Numerical Optimization

Assignment 4

Xingrong Zong, tnd179

Hold 03, Group 5

March 1st, 2022

Introduction

I implement Algorithm 4.1 with exact solution of the sub-problem 4.3 (p_k fulfills Theorem 4.1).

Theoretical exercises

Exercise 1

In the book, Theorem 4.1 states that: The vector p^* is a global solution of the trust-region problem

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

if and only if p^* is feasible and there a scalar $\lambda \geq 0$, where B is strictly positive definite, $||p(0)|| > \Delta$ and $p(\lambda) = -(B + \lambda I)^{-1} g$, these conditions are satisfied. Also, from Theorem 4.1 we can get $||p^*|| = \Delta$. According to the algorithm we know that when we find an interval $[\lambda_0, \lambda_1]$, then $||p(\lambda_0)|| > \Delta$ and $||p(\lambda_1)|| < \Delta$. Now we assume $0 < \lambda_0 < \lambda_1$, when λ increases, $(B + \lambda I)^{-1}$ declines, which increases the overall p. Therefore, p is a monotone function in the interval $[0, \infty]$, and $[\lambda_0, \lambda_1] \subset [0, \infty]$. We have $\lambda' = \frac{\lambda_0 + \lambda_1}{2}$, $\lambda' \in [\lambda_0, \lambda_1]$, and since p is

monotone in the interval $[0, \infty]$, and $[\lambda_0, \lambda_1] \subset [0, \infty]$, when we update λ_0 or λ_1 to λ', the interval $[\lambda_0, \lambda']$ or $[\lambda', \lambda_1]$ always maintains a strict subset of $[\lambda_0, \lambda_1]$, so during n iterations:

$$\lim_{n\to\infty} ||p(\lambda_0)|| = \lim_{n\to\infty} ||p(\lambda_1)|| = \lim_{n\to\infty} ||p(\lambda')|| = \Delta$$

this means that we will eventually achieve convergence $||p(\lambda^*)|| = \Delta$. Therefore,

$$||p(\lambda^*)|| = \Delta = ||p^*|| \to p(\lambda^*) = p^*$$

Exercise 2

When B is indefinite, we want to have an interval which include an λ^* satisfies $||p(\lambda^*)|| = \Delta$. In order to achieve this, we assume there is an orthogonal matrix Q and a diagonal matrix $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_n, \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ such that $B = Q\Lambda Q^\intercal$, now we have

$$||p(\lambda)||^2 = \sum_{i=1}^{n} \frac{(q_i^\intercal g)^2}{(\lambda_i + \lambda)^2}$$

where q_i denotes the i-th column of Q. From the equation above we know that, if $q_i^\intercal g \neq 0$, then $\lim_{\lambda\to-\lambda_i} ||p(\lambda)|| = \infty$. When λ increases, $(\lambda_i + \lambda)^2$ increases and we get $\lim_{\lambda\to\infty} ||p(\lambda)|| = 0$.

From the attributes above, we can adjust the algorithm to let λ_0 (this is the concept from the algorithm in Theorem 4.1) equal to $-\lambda_1$ (this λ_1 is the first eigenvalue of matrix Λ), and we will have an interval $(-\lambda_1, \infty)$ which definitely include a λ^* satisfies $||p(\lambda^*)|| = \Delta$ (Fig 4.5 at page 85 in the book). So, we can choose two values in $(-\lambda_1, \infty)$ and make the iterations just the same as (1) to get convergence.

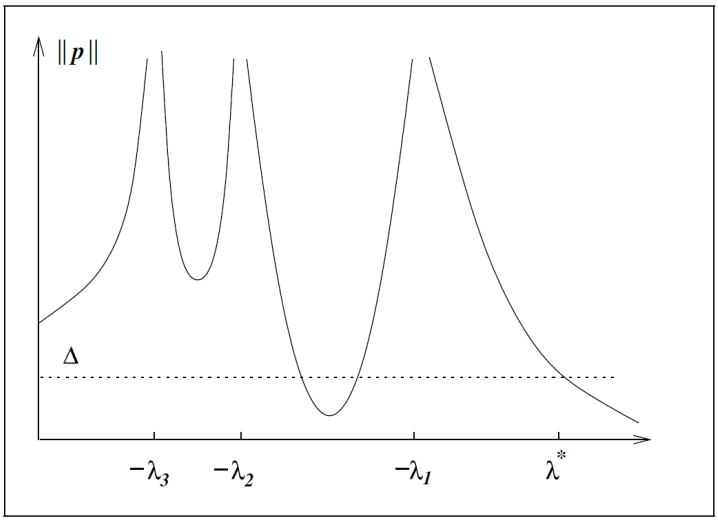

Figure 4.5 $\|p(\lambda)\|$ as a function of λ.

Figure 1: Fig 4.5 at page 85 in the book

Programming exercises

Sub-problem

To solve the sub-problem, we use Algorithm 4.3 in Section 4.3 t. It is based on root-finding Newton's Method and Cholesky factorization which is used for finding the values of λ in many iterations. To check whether all three conditions are satisfied for the found value of λ^* to test my implementation.

First, let's set the initial value $\lambda^0 = -\lambda_1^e + c$ (c which is a small constant and λ_1^e is the smallest eigenvalue of B). Then we assure that $\lambda^{(\ell)} > -\lambda_1$. Therefore, all eigenvalues of $(B + \lambda^\ell I)$ should be positive , and indefinite Hessians should not affect the entire calculation.

For the stopping criterion, we use a small parameter $\varepsilon = 10^{-7}$, which means that when $\|\lambda^\ell - \lambda^{\ell+1}\| < \varepsilon$, iteration stops.

Besides, we did a pruning of the function. If B is positive-definite and $\|B^{-1}g\| \leq \Delta$, the iteration can be terminated immediately with $\lambda^* = 0$.

Hard case

For the hard case $\lambda = -\lambda_1$, where λ_1 is the smallest eigenvalue of matrix B, it is not enough to delete the terms for which $\lambda_j = \lambda_1$ from the formula(4.38) and find p:

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j$$

Instead, we can use a vector z, where $||z|| = 1$ and $(B - \lambda_1 I)z = 0$, then for any scalar τ we have

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j + \lambda} q_j + z\tau$$

$$||p||^2 = \sum_{j:\lambda_j \neq \lambda_1} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2} + \tau^2$$

Therefore, it is always possible to choose τ to ensure that $||p|| = \Delta$, and it is easy to check that the condition(4.8) holds for this choice of p and $\lambda = -\lambda_1$. Therefore, we can set scalar τ:

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

The final formula is:

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

Parameters

- Initial trust region radius ($\Delta_0 = 10^{-10}/10$)
- Maximum trust region radius $\tilde{\Delta} = 10^3$
- Initial coefficient $\lambda^0 = -\lambda_1^e + 0.001$
- Tolerance $\varepsilon = 10^{-7}$

The whole process is sensitive to the parameter Δ, and the change of the initial value of A will have a great impact on the final result.

Testing protocol

We use following methods to test the effectiveness of my implementation:

- Average value of distance to the optimum at Line-Search algorithm by using Steepest Descent and Newtons algorithm, as well as Trust-region algorithm.
- Average number of iterations to the optimum at Line-Search algorithm by using Steepest Descent and Newtons algorithm, as well as Trust-region algorithm.
- The convergence plot with the number of iterations on the x-scale and the norm of gradient on the y-scale.
- The Trust-Region plot with the number of iterations on the x-scale and the Delta on the y-scale.

Result

We use a random starting point taken from the uniform distribution in the interval between -5 to 5 and repeated each optimization 100 times to get the median. Let's take f1 as an example:

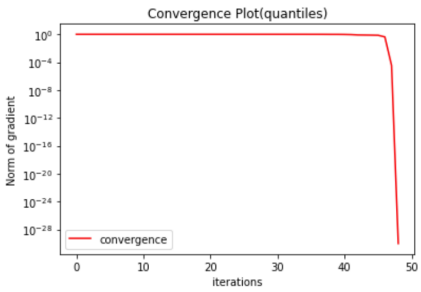

Figure 2: Norm of gradient of f1

Figure 3: Delta of f1

Then we compare it to the Line-Search algorithm by using Steepest Descent and Newtons algorithm about average value of distance and average number of iterations.

	f_1	f_2	f_3	f_4	f_5
stepest descent	3.24×10^{-4}	1.36×10^{-3}	2.57×10^{-8}	6.33×10^{-4}	6.28×10^{-2}
newton	0.0	1.53×10^{-6}	4.52×10^{-13}	3.19×10^{-4}	1.74×10^{-2}
trust region	0.0	2.18×10^{-11}	1.25×10^{-16}	1.14×10^{-7}	5.35×10^{-7}

Table 1: Average value of distance to the optimum at Line-Search algorithm by using Steepest Descent and Newtons algorithm, as well as Trust-region algorithm.

	f_1	f_2	f_3	f_4	f_5
stepest descent	2600	4022	1812	48	325
newton	2	15	15	3	11
trust region	6	22	28	14	24

Table 2: Average number of iterations to the optimum at Line-Search algorithm by using Steepest Descent and Newtons algorithm, as well as Trust-region algorithm.