

Numerical Optimization

Re-exam Handin 3

Dmitry Serykh (qwl888)

June 14, 2020

1 The Setup

I implemented the steepest descent and Newton's algorithms using the backtracking line search in order to find the step length (Algorithm 3.1).

1.1 Hessian Modification for the Newton's Method

The problem with Newton's method is that for non-convex functions, the Newton's step $p_k^N = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$ may not be in a descent direction and Hessian could be non-invertible. I have modified the Hessian using the Cholesky with Added Multiple of Identity method (Algorithm 3.3). A value of τI is added to the Hessian. If the Cholesky factorization of the resulting matrix succeeds, we are done. Otherwise, a new value of τ is calculated.

1.2 Parameters

After some testing, I have determined that for my implementation, different parameters must be used for the steepest descent and Newton's methods. Following parameter values were used:

- **Initial step length.** For the Newton's method, I have used the natural value of ($\bar{\alpha} = 1$). For the steepest descent, I empirically found the value of $\bar{\alpha} = 0.1$ to perform best
- **Factor ρ .** For the steepest descent, I used the value of $\rho = 0.2$. For the Newton's method, I have used the value of $\rho = 0.5$.
- **Constant c .** For both methods, I have used to value of $c = 10^{-5}$.
- **Stopping criteria.** For both methods, I have used $\varepsilon = \|\nabla f\| = 10^{-7}$. Furthermore, I have terminated the algorithm at 20000 iterations. However this was only relevant to the steepest descent.
- **Value of β .** For the Hessian Modification (Algorithm 3.3), I have used a value of $\beta = 10^{-3}$.

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 log of the Euclidean distance between the current value of x and the optimum. The results can be seen on Figure 1. I applied the low shelf to the results, so anything would stop at 10^{-6} .
- The convergence plots with the gradient magnitude. The results can be seen on Figure 2.
- **Efficiency.** The number of steps until the gradient magnitude reaches 10^{-7} . The results can be seen on Table 2.
- **Accuracy.** The Euclidean distance to the optimum at the termination point. The results can be seen on Table 1.

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 both 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, which is important due to the stochastic nature of the algorithm.

	f_1	f_2	f_3	f_4	f_5
steepest descent	$1.437 \cdot 10^{-4}$	$7.59 \cdot 10^{-4}$	$1.44 \cdot 10^{-10}$	$6.51 \cdot 10^{-7}$	$1.13 \cdot 10^{-4}$
newton	0.0	$8.15 \cdot 10^{-8}$	$4.59 \cdot 10^{-16}$	$6.4 \cdot 10^{-7}$	$4.56 \cdot 10^{-7}$

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
steepest descent	3392	6991	5546	488	52
newton	1	30	16	29	2

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

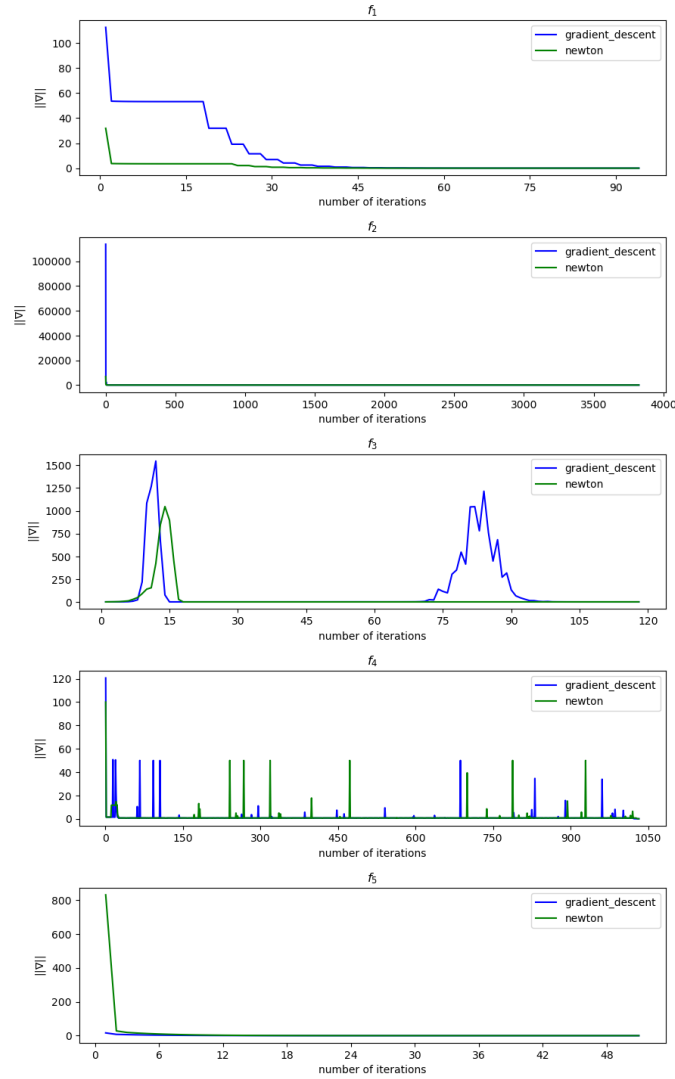


Figure 1: Convergence plot with norm of the gradient on y-axis

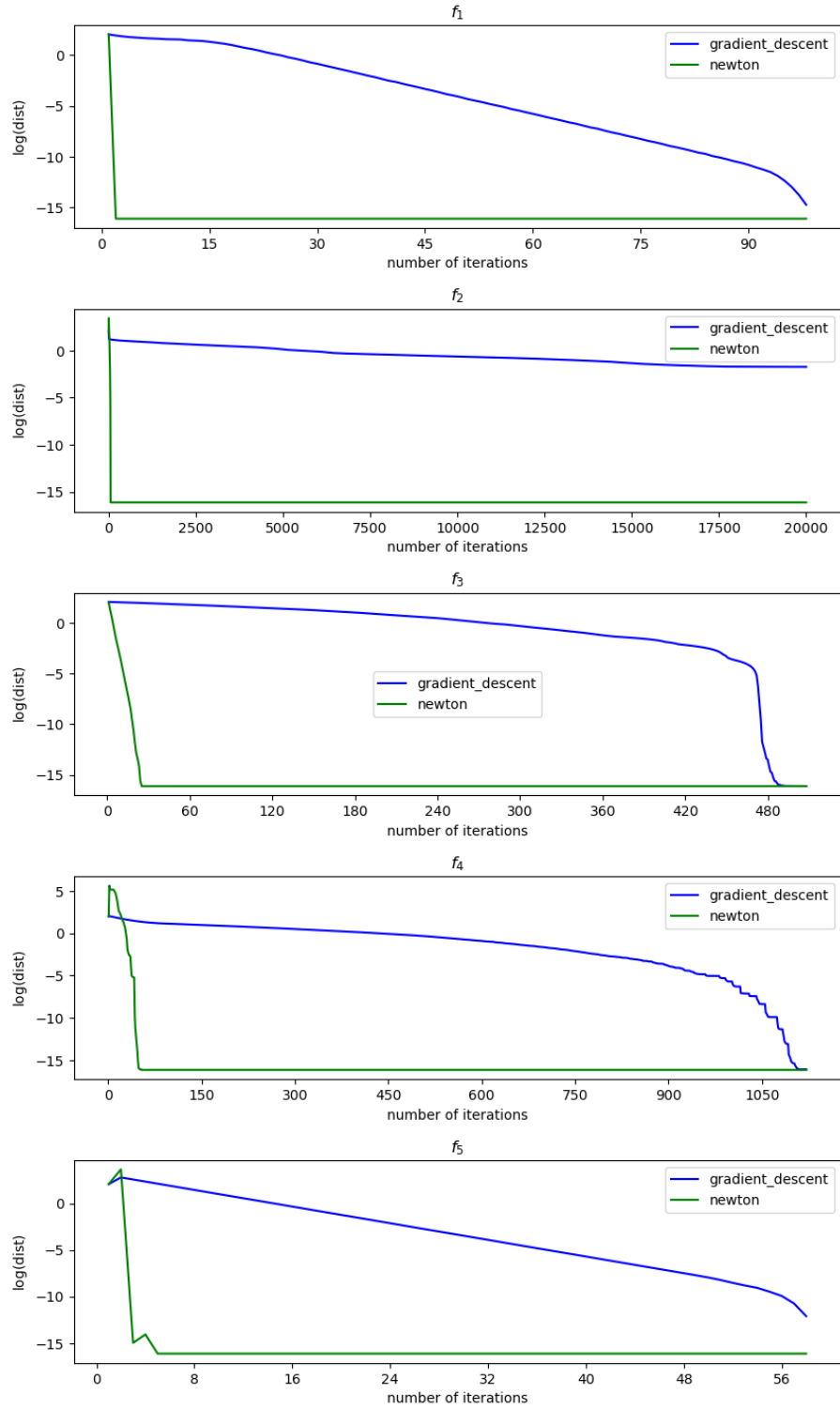


Figure 2: Convergence plot with log of euclidean distance on the y-axis

3 Theoretical Part

3.1 Exercise 3.7

Let us start by looking at the expression:

$$\begin{aligned}\nabla f(x_k) &= Q(x_k - x^*) \\ Q^{-1}\nabla f(x_k) &= (x_k - x^*)\end{aligned}$$

$f(x_k)^T = (x_k - x^*)^T Q^T = (x_k - x^*)^T Q$, since Q is symmetric, then:

$$\begin{aligned}\nabla f(x_k)^T Q^{-1}\nabla f(x_k) &= (x_k - x^*)^T Q(x_k - x^*) \\ \|x_k - x^*\|_Q^2 &= \nabla f(x_k)^T Q^{-1}\nabla f(x_k)\end{aligned}$$

This would be useful later.

By the definition of the step direction in the Steepest Descent method, and subtracting the x^* on both sides, we have:

$$x_{k+1} - x^* = x_k - x^* - \alpha \nabla f(x_k)$$

Then, by the definition of the Q -weighted norm, we have:

$$\begin{aligned}\|x_{k+1} - x^*\|_Q^2 &= (x_k - x^* - \alpha \nabla f(x_k))^T Q(x_k - x^* - \alpha \nabla f(x_k)) \\ &= (x_k - x^*)^T Q(x_k - x^*) - 2\alpha \nabla f(x_k)^T Q(x_k - x^*) + \alpha^2 \nabla f(x_k)^T Q \nabla f(x_k) \\ &= \|x_k - x^*\|_Q^2 - 2\alpha \nabla f(x_k)^T Q(x_k - x^*) + \alpha^2 \nabla f(x_k)^T Q \nabla f(x_k)\end{aligned}$$

We know that $\nabla f(x_k) = Q(x_k - x^*)$, and we have the exact solution for the step length for the quadratic function that is given by: $\alpha = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T Q \nabla f(x_k)}$, therefore:

$$\begin{aligned}\|x_{k+1} - x^*\|_Q^2 &= \|x_k - x^*\|_Q^2 - 2\alpha \nabla f(x_k)^T \nabla f(x_k) + \alpha^2 \nabla f(x_k)^T Q \nabla f(x_k) \\ &= \|x_k - x^*\|_Q^2 - 2 \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T Q \nabla f(x_k)} \nabla f(x_k)^T \nabla f(x_k) + \left(\frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T Q \nabla f(x_k)} \right)^2 \nabla f(x_k)^T Q \nabla f(x_k) \\ &= \|x_k - x^*\|_Q^2 - \frac{2 (\nabla f(x_k)^T \nabla f(x_k))^2}{\nabla f(x_k)^T Q \nabla f(x_k)} + \frac{(\nabla f(x_k)^T \nabla f(x_k))^2}{\nabla f(x_k)^T Q \nabla f(x_k)} \\ &= \|x_k - x^*\|_Q^2 - \frac{(\nabla f(x_k)^T \nabla f(x_k))^2}{\nabla f(x_k)^T Q \nabla f(x_k)} \\ &= \|x_k - x^*\|_Q^2 \left[1 - \frac{(\nabla f(x_k)^T \nabla f(x_k))^2}{(\nabla f(x_k)^T Q \nabla f(x_k)) \|x_k - x^*\|_Q^2} \right]\end{aligned}$$

And since $\|x_k - x^*\|_Q^2 = \nabla f(x_k)^T Q^{-1} \nabla f(x_k)$:

$$\|x_{k+1} - x^*\|_Q^2 = \|x_k - x^*\|_Q^2 \left[1 - \frac{(\nabla f(x_k)^T \nabla f(x_k))^2}{(\nabla f(x_k)^T Q \nabla f(x_k)) (\nabla f(x_k)^T Q^{-1} \nabla f(x_k))} \right]$$

3.2 Application to the Ellipsoid Function

Let $g(k) = \frac{\|f_{k+1}(x) - f(x^*)\|_Q^2}{\|f_k(x) - f(x^*)\|_Q^2}$. Then, by Eq. 3.27 and 3.29:

$$g(k) \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right)^2$$

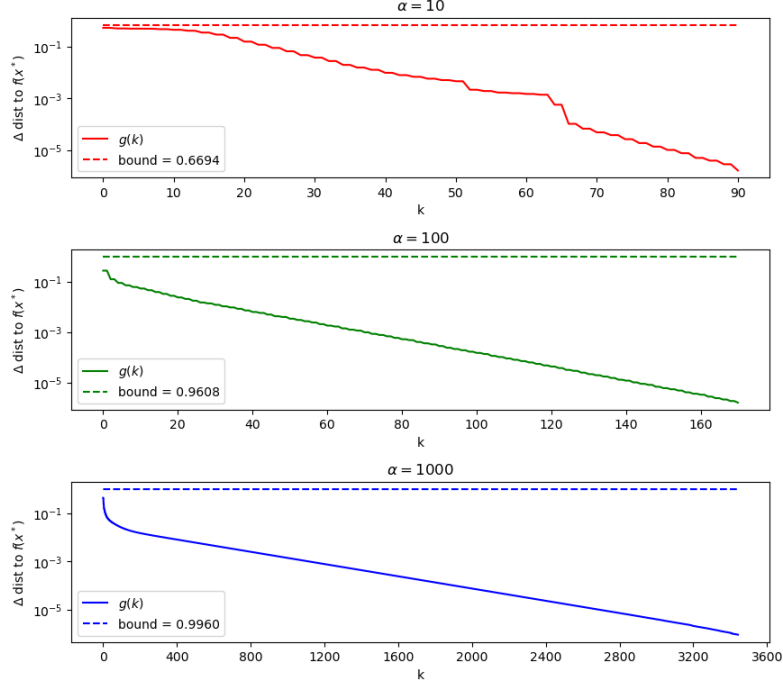


Figure 3: Plot of $g(k)$ and bound 3.29 for the Ellipsoid function with $d = 5$ and $\kappa(Q) = \alpha \in \{10, 100, 1000\}$

where $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ are the eigenvalues of Q . The Ellipsoid function is given by:

$$f_1(x) = \sum_{i=1}^d \alpha^{\frac{i-1}{d-1}} x_i^2$$

In matrix form, it can be represented as $f_1(x) = \mathbf{x}^T \mathbf{Q} \mathbf{x}$, where \mathbf{Q} is a diagonal matrix with values: $[1, \alpha^{\frac{1}{4}}, \alpha^{\frac{2}{4}}, \alpha^{\frac{3}{4}}, \alpha]$ for $d = 5$. Therefore $\lambda_1 = 1$ and $\lambda_n = \alpha$, and bound would equal:

$$g(k) \leq \left(\frac{\alpha - 1}{\alpha + 1} \right)^2$$

I validate this bound empirically for different values of conditioning, $\kappa(Q) = \frac{\lambda_n}{\lambda_1} = \alpha$. The plot of the results can be seen on Figure 3. It can be seen that the bound holds for all values of α .

4 Conclusion

I have implemented the steepest descent and Newton's method minimizers using the backtracking line search (Algorithm 3.1) to find the step size. The convergence plots, that I have provided, demonstrate that the Newton's method outperforms steepest descent for all case functions, and there is an asymptotic difference between the convergence rates of the Steepest Descent and Newton's method, former being linear and latter being quadratic, which agrees with the theorems in the book. The gradient descent particularly struggles with the first three functions. However, both methods manage to find the optima of all functions.