Numerical Optimization
Re-exam Handin 4

Dmitry Serykh (qwl888)

June 14, 2020

1 The Setup

I implemented the Trust Region algorithm following the basic algorithm structure, outlined Algorithm 4.1. In order to solve the Sub-problem 4.7, I used the method from Section 4.3, and Algorithm 4.3 in particular, where the root-finding Newton's Method and Cholesky factorization is used to iteratively find the values of constant λ, and consequently the solution to the Sub-problem 4.7 (value of p), such that the conditions from the Theorem 4.1 are satisfied.

$$\begin{aligned} (B + \lambda I)p &= -g \\ \lambda\left(\Delta - \|p\|\right) &= 0 \\ (B + \lambda I) &\text{ is positive semi-definite} \end{aligned}$$

1.1 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** $\eta = 0.2$
- **Tolerance** $\varepsilon = 10^{-7}$ **on the gradient magnitude**

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 trust region radius. The resulting plot can be seen on Figure 2.
- **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 average. This was done to remove the influence of the starting position from the results of the optimization. I used the mean 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.

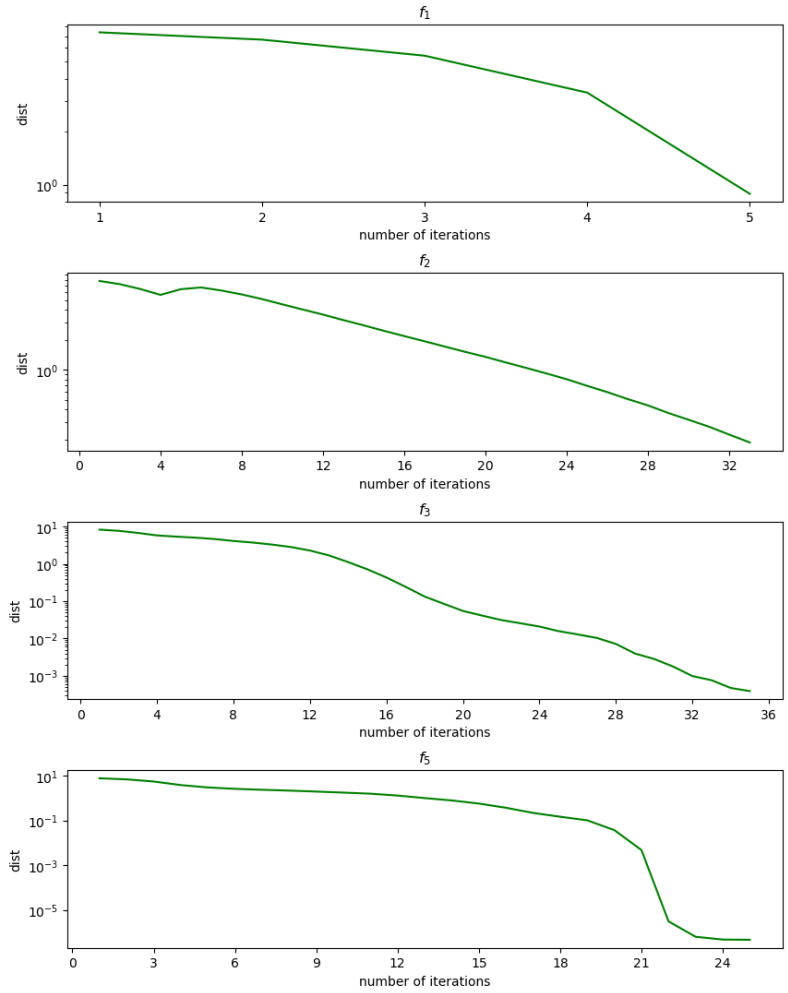

Figure 1: Convergence plot with euclidean distance on the y-axis(log scale)

	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	$6.11 \cdot 10^{-18}$	3.17^{-15}	$1.08 \cdot 10^{-12}$	n/a	$1.70 \cdot 10^{-7}$

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 $[-10, 10]$

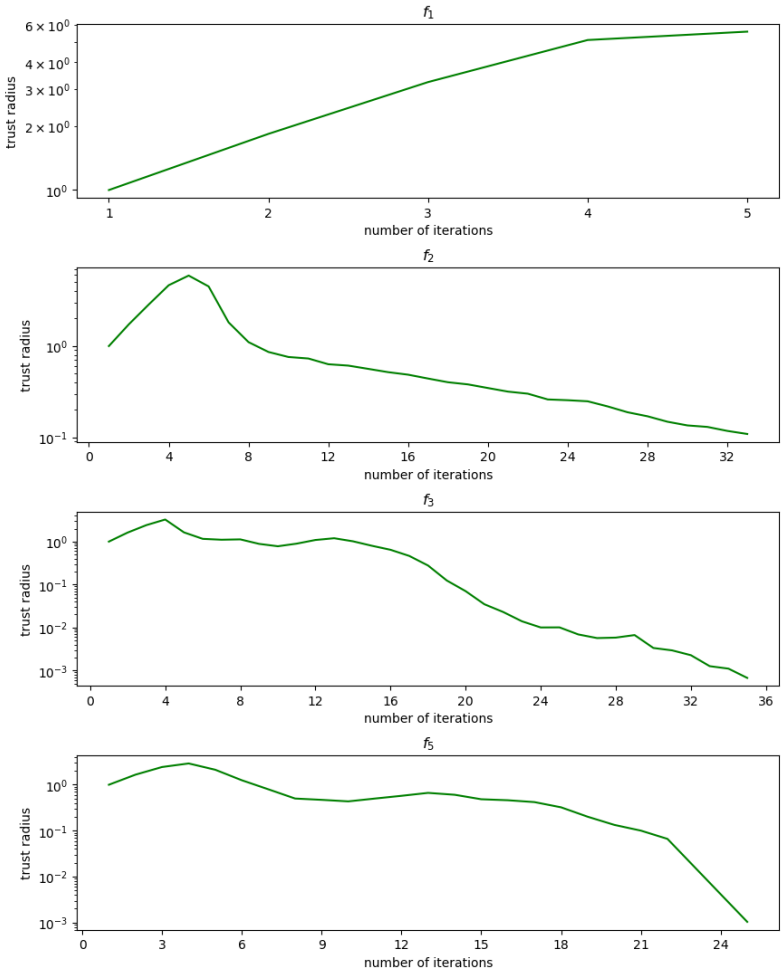

Figure 2: Plot of the trust region radius as a function of number of iterations (log scale)

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 (λ_0, λ_1). (p.85 in the Book). Then, since $\lambda_0 < \lambda' < \lambda_1$, $\|p(\lambda_0)\| > \Delta$ and $\|p(\lambda_0)\| < \Delta$, it would hold that $\|p(\lambda_1)\| < \|p(\lambda')\| < \|p(\lambda_0)\|$.

Let $\mu(\lambda') = \|p(\lambda') - p(\lambda^*)\|$, then we can express the value of p as a function of n, where n is the number of algorithm iterations:

$$p_{\lambda'}(n) \leq p(\lambda *) \pm \left(\frac{1}{2}\right)^n \mu(\lambda')$$

Since:

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

we have that:

$$\lim_{n \to \infty} p_{\lambda'}(n) = p(\lambda^*) = p^*$$

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.

For the indefinite case, we can first make the matrix B positive definite, using one of the methods, suggested in Chapter 3.

4 Conclusion

I have implemented the Trust Region algorithm, while solving sub-problem 4.7. 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 Seepest Descent method, but slower than the Newtons method from the last assignment.