Numerical Optimization - Hand-In 4

Isabela Blucher

March 2, 2018

Exercise 4.10

Since B is symmetric, there exists a diagonal matrix Λ, where $\Lambda = diag\{\lambda_1, \lambda_2, ..., \lambda_n\}$ and $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are eigenvalues of B and an orthogonal matrix Q such that $B = Q\Lambda Q^T$. Consider the following two cases:

1) If $\lambda_1 > 0$, then all eigenvalues of B are positive, and so B is positive definite. In this case, $B + \lambda I$ is positive definite for $\lambda = 0$.

2) If $\lambda_1 \leq 0$ we can choose $\lambda = -\lambda_1 + \varepsilon > 0$ for any fixed real value of ε. Since λ_1 is the most negative eigenvalue of B, $\lambda_i + \lambda \geq \varepsilon > 0$ holds for all $i = 1, 2, ..., n$. If we note that $B + \lambda I = Q(\Lambda + \varepsilon I)Q^T$, we can see that $0 < \lambda_1 + \varepsilon \leq \lambda_2 + \varepsilon \leq ... \leq \lambda_n + \varepsilon$ are the eigenvalues of $B + \lambda I$. So, for this choice of λ, $B + \lambda I$ is positive definite.

Programming Assignment

Exercise 4.2 states "Write a program that implements the dogleg method. Choose B_k to be the exact Hessian. Apply it to solve Rosenbrock's function (2.22). Experiment with the update rule for the trust region by changing the constants in Algorithm 4.1, or by designing your own rules.".

For the starting value of Δ, since it has to be a value inside the $[0, \hat{\Delta}]$ interval, the fixed relation $\Delta_0 = \frac{2}{3}\hat{\Delta}$ was chosen. The reason for this is, we want a value of Δ that is closer to the end of the interval so that the algorithm doesn't start with really small and inefficient steps. But we also don't want too big a value of Δ, as it may take some extra computation for the value o x_k to converge.

Table 1 illustrates the performance of our trust-region algorithm implemented with the dogleg method. For several experiences with different values of the constants $\hat{\Delta}$, η and for $x_0 = (-1.2, 1)^T$ the following results were found:

$\hat{\Delta}$ \ η	0	10^{-5}	1/5
0.1	646	646	646
1	25	25	25
10	30	30	30

Table 1: Number of iterations until convergence for $x_0 = (-1.2, 1)^T$ and different constant values

Table 1 may seem strange, but changing the value of η did not affect the algorithm's performance in any way. After analyzing the computed values of ρ_k it seems that the ratio is almost always close to 1, which indicates a good agreement between the model and the actual function over the iterations, which is why, the value of η seems to have little influence on the number of iteration until convergence.

Exercise: Make convergence plots and experimentally verify that your dog-leg method has the expected convergence rate.

For the following convergence plots, the parameters of the algorithm were set as fixed values and what changed between the two plots were the starting points x_0. The convergence was measured in terms of the euclidean norm of the difference between the current value of x_k and the local minimizer x^*, which is known for the Rosenbrock function. The plots show in the x axis the number of iterations and on the y axis the log of $\|x_k - x^*\|$ and for each line is a different value of $\hat{\Delta}$. The values of the parameters for both plots were $\eta = 10^{-4}$, tolerance used in the stopping criteria for the norm of the gradient = 10^{-8} and maximum number of iterations = 10^4.

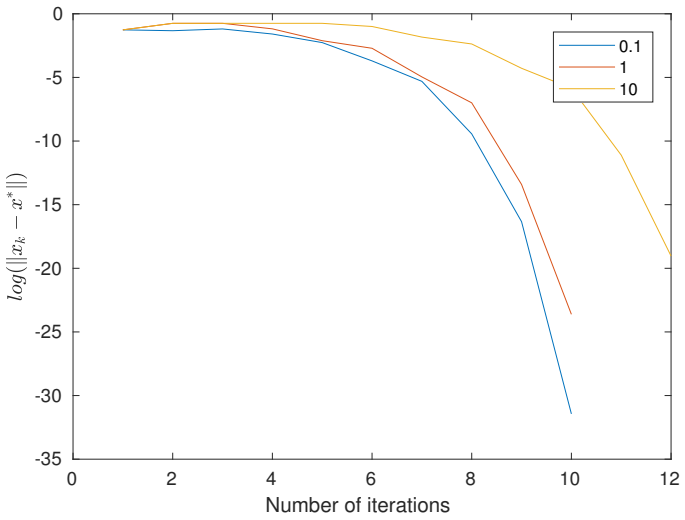
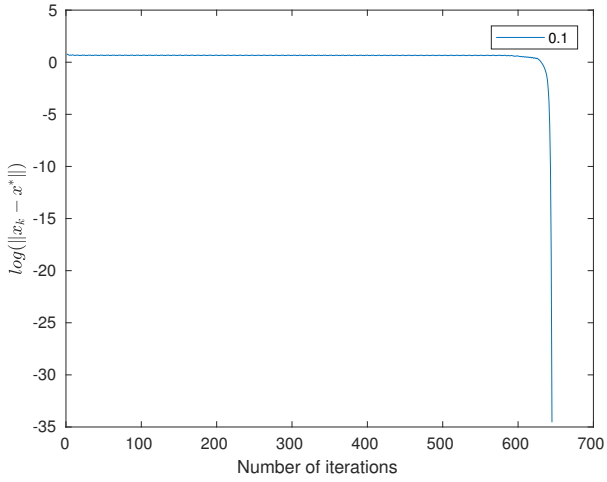
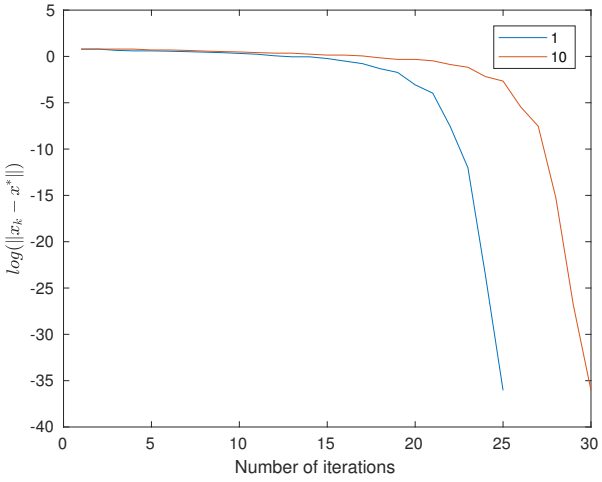


Figure 1: Log plot of the convergence rates for $x_0 = (1.2, 1.2)^T$ for different values of $\hat{\Delta}$

For the next plots, both have the same starting point $x_0 = (-1.2, 1)^T$, but since the number of iterations until convergence for the case where $\hat{\Delta} = 0.1$ is much larger than for the other two cases, displaying them separately seemed better.



(a) $\Delta \hat{=} 0.1$



(b) $\Delta \hat{=} 1$ and $\Delta \hat{=} 10$

Figure 2: Log plot of the convergence rates for $x_0 = (-1.2, 1)^T$ for different values of $\hat{\Delta}$

From visual inspection of Figures 1 and 2 and continuous experimentation with the code, it is possible to affirm that the implemented trust region algorithm with the dog-leg method converges quadratically as expected.

Exercise: Make a plot of how Algorithm 4.1 changes the trust region radius as a function of iterations. Consider if the plot makes sense? Does your implementation of Algorithm 4.1 work?

The following plot shows how the trust region radius Δ_k changes over the iterations of the trust region algorithm. The parameters were fixed with the same values for the last plots.

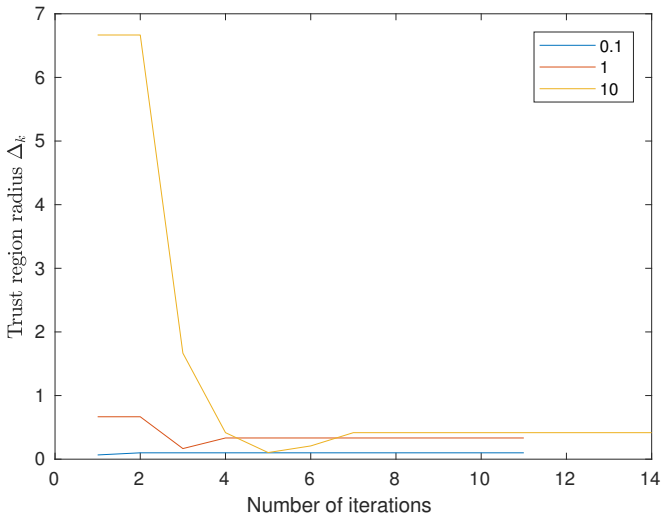


Figure 3: Trust region radius Δ_k over number of iterations for different values of $\hat{\Delta}$

By analyzing the plot we can see that for small values of $\hat{\Delta}$, the trust region radius tends to only go up, which is the case for $\hat{\Delta} = 0.1$. For $\hat{\Delta} = 10$, the radius decreases rapidly and for $\hat{\Delta} = 1$, it stabilizes pretty quickly in the same magnitude as the starting Δ_0. It makes sense as in the algorithm behaves in accordance to Algorithm 4.1 in the book. Given that and the fact that for all tests on the Rosenbrock function, the algorithm found the local minimizer $(1, 1)^T$, it seems that my implementation works.

Exercise: Consider how to select parameter values for Algorithm 4.1 and how sensitive your results are to their settings.

The parameters for Algorithm 4.1 are η, $\hat{\Delta}$ and Δ_0. As it was mentioned before, my implementation of the algorithm is not very sensitive to changes in the value of η, but the number of iterations does change for different values of $\hat{\Delta}$.

As for how to select parameter values, it is important to consider not too big of an η since it would make it much harder for the algorithm to take good steps along a descent direction p_k computed by the dog-leg method. On the other hand, for $\hat{\Delta}$, too small a value would make convergence more difficult, since it represents an overall bound on the step lengths, which is why tests with larger values for the trust region bound converged faster.

Exercise: Of all methods you have worked with so far, which one is the "best"?

Both line-search and trust region algorithms performed well on finding the minimizer for the Rosenbrock function. While Steepest Descent was a lot slower, by testing the algorithms multiple times with different parameter values and starting points, Newton's method and the trust region implemented with the dog-leg method had very similar perfomances, and both converged quadratically. It is hard to unanimously select a method as the best, as I'm sure both can be the best for different applications.