Numerical Optimization
Re-exam Handin 4

Dmitry Serykh (qwl888)

June 22, 2020

1 The Setup

I implemented the Trust Region optimizer by following the basic algorithm structure, outlined Algorithm 4.1 and then solved the Sub-problem 4.7 (finding the value of p), such that the conditions from the Theorem 4.1 are satisfied (4.8a, 4.8b and 4.8c).

1.1 Solution to the subproblem

I decided to implement and experiment with both methods. I validated the implementation by checking if all three conditions hold for the found value of λ^*. I determined experimentally that the first approach exhibits better performance and the minima of the case function are found in fewer iterations, hence it was chosen.

1.1.1 Algorithm 4.3

The first method, that I tried was from Section 4.3, and Algorithm 4.3 in particular. It is based on root-finding Newton's Method and Cholesky factorization that is used to iteratively find the values of λ.

I have added a safeguard, such that the Cholesky factorization always exists. I set the initial value of $\lambda^0 = -\lambda_1^e + c$, where c is a small constant and λ_1^e is the smallest eigenvalue of B. This way, I make sure that $\lambda^{(\ell)} > -\lambda_1$, and hence all eigenvalues of $(B + \lambda^\ell I)$ would be positive and indefinite Hessians would not cause any problems.

The book does not specify any stopping criterion for the Algorithm 4.3. I decided to use a treshold parameter, such that the iteration would terminate when $\|\lambda^\ell - \lambda^{\ell+1}\| < \varepsilon$, where ε is some small constant.dditionally, I have added a shortcut that is described on page 85 in the book. If B is positive-definite and $\|B^{-1}g\| \leq \Delta$, the procedure can be terminated immediately with $\lambda^* = 0$. It is then easy to solve for p^*.

1.1.2 Bijection

I start with the same value of λ_0 as in the previous method. Then I iteratively find the value of λ_1, I start by increasing λ by $2^0 \cdot |\lambda_0$, then by $2^1 \cdot |\lambda_0$ and so on. This continues until I find a value of λ_1, s.t. $\|p(\lambda_1)\| < \Delta$. Afterwards, I execute the algorithm as specified in the assignment text. The algorithm details and the reasoning behind some of the desertions are described in Section 3. Finally, the procedure is terminated when the interval size $(\lambda_0 - \lambda_1)$ reaches some small constant ε.

1.2 The hard case

I have implemented the method, described on p.88, in order to handle the hard case. λ would equal $-\lambda_1$, and I can find p by first solving for τ:

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j + z \sqrt{\Delta - \sum_{j:\lambda_j \neq \lambda_1} \frac{\left(q_j^T g\right)^2}{\left(\lambda_j + \lambda\right)^2}}$$

However, I did not get to test my solution, since I could not detect the hard case happening in any of the case functions.

1.3 Parameters

I used following parameter values in my implementation. Some values were taken from the literature, while others were determined empirically.

- Initial trust region radius ($\Delta_0 = 1$)
- Maximum trust region radius $\tilde{\Delta} = 10^3$
- Lower bound for the actual/predicted reduction ratio $\eta = 0.2$
- Initial coefficient $\lambda^0 = -\lambda_1^e + 0.001$
- Tolerance $\varepsilon = 10^{-7}$ on the gradient magnitude
- Tolerance $\varepsilon_\lambda = 10^{-7}$ for the both the iterative solutions to the trust region sub-problem.

1.4 Calculation of ρ

The calculation of ρ is an important part of the algorithm, and it is calculated by:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

However, I have experienced a problem with minimization of some functions. The step was getting so small, that the denominator of the fraction above was disappearing. Therefore, I have added a small constant (close to machine precision) to the denominator to prevent this from happening.

2 Testing protocol

In order to test the effectiveness of my implementation, I came up with a testing protocol, where I used following metrics:

- The convergence plots with the number of iteration on the x-scale and the Euclidean distance between the current value of x and the optimum. The resulting plot can be seen on Figure 1.
- The convergence plots with the gradient magnitude. The resulting plot can be seen on Figure 3.
- Plot for the median trust region as a function of number of iterations for the Log-Ellipsoid function can be seen on Figure 2. Trust region is monotonically non-increasing throughout the execution of the algorithm for the Log-Ellipsoid function. I have also experimented with other functions, and for the Rosenbrock function, there are regions for which the radius is increasing. However, the general trend is for the radius to be decreasing throughout the execution of the algorithms. Intuitively, this also makes sense, since we are taking smaller steps when approaching the optimum, because the curvature is less pronounced there.
- **Accuracy**. The Euclidean distance to the optimum at the termination point. The results can be seen on Table 1. The performance of my implementation of the trust region algorithm is compared to the performance of the line search methods from the previous assignment, namely Steepest Descent and Newton's Algorithm.
- **Efficiency**. The number of steps until the gradient magnitude reaches 10^{-7}. The results can be seen on Table 2.

I used a random starting point taken from the uniform distribution in the interval between -10 to 10 and repeated each optimization 100 times for all metrics and took the median. This was done to remove the influence of the starting position from the results of the optimization. I used the median to minimize the effect of the outliers.

As suggested, I have modified my implementation of the Log-Epsilon function and excluded the first Attractive Region function from the experiments.

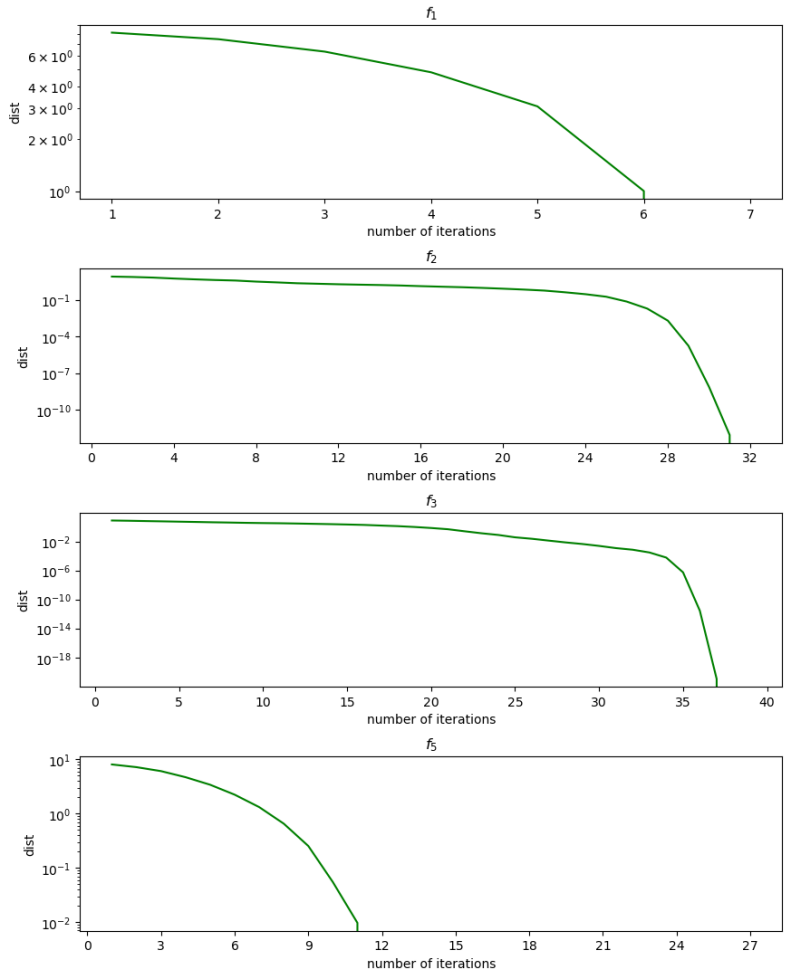

Figure 1: Convergence plot of the median euclidean distance on the y-axis

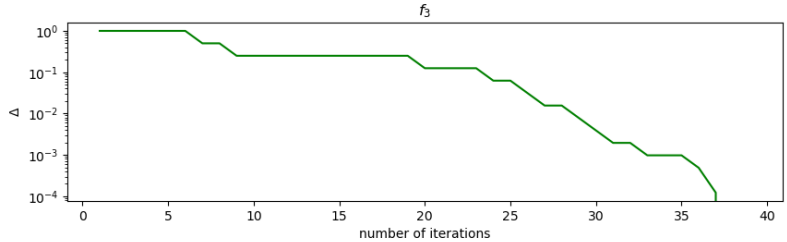

Figure 2: Plot of the median trust region radius as a function of number of iterations for the Log-Ellipsoid function

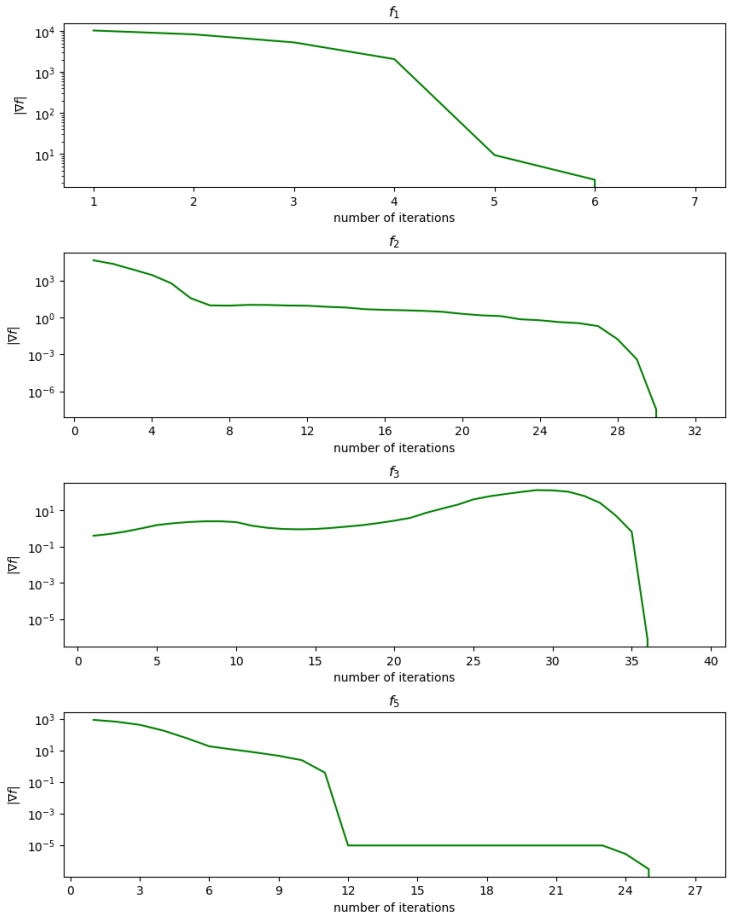

Figure 3: Convergence plot of the median gradient magnitude as a function of number of iterations

	f_1	f_2	f_3	f_4	f_5
stepest descent	$1.44 \cdot 10^{-4}$	$7.59 \cdot 10^{-4}$	$1.44 \cdot 10^{-10}$	$6.51 \cdot 10^{-7}$	$1.13 \cdot 10^{-4}$
newton	0.0	$8.15 \cdot 10^{-8}$	$4.59 \cdot 10^{-16}$	$6.4 \cdot 10^{-7}$	$4.56 \cdot 10^{-7}$
trust region	0.0	$1.05 \cdot 10^{-11}$	$1.72 \cdot 10^{-16}$	n/a	$2.67 \cdot 10^{-7}$

Table 1: Average value of distance to the optimum at algorithm termination for 100 random starting points in the interval $[-10, 10]$

	f_1	f_2	f_3	f_4	f_5
stepest descent	3392	6991	5546	488	52
newton	1	30	16	29	2
trust region	5	32	35	n/a	24

Table 2: Average number of iterations until algorithm termination for 100 random starting points in the interval $[-10, 10]$

3 Theoretical Part

In the theoretical part, I choose the second option.

Problem 4.7 from the Book states:

$$\min_{p \in \mathrm{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad \text{s.t. } \|p\| \leq \Delta$$

We assume that B is strictly positive definite. I will show that the described algorithm converges to λ^*, s.t $p(\lambda^*) = p^*$

We know that $\|p(\lambda)\|$ is continuous and strictly nonincreasing function in the interval $(-\lambda_1^e, \infty)$, where λ_1^e is the smallest eigenvector of B (p.85 in the book). We then set $\lambda_0 = -\lambda_1^e$, since $\lim_{\lambda \to -\lambda_1^e} \|p(\lambda)\| = \infty$ and find a value of λ_1, s.t. $\|p(\lambda_1)\| < \Delta$, which would always exist, since we do not consider the hard case. Hence for any $\lambda_a, \lambda_b, \lambda_c \in (-\lambda_1^e, \infty)$ s.t. $\lambda_a \leq \lambda_b \leq \lambda_c$, we have that:

$$\lambda_a \leq \lambda_b \leq \lambda_c \Leftrightarrow \|p(\lambda_a)\| \leq \|p(\lambda_b)\| \leq \|p(\lambda_c)\|$$

Let λ'_n be the value of λ' at bisection iteration $n \in \{0, 1, 2, ...\}$, then:

$$|\lambda'_n - \lambda^*| \leq \left(\frac{1}{2}\right)^n (\lambda_0 + \lambda_1)$$

and since:

$$\lim_{n \to \infty} \left(\frac{1}{2}\right)^n = 0$$

we have that:

$$\lim_{n \to \infty} |\lambda'_n - \lambda^*| = 0 \Leftrightarrow \lim_{n \to \infty} \lambda'_n = \lambda^*$$

Hence, the algorithm would converge to $p(\lambda^*) = p^*$. Additionally, the tolerance of this algorithm could be adjusted, since we have an upper bound on the error and can use it to get the maximum number of iterations.

Regarding the indefinite case, I solve it by choosing the $\lambda^0 = -\lambda_1^e + c$, where c is a small constant and λ_1^e is the smallest eigenvalue of B. This way, all the eigenvalues would stay positive.

4 Conclusion

I have implemented the Trust Region algorithm, while experimenting with two iterative solutions to the trust region sub-problem. Algorithm manages to find the value of optimum of functions f_1, f_2, f_3 and f_5 and its performance is much better than the Steepest Descent method, but worse than the Newtons method that I have implemented in Assignment 3.