Numerical Optimization Assignment 1

Dilnaz-str989, Hamsa-nbp737, Rasmus-kpn134

February 16, 2026

1 Introduction

The following report answers the given theoretical question (described in Section 2), and the minimizer benchmarking (described in Section 3). For the programming part, we have measured the minimizers through two general experiments, which respectively measures execution for a single starting point and across multiple points as further described below.

For the project, all team-members have contributed equally. For benchmarking, the old version of the case_studies.py file was used.

2 Theory

When $d = 1$, and $\alpha^{\frac{0}{0}} = 1$, then $f_1(x) = x^2$, $f_3(x) = \frac{f_1(x)}{\epsilon + f_1(x)}$, i.e. $f_3(x) = \frac{x^2}{\epsilon + x^2}$
We can rewrite the function as

$$f_3(x) = x^2 \cdot (\epsilon + x^2)^{-1}$$

$$\begin{aligned}\bigtriangledown f_3(x) &= 2x(\epsilon + x^2)^{-1} + x^2(-1)(\epsilon + x^2)^{-2}(2x) \\ &= \frac{2x}{\epsilon + x^2} - \frac{2x^3}{(\epsilon + x^2)^2} = \frac{2x\epsilon + 2x^3 - 2x^3}{(\epsilon + x^2)^2} = \frac{2x\epsilon}{(\epsilon + x^2)^2}\end{aligned}$$

To find $\bigtriangledown^2 f_3(x)$, we rewrite the $\bigtriangledown f_3(x) = 2x\epsilon \cdot (\epsilon + x^2)^{-2}$

$$\begin{aligned}\bigtriangledown^2 f_3(x) &= 2\epsilon(\epsilon + x^2)^{-2} + 2x\epsilon(-2)(\epsilon + x^2)^{-3}(2x) \\ &= \frac{2\epsilon}{(\epsilon + x^2)^2} - \frac{8x^2\epsilon}{(\epsilon + x^2)^3} = \frac{2\epsilon(\epsilon + x^2) - 8x^2\epsilon}{(\epsilon + x^2)^3} = \frac{2\epsilon^2 - 6x^2\epsilon}{(\epsilon + x^2)^3}\end{aligned}$$

If Hessian is positive definite, $\bigtriangledown^2 f_3(x) > 0$, i.e. $\frac{2\epsilon^2 - 6x^2\epsilon}{(\epsilon + x^2)^3} > 0 \Leftrightarrow 2\epsilon^2 - 6x^2\epsilon > 0 \Leftrightarrow |x| < \sqrt{\frac{\epsilon}{3}}$
Thus, the transformed ellipsoid f_3, has positive Hessian in the range $|x| < \sqrt{\frac{\epsilon}{3}}$.

3 Programming

3.1 Experiments 1

Objective: To test the speed of convergence of different algorithms and also their robustness to different problems within their class.
Approach: Our main goal of the experiment was to test the limitations of the algorithms by giving them various functions, these being f_1, f_2 and f_3. To remove any irrelevant or noisy factors, we kept every other parameter constant. The maximum number of iterations was set to 1000 and the threshold was set to 10^{-10}. The reason for the chosen threshold is that the limited precision of floating point numbers might prevent the algorithm from reaching the optimum if the chosen threshold is too small. The maximum number of iterations would show the efficiency of the algorithm,

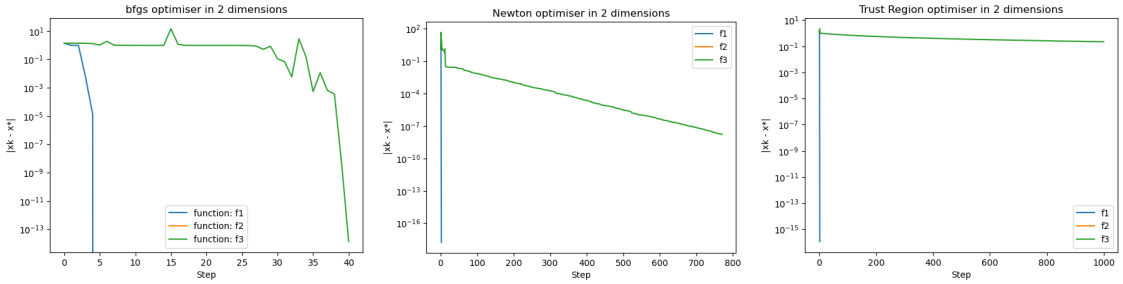
whether it used the resources well enough to get an accurate result. Lastly the starting point for all algorithms is the matrix of ones.

We have selected the metric "distance of x from optimum": $|x_k - x^*|$ as our measure, which effectively illustrates the convergence rate of algorithms. This metric highlights that smaller steps lead to faster algorithmic convergence. By plotting according to this metric, we can effectively visualize the speed of convergence. And the accuracy can be compared by the final distance of x from optimum.

The advantage of this approach is that we can directly compare the algorithms performance, since they are tested on the same set of objective functions and with the same parameters. This experiment can highlight differences in convergence behavior across the algorithms. One of the limitations of this approach is using the same starting point for all algorithms, this may introduce bias.

Plots and results

The results of just running the first three case studies on the different optimizers can be seen in Figure 1. The starting point was set to the constant $(1, 1)$. First thing to notice is that you only see the graphs for function f_1 and f_3, this is because the optimizers only did one iteration for function f_2. So it is only one point in the graph, which is a limitation of using the same starting point $(1, 1)$ for all evaluations. We learned from the theory exercise 1, that the Rosenbrock function has a minimiser at the point $(1, 1)$, so because our starting point is $(1, 1)$, the algorithm terminates after evaluating the point once.



(a) BFGS algorithm (b) Newton algorithm (c) Trust region algorithm

Figure 1: The optimizers applied to the first 3 case study functions

3.2 Experiments 2

Objective: Test the stability of three algorithms under different starting points selection.
Approach: For the parameters "maximum iterations", "tolerance" and dimension of starting points, we fixed their values as 1000, $1 \cdot 10^{-10}$ and 2 respectively. We randomly picked starting points from a uniform distribution in the range $[-10, 10]$. This was done 100 times. Because the chosen dimension is 2, the size of starting points were also 2. After getting these parameters, we apply them in the different algorithms and functions, storing the metric "number of function evaluations". This time, we have 100 values for each combination of algorithms with functions, so we can calculate the variance of these 100 values of each combination, which can show if the algorithm can perform stably when taking different starting points. We made a table in the report to show the variances of evaluation number. By focusing on the variance of the number of function evaluations, the approach quantifies the stability of each algorithm under different starting conditions. High variance indicates instability, while low variance suggests consistent performance across multiple runs.

Table and results

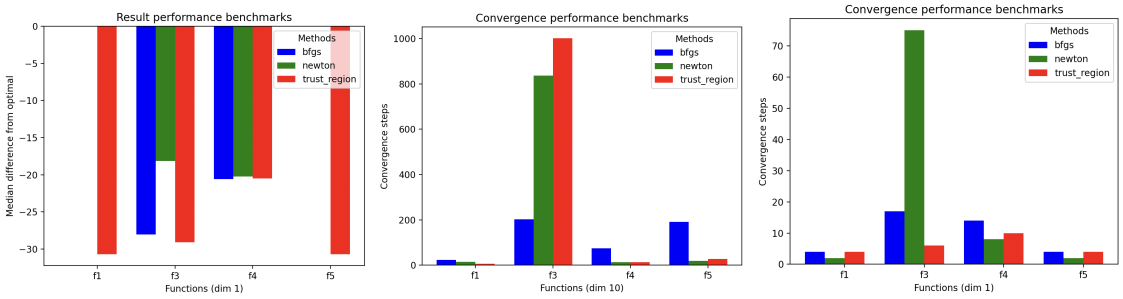
From table 1 we see that, when choosing different starting points for f_2 and f_3, Newton's performance is unstable compared to the optimization of the other three functions, given the high variances of

Variance / Functions / Algorithms	f1	f2	f3	f4	f5	Average
BFGS	2.910	810.766	69.884	74.398	13.948	194.3810
Newton	0.684	132972.526	8800.368	4.930	7.014	28357.1041
Trust region	0.464	87.201	0.000	0.688	4.686	18.6079

Table 1: Results of multiple starting points

132972.526 and 8800.368. From the average column, we see for these functions that the Trust region algorithm performs the best in stability when taking different starting points, since the average variance of convergence is 18.6079 – smallest among the three algorithms.

To also visualize the *general* tendency, we additionally constructed a set of bar plots of the minimizer, across functions and summarized for all points (see Figure 2). We did this for both the "distance" between the found function value vs. the optimum value (in log-scale), and the number of steps before convergence (this time for multiple dimensions, like 1 and 10 as shown below). We included all functions besides f_2, due to its fixed dimensionality requirement. We initially summarized the tendency across points through their mean, but after discovering huge performance outliers, we transitioned to using the median instead to capture the general trend. As seen, the distances generally approach zero for all minimizers across all functions (given by the downward facing bars on log-scale), which suggests that outliers are infrequent. In terms of convergence steps, we generally see best performance for Newton's method and Trust regions for f_1, f_4, and f_5 but worse performance at f_3 compared to BFGS. This tendency changes for Trust regions however, when dimensions are increased (e.g. from 1 to 10 as shown).



(a) distance from opt. ($dim = 1$) (b) Steps ($dim = 1$) (c) Steps ($dim = 10$)

Figure 2: Bar plot summary of minimizer performance.

4 Discussion & Conclusion

Based on the plots of Experiment 1 for f_3 (visualized in log-scale), there is an indication that the BFGS algorithm exhibits super-linear convergence, while Newton's is linear and Trust region either linear or supralinear. To determine this exactly however, one would need to analyze the algorithms (or in any case perform more runs of the case studies).

In terms of algorithm accuracy, all methods tend to approximate the optimal value, with the exception of some huge outliers. In terms of convergence steps, the BFGS method (which seems to be superlinear, at least for f_3) would be theoretically preferable. In practice however the algorithms perform very differently when subject to different case functions. Choosing the right algorithm will therefore depend on the problem at hand. In terms of convergence in higher dimensions, Trust region generally seems to perform best for our benchmarking strategy (while also exhibiting the lowest variance in terms of outliers). Aside from this, BFGS seems to be the most stable in terms of general trend, as seen on Figure Sub-figure 2b and 2c.