List of Tables

Table 1: Contributions of s232101 and s233334 to Each Report Section

Section	s232101's Contribution	s233334's Contribution
Section 1	50%	50%
Section 2	50%	50%
Section 3	50%	50%
Section 4	50%	50%

1 One-page report on the exercises for week 5 on the Rosenbrock function (60%).

$$f(x) = f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{1}$$

1.1 (by hand) Compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ of (1).

$$\nabla f(x_1, x_2) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right] = \left[-400(x_2 - x_1^2)x_1 - 2(1 - x_1), 200(x_2 - x_1^2)\right] \tag{2}$$

$$\nabla^2 f(x_1, x_2) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 1200x_1 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \tag{3}$$

1.2 (by hand) Show that $x^* = [1, 1]^T$ is the only local minimizer of this function.

$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5}$$

As shown from (4), and (5), x^* is the only local minimizer of (1).

1.3 (by hand) Verify that the minimizer satisfies the sufficient optimality conditions.

The sufficient Optimality conditions are shown below:

$$\text{Let } x^* \in \mathbb{R} \text{ satisfy } \frac{df}{dx}(x^*) = 0 \text{ and } \frac{d^2 f}{dx^2}(x^*) > 0.$$

x^* is a local minimizer. While the first condition is confirmed in section 1.2 for (1), we still need to ensure the Hessian matrix is positive definite. This requires calculating the eigenvalues of the Hessian at $x^* = (x_1^*, x_2^*)$.

$$\nabla^2 f(x_1^*, x_2^*) = \begin{bmatrix} 1200x_1^* - 400x_2^* + 2 & -400x_1^* \\ -400x_1^* & 200 \end{bmatrix} = \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix} \tag{6}$$

$$Elginvalues = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1.006 \times 10^3 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7}$$

From (7), it can be seen that (1) meets the Sufficient Optimality conditions.

1.4 Make a contour plot of $f(x)$ in the interval $-1 \leq x_1 \leq 2$ and $-1 \leq x_2 \leq 2$. Locate the minimizer in this plot.

Shown in Figure 1.

1.5 Make a contour plot of $\log(f(x))$. Compare this to your previous contour plot. Do they give the same information?

$$\log(f(x)) = \log(f(x_1, x_2)) \tag{8}$$

The analysis revealed that both the contour plot of the original function Figure 1 and its logarithm Figure 2 provided similar insights. Yet, the logarithmic transformation in Figure 2 offered enhanced visualization by compressing the range of function values, revealing patterns more clearly than in Figure 1. In essence, using a logarithmic scale enhanced our understanding of the data, especially for functions with a broad range.

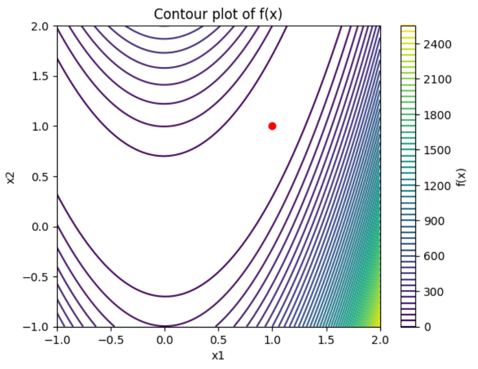

Figure 1: Contour plot of Rosenbrock function, $f(x)$.

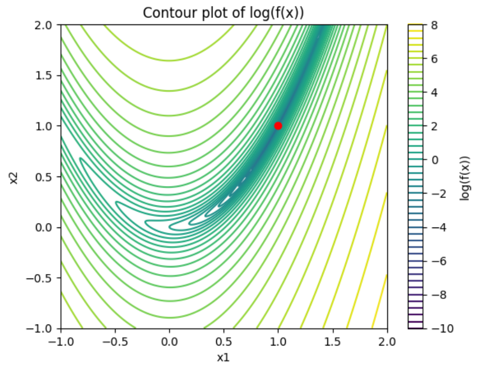

Figure 2: Contour plot of the logarithm of the Rosenbrock function, $\log(f(x))$.

1.6 Write a Matlab/Python function `rosenbrock` to return the function value $f(x)$, its gradient $\nabla f(x)$, and Hessian $\nabla^2 f(x)$.

The code used to create a `rosenbrock` function in python , its gradient $\nabla f(x)$, and Hessian $\nabla^2 f(x)$ can be found in the Appendix, section A.1.

1.7

1.7.1

Initially, we experimented with various α values, ranging from large (see Figure 10 in the Appendix) to small (see Figure 3), determining 0.01 as the optimal α. Using steepest descent with $\alpha = 0.01$ resulted in a final error of approximately 0.05 (refer to Figure 4). Notably, the algorithm did not converge to an optimum, irrespective of the starting point.

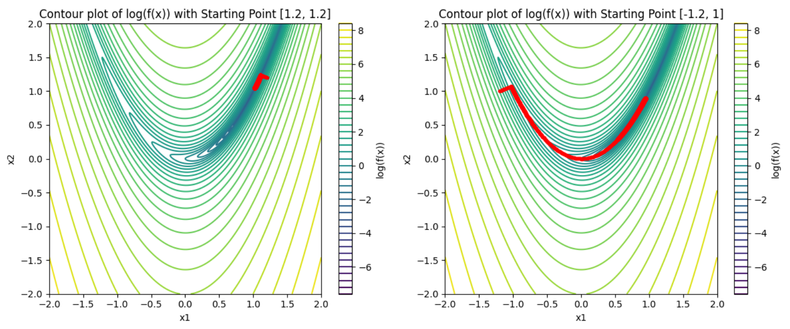

Figure 3: Optimum α for Steepest descent method.

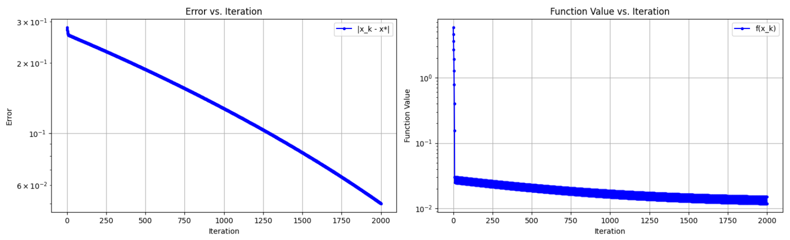

Figure 4: Error for optimum α for Steepest descent method, for $x_0 = [1.2, 1.2]^T$.

1.7.2

α changes shown in Figure 5.

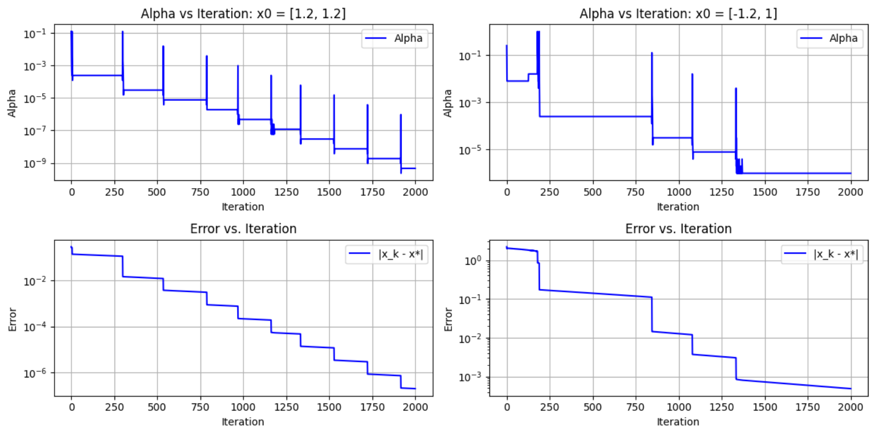

Figure 5: α changes with iterations, for $x_0 = [1.2, 1.2]^T$ and $x_0 = [-1.2, 1]^T$.

1.7.3

For initial points $[1.2, 1.2]^T$ and $[-1.2, 1]^T$, Newton's method converges after five and seven iterations respectively, with the latter necessitating additional iterations due to its increased initial distance from the minimum. Despite a consistent decline in function values $f(x_k)$ with each iteration, an overshoot in the second iteration indicates a temporarily excessive step length.

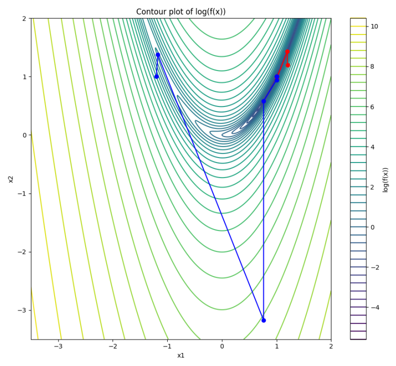

Figure 6: Iterations for Newton's method convergence for $x_0 = [1.2, 1.2]^T$, $[-1.2, 1]^T$.

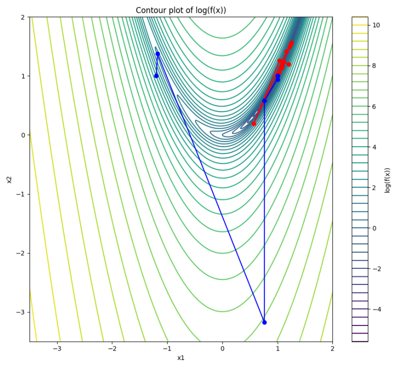

Figure 7: Iterations for BFGS' method convergence for $x_0 = [1.2, 1.2]^T$, $[-1.2, 1]^T$.

Section A.4 confirms an increase in $f(x)$ after the second iteration (Figure 14), with a concurrent steady error reduction in Newton's method (Figure 13). This decrease may be due to iterations nearing the minimum, even while transitioning to a different contour curve (Figure 6).

1.7.4

BFGS iteration values of $f(x_k)$ for each x_k are depicted in Figure 7. Refer to the Appendix for visualizations of the error term, e_k, and function values, $f(x_k)$, displayed in Figure 12 and Figure 11, respectively.

1.7.5

Figures illustrating the error, e_k, and $f(x_k)$ for Steepest Descent (incorporating line search), Newton's Method (utilizing a step size of 1), and the BFGS Method (with line search) are presented in the Appendix as Figure 13 and Figure 14, respectively.

2 Convexity (by hand, 10%)

1. $f(x)$ is convex if its domain, S, is a convex set and satisfies:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2) \tag{9}$$

2. Let $f(x)$ be twice differentiable. Then, $f(x)$ is convex if its domain, S, is a convex set and its Hessian, $\nabla^2 f(x_1, x_2)$, is positive semidefinite:

$$\nabla^2 f(x_1, x_2) \succeq 0 \tag{10}$$

$$f(x) = x_1 x_2, \quad \nabla f(x) = \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}, \quad \nabla^2 f(x_1, x_2) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} < \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{12}$$

As shown in (12), the eigenvalues of $\nabla^2 f(x)$ are < 0. Hence, $\nabla^2 f(x)$ is not positive semidefinite, which means that $f(x)$ is not convex.

3 Stationary points and steepest descent method (20%)

3.1 Compute the gradient $\nabla f(x)$ and Hessian $\nabla^2 f(x)$ by hand.

$$f_1(x) = x_1 + x_2^2 - 1, \quad f_2(x) = x_1 + 3, \quad f(x) = f_1(x)^2 + f_2(x)^2 \tag{13}$$

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 2x_2^2 + 4 \\ 4(x_2^2 + x_1 - 1)x_2 \end{bmatrix}, \quad \nabla^2 f(x_1, x_2) = \begin{bmatrix} 4 & 4x_2 \\ 4x_2 & 12x_2^2 + 4x_1 - 4 \end{bmatrix} \tag{14}$$

3.2 Find the three stationary points

The sufficient Optimality conditions are shown below:

$$\text{Let } x^* \in \mathbb{R} \text{ satisfy } \frac{df}{dx}(x^*) = 0 \text{ and } \frac{d^2 f}{dx^2}(x^*) > 0.$$

The following are the stationary points of $f(x)$:

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 2x_2^2 + 4 \\ 4(x_2^2 + x_1 - 1)x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{15}$$

$$x_1^* = \begin{bmatrix} -3 \\ 2 \end{bmatrix}, \quad x_2^* = \begin{bmatrix} -3 \\ -2 \end{bmatrix}, \quad x_3^* = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \tag{16}$$

To check the nature of each stationary point, it is necessary to evaluate the eigevalues of $f(x)$'s Hessian:

$$Eigenvalues(x_1^*) = \nabla^2 f(x_1^*) = \begin{bmatrix} 4 & 8 \\ 8 & 32 \end{bmatrix} \therefore \lambda_1 = \begin{bmatrix} 34.124 \\ 1.875 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{17}$$

$$Eigenvalues(x_2^*) = \nabla^2 f(x_2^*) = \begin{bmatrix} 4 & -8 \\ -8 & 32 \end{bmatrix} \therefore \lambda_1 = \begin{bmatrix} 34.124 \\ 1.875 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{18}$$

$$Eigenvalues(x_3^*) = \nabla^2 f(x_3^*) = \begin{bmatrix} 4 & 0 \\ 0 & -8 \end{bmatrix} \therefore \lambda_1 = \begin{bmatrix} 4 \\ -8 \end{bmatrix} < \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{19}$$

From (17), (18), and (19) it can be seen that x_1^* and x_2^* are minimizers, given that their Hessians are positive definite. However, for x_3^*, we have that the Hessian is not positive definite, so this means that x_3^* is a saddle point.

To check whether x_1^* and x_2 are global minimizers it is necessary to see their respective values of $f(x)$ shown below:

$$f(x_1^*) = 0, \quad f(x_2^*) = 0 \tag{20}$$

From Equation (20), it is evident that both minimizers, x_1^* and x_2^*, have the same value for $f(x)$, signifying that both are global minimizers.

3.3 Use Python to apply the steepest descent method with backtracking line search on the problem of minimizing $f(x)$.

Table 2: Convergence Data for Steepest Descent method with Line Search

Initial Points (x_0)	Stationary Point (x^*)	Number of Iterations
$[-2, 1]^T$	$[-3, 2]^T$	51
$[-2, -1]^T$	$[-3, -2]^T$	51
$[-2, 0]^T$	$[-1, 0]^T$	1

In Table 2, the convergence data for each stationary point (x_1, x_2, and x_3^*) is presented, detailing both the final convergence point and the corresponding iteration count. It is notable that achieving convergence to the global minimizers requires 51 iterations. In contrast, convergence to the saddle point is achieved in a mere single iteration, attributable to the proximity of the initial point to this stationary position. This proximity effect is shown further in Figure 8.

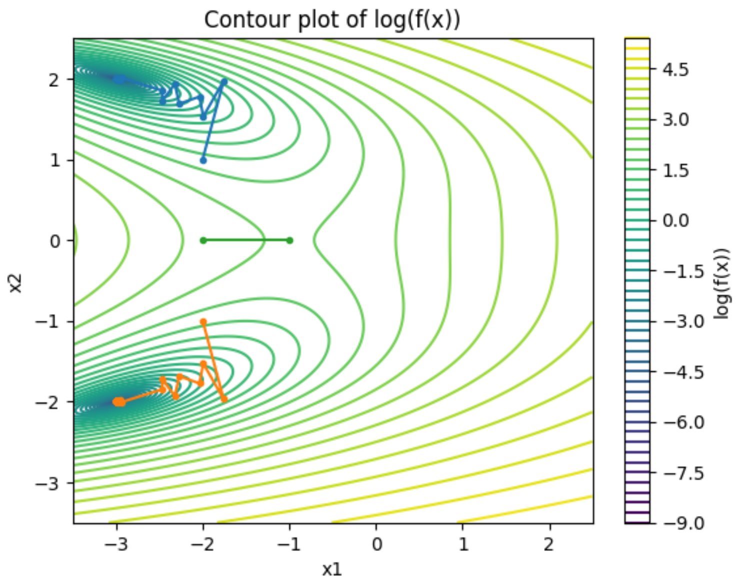

Figure 8: Steepest gradient with line search

4 Newton method (by hand, 10%)

$$f(x) = x^S \text{ with } S \geq 2 \text{ and } x \in \mathbb{R}_{++} \text{ i.e. } x > 0 \tag{21}$$

From Table 3, it can be seen that for $S \geq 2$ and $x \in \mathbb{R}_{++}$, the Hessian, $\nabla^2 f(x)$, is always positive.

$$\nabla f(x) = nx^{n-1} = 0 \text{ at } x = 0 \text{ for } x \in \mathbb{R}_{++} \tag{22}$$

Table 3: Function, Gradient, and Hessian for Various S

S	$f(x)$	$\nabla f(x)$	$\nabla^2 f(x)$
2	x^2	$2x$	2
3	x^3	$3x^2$	$6x$
4	x^4	$4x^3$	$12x^2$
n	x^n	nx^{n-1}	$n(n-1)x^{n-2}$

Furthermore, the Newton's direction is a descent direction, meaning that it gets closer to the global minimum, $x = 0$, as seen in (22).

$$x_{k+1} = x_k + \alpha p_k^N \tag{23}$$

$$p_k^N = -(\nabla^2 f(x_k))^{-1} \nabla f(x) \tag{24}$$

Newton's method, as delineated in (23), necessitates the calculation of the search direction, p_k^N, which is explained in (24). A careful examination of Table 4 reveals that the method exhibits linear convergence at each iterative step.

Table 4: Calculation of p_k^N for Various S

S	p_k^N	Simplified p_k^N
1	$\frac{-2x}{2}$	$-\frac{x}{1}$
2	$\frac{-3x^2}{6x}$	$-\frac{x}{2}$
3	$\frac{-4x^3}{12x^2}$	$-\frac{x}{3}$
n	$\frac{-nx^{n-1}}{n(n-1)x^{n-2}}$	$-\frac{x}{n}$

The observation from Table 4 substantiates the assertion that Newton's method is well-defined, with linear convergence towards zero. Moreover, the convergence factor discerned from Table 4 is $-\frac{1}{n}$, which is applied to the step size, α, throughout all iterations of Newton's method.

A Appendix

A.1 Rosenbrock function in Python

```
def Myfun(x):
    def f(x):
        return 100 * (x[1] - x[0]**2)**2 + (1 - x[0])**2

    def df(x):
        return np.array([-400 * x[0] * (x[1]-x[0]**2) - 2*(1-x[0]), 200 * (x[1] -
            x[0]**2)])

    def d2f(x):
        return np.array([[1200 * x[0]**2 - 400 * x[1] + 2, -400 * x[0]], [-400 * x
            [0], 200]])

    f = f(x)
    df = df(x)
    d2f = d2f(x)

    return f, df, d2f
```

A.2 Rosenbrock function steepest descent method

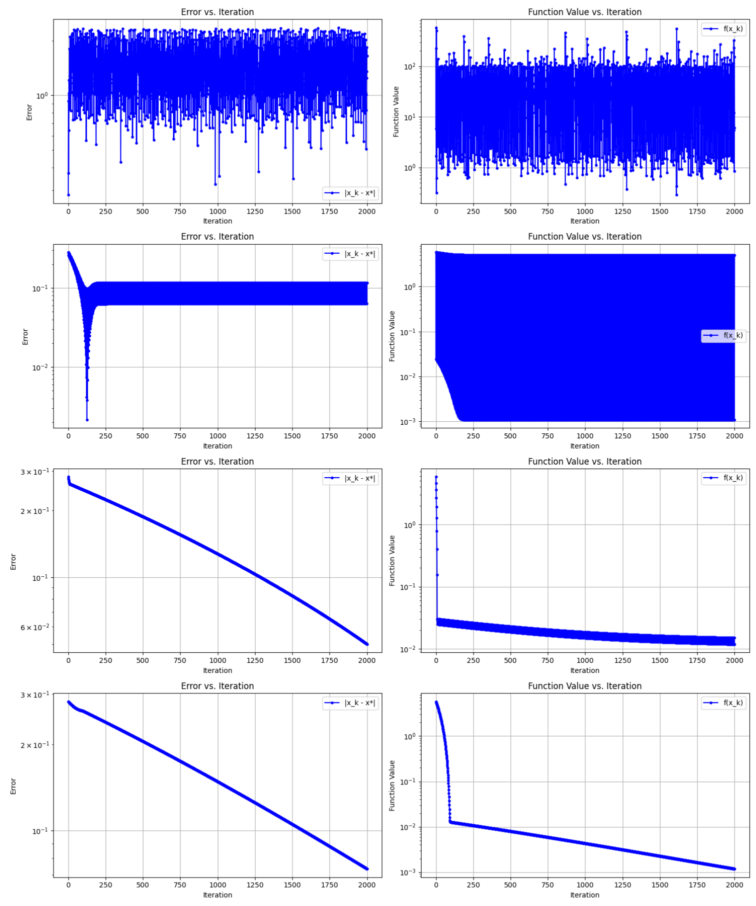

Figure 9: Steepest descent method error with different values of α.

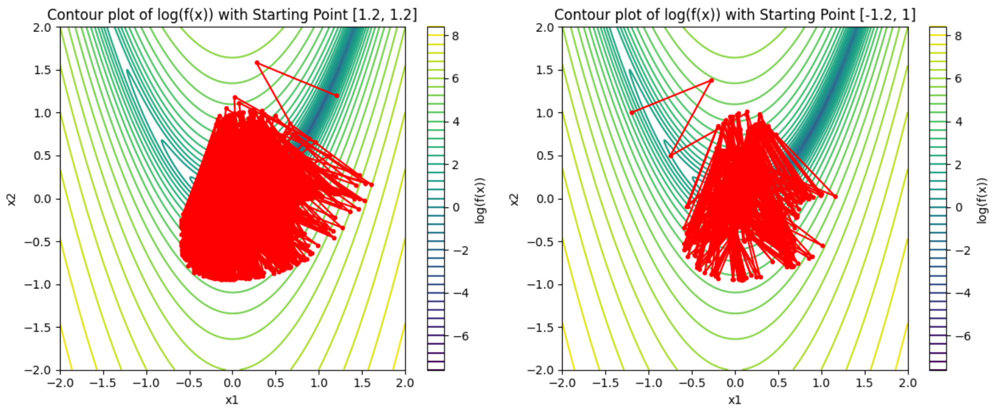

Figure 10: Steepest descent method with different starting points x_0.

A.3 Rosenbrock function BFGS

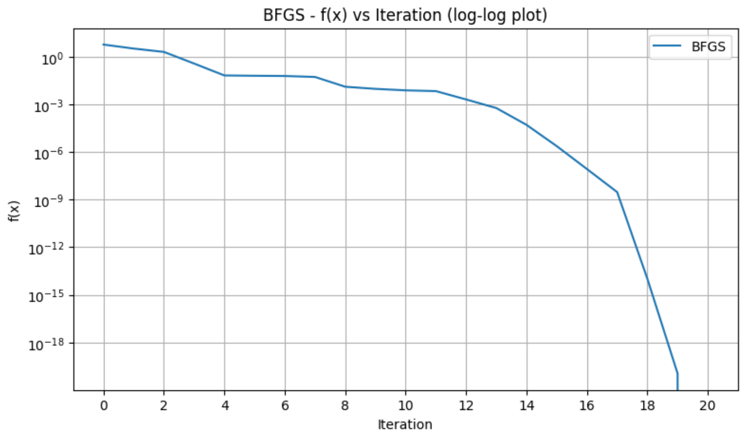

Figure 11: Funtion value with respect to iterations for BFGS method

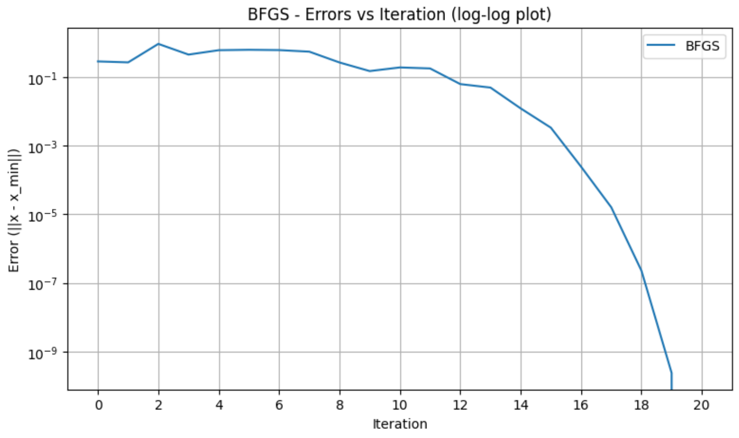

Figure 12: Error with respect to iterations for BFGS method

A.4 Error and $f(x_k)$ for Steepest descent (with line search), Newton's method (with step size 1), and BFGS (with line search)

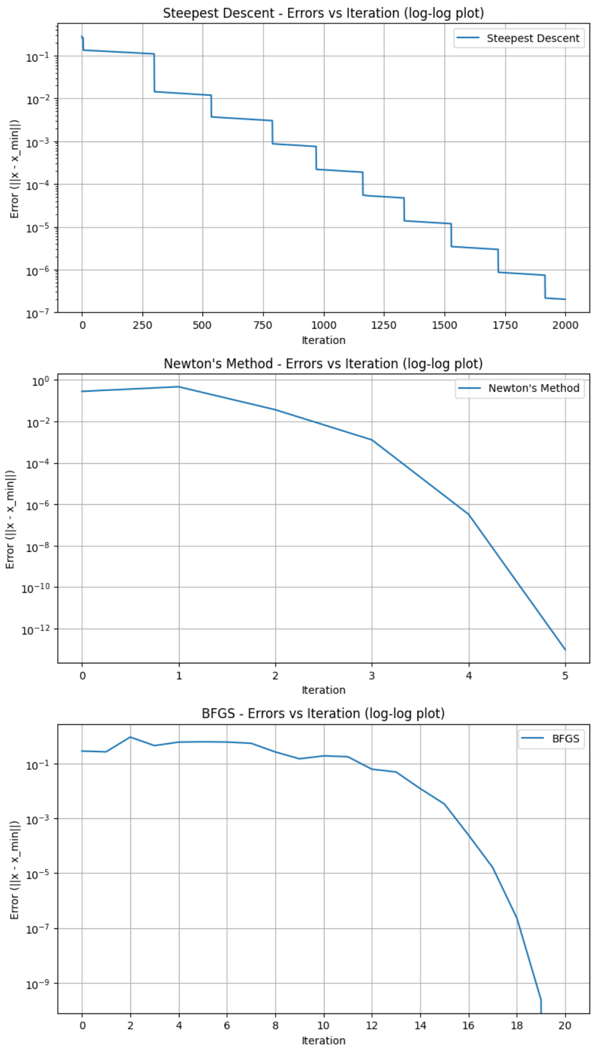

Figure 13: Error for Steepest descent (with line search), Newton's method (with step size 1), and BFGS (with line search)

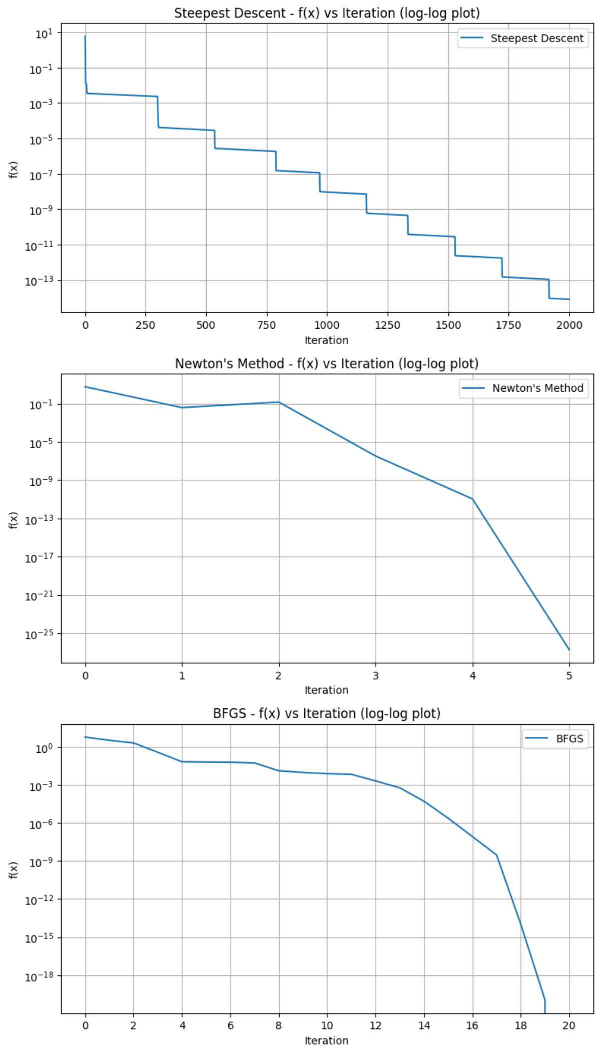

Figure 14: $f(x_k)$ for Steepest descent (with line search), Newton's method (with step size 1), and BFGS (with line search)