Numerical Optimization - Hand-In 6

Isabela Blucher

March 16, 2018

Exercise 12.5

Rewriting the first unconstrained optimization problem we have $f(x) = \|v(x)\|_\infty = max\ |v_i(x)|, i = 1...m$. We define another problem equivalent to minimizing f, and that is minimizing t, where $|v_i(x)| \leq t, i = 1...m$. Our new optimization problem is

$$\begin{aligned} \min_x \quad & t \\ \text{subject to} \quad & t - v_i(x) \geq 0, i = 1...m, \\ & t + v_i(x) \geq 0, i = 1...m. \end{aligned}$$

For the second unconstrained minimization problem $f(x) = max\ v_i(x), i = 1...m$, we can define a similar problem to the first one, but without the absolute value on the v function, which means that the new problem is

$$\begin{aligned} \min_x \quad & t \\ \text{subject to} \quad & t - v_i(x) \geq 0, i = 1...m. \end{aligned}$$

Exercise 12.10

The constraints defined in (12.32) are $c_1(x) = 1 - x_1^2 - (x_2 - 1)^2 \geq 0$ and $c_2(x) = -x_2 \geq 0$. We calculate their respective gradients

$$\nabla c_1(x) = \begin{bmatrix} -2x_1 \\ -2(x_2 - 1) \end{bmatrix}, \nabla c_2(x) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Given $x^* = (0, 0)^T$, we see that in x^* both of our constraints are active. Calculating the gradients at x^* we get $\nabla c_1(x^*) = (0, 2)^T$ and $\nabla c_2(x) = (0, -1)^T$. By constructing a matrix with these gradients as columns and calculating its determinant we can see that the determinant is 0 and so the set $\{\nabla c_i | i \in A(x^*)\}$ is not linearly independent, and the LICQ does not hold.

For the MFCQ, we create $w = (w_1, w_2)^T$ and multiply the gradients at x^* by the vector w. For the first constraint we get $2w_2$ and for the second $-w_2$. Since there are no values of w_2 that could make both multiplications strictly greater than 0, it means that $\nexists w \in \mathbb{R}^2, \nabla c_i(x^*)^T w > 0$ and so the MFCQ does not hold.

Exercise 12.13

We can rewrite the given constraints in the standard form $c_1(x) = 2 - (x_1 - 1)^2 - (x_2 - 1)^2 \geq 0$, $c_2(x) = 2 - (x_1 - 1)^2 - (x_2 + 1)^2 \geq 0$ and $c_3(x) = x_1 \geq 0$. We then calculate the respective gradients for each constraint

$$\nabla c_1(x) = \begin{bmatrix} -2(x_1 - 1) \\ -2(x_2 - 1) \end{bmatrix}, \nabla c_2(x) = \begin{bmatrix} -2(x_1 - 1) \\ -2(x_2 + 1) \end{bmatrix}, \nabla c_3(x) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The definition of LICQ is that the set of active constraint gradients be a linearly independent set. In this exercise, all constraints are active at $x^* = (0, 0)^T$, which means that we have three active constraints in a 2-dimensional problem, so $\{\nabla c_i | i \in A(x^*)\}$ is not linearly independent, and thus LICQ does not hold.

For the MFCQ, if we set $w = (1, 0)^T$, all constraints multiplied by w will be strictly greater than 0, that is $\exists w \in \mathbb{R}^2, \nabla c_i(x^*)^T w > 0$ and so the MFCQ does hold.

Exercise 12.17

We can write the KKT conditions as $\nabla_X \mathcal{L}(x^*, \lambda^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* \nabla c_i(x^*) = 0$. Suppose that there exists λ_1, λ_2 that satisfy both the KKT and the LICQ. We then have

$$\nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_{i1} \nabla c_i(x^*) = \nabla f(x^*) - \sum_{i \in \mathcal{A}(x^*)} \lambda_{i2} \nabla c_i(x^*) = 0$$

$$\sum_{i \in \mathcal{A}(x^*)} (\lambda_{i2} - \lambda_{i1}) \nabla c_i(x^*) = 0$$

Since the LICQ holds, we know that the set $\{\nabla c_i | i \in A(x^*)\}$ is linearly independent, which means that for the sum above, the only way that it results in zero, is if $(\lambda_2 - \lambda_1) = 0 \implies \lambda_2 = \lambda_1$ which shows the Lagrange multiplier is unique. For the indexes that are not in the active set, that is, for inactive inequality constraints, the Lagrange multipliers are zero, and are also unique.

Programming Assignment

The implemented algorithm this week was a modification of Newton's method that deals with constrained optimization. At each step, we enforce the constraints for the descent and the Newton directions by projecting the coordinates that go out of the given bounds. The implementation for the step length computation was done with a line search within the feasible region of the given problem. The following discussion and tests are part of this week's convergence studies.

Parameter selection and algorithm sensitivity

The algorithm this week has different parameters to be analyzed. One could change the values of $B = A^T A$ and $c = A^T b$ for functions with greater dimensions, change the constraints m and M for different feasible regions and, as usual, start from various starting iterates inside the feasible region. In the following sections, we will see how these parameters affect our algorithm's efficiency, precision and robustness.

Stopping criteria

The stopping criteria for the algorithm were, the norm of the projected Newton step had to be smaller than a fixed tolerance value or the number of iterations had reached a fixed maximum quantity. The values selected for testing the algorithm were, for the Newton step gradient norm, 10^{-6}, and for the maximum number of iterations, 10^4.

Convergence studies

For the convergence plots, we will fix the following values for the function coefficients.

$$A = \begin{bmatrix} 2 & 3 & -1 \\ 3 & 0 & -1 \\ -8 & 2 & 1 \\ -1 & 2 & 4 \\ 1 & -1 & 3 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Our problem function has the form $f(x) = \frac{1}{2}\|Ax - b\|^2$ and we want to find a value of x that minimizes the function but is also satisfying the given constraint vectors m and M. For our convergence tests we will fix the constraints at $m_i = -5$ and $M_i = 5, \forall i = 1...3$ and randomly sample 100 starting iterates that respect the given constraints.

The figure below illustrate the convergence of the modified Newton's method. Both figures show a comparison between the norm of the gradient of the projected Newton step and the number of iterations, but Figure 2 is a log-plot.

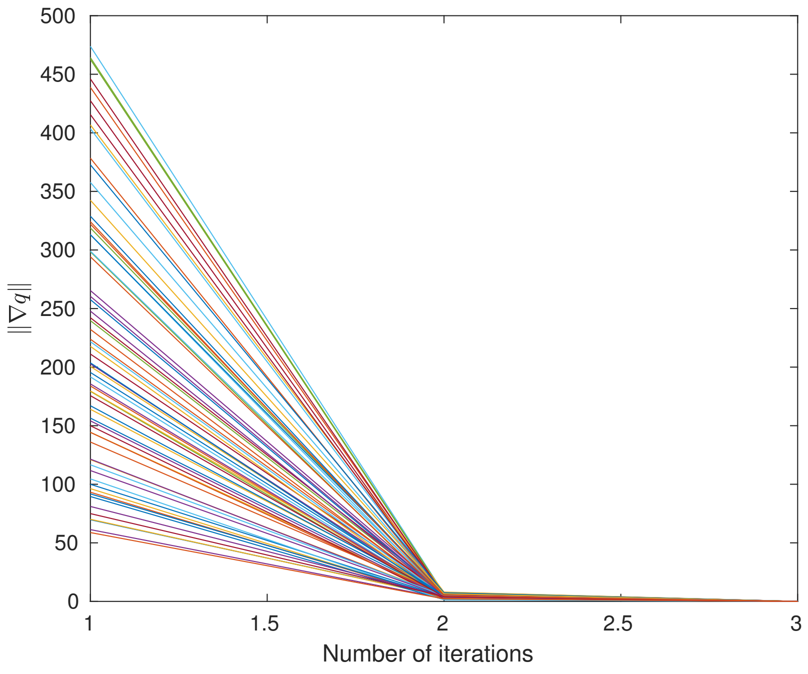

Figure 1: Convergence plot of projected Newton step gradient versus number of iterations

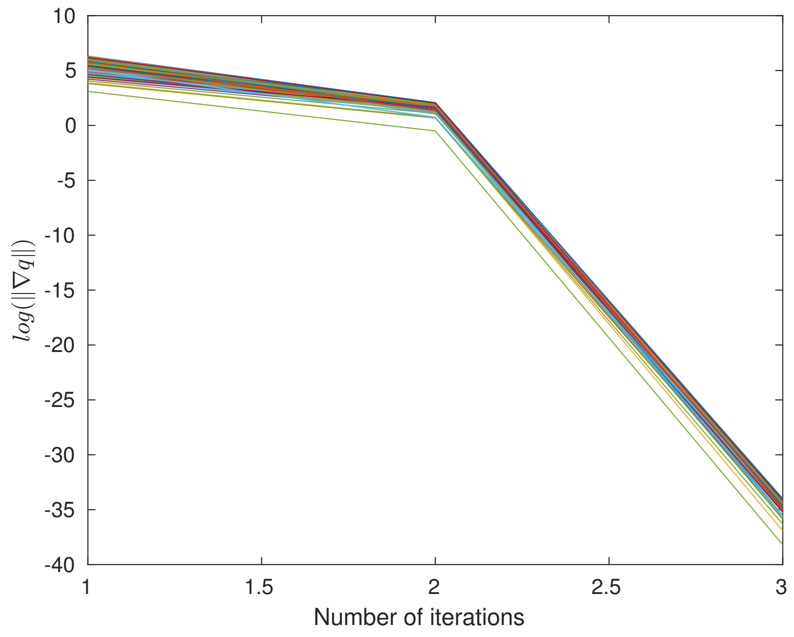

Figure 2: Convergence log-plot of projected Newton step gradient versus number of iterations

By visually inspecting Figure 2 we can see that it takes the shape of quadratic convergence, and by Figure 1 we see that for very different starting values of x_0 and different gradient norms, the algorithm seems to converge in no more than 3 iterations. Table 1 shows the mean, median and standard deviation of number of iterations until convergence for the same 100 random starting iterates.

Mean	Median	Standard Deviation
2.1900	2	0.8610

Table 1: Statistics on the number of iterations until convergence for 100 random starting iterates

From Table 1 we can conclude that our results are very stable in terms of efficiency. No run takes more than

3 iterations to converge and we don't have much variability in convergence speed.

Playing with constraints

In this section, we try to analyze if for constraints of a very different order of magnitude, the behavior of the algorithm changes. We keep the A and b values from the convergence tests in the section above. We also randomly sample 100 starting iterates that satisfy the given bounds.

Let's firstly make our feasible region smaller by setting $m_i = 0$ and $M_i = 1, \forall i = 1...3$. Below are the results obtained for these values of m and M.

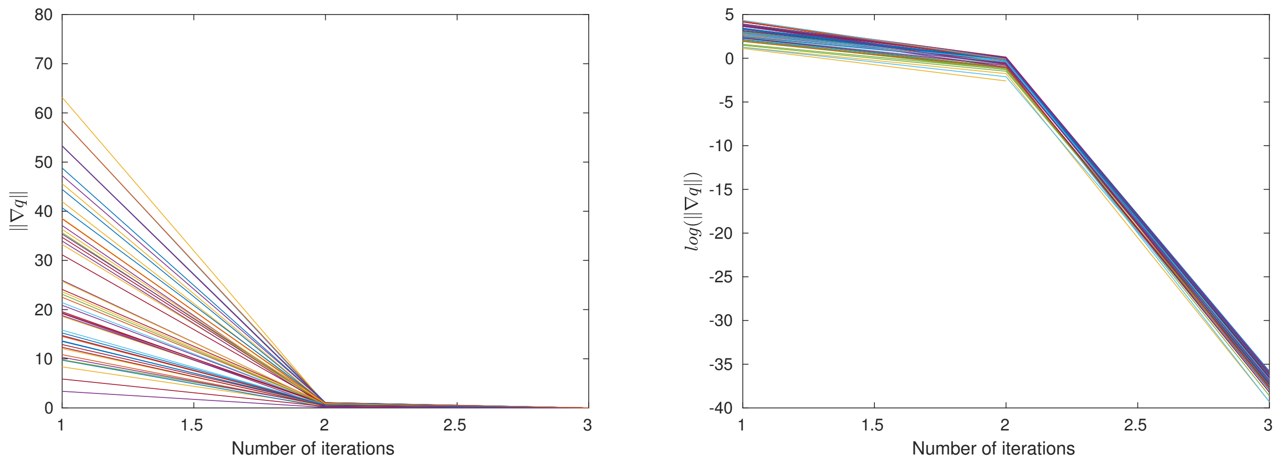

Figure 3: Convergence plot and log-plot for a smaller feasible region

Mean	Median	Standard Deviation
2.0600	3	0.9829

Table 2: Statistics on the number of iterations for a smaller feasible region

The plots in Figure 3, and the mean, median and standard deviation values in table 2 show almost no difference to our starting problem.

Let's make the feasible region bigger. We set $m_i = -100$ and $M_i = 100, \forall i = 1...3$. Below are the results for these values of m and M.

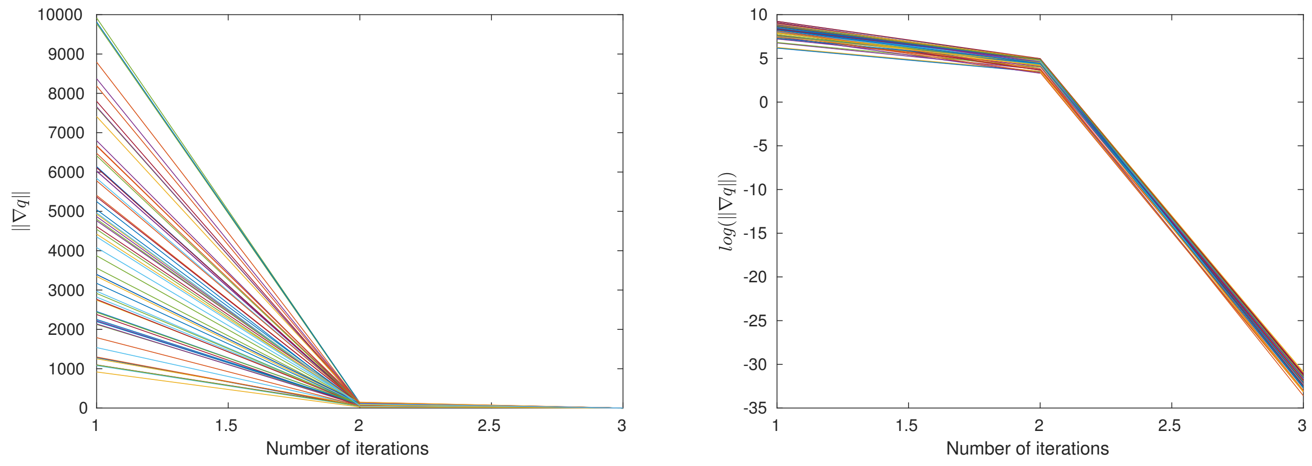

Figure 4: Convergence plot and log-plot for a larger feasible region

Mean	Median	Standard Deviation
2.1700	2	0.8883

Table 3: Statistics on the number of iterations for a larger feasible region

The plots in Figure 4 and the statistics in Table 3 also show no significant difference in the efficiency and precision of the algorithm. The only difference is the starting gradient norm for different constraints, due to our starting iterates being spread out randomly inside the feasible region.

Conclusion

The algorithm performs very well for a variety of starting iterates and constraint values. In terms of robustness, different points inside the feasible region yielded similar running times and results. In terms of efficiency it is very fast, as no run took more than 3 iterations to converge to a minimizer within the feasible region. One could assume that for higher dimensional problems, it would also be as efficient and robust, even if more computation is needed to reach convergence.