Homework 3

郭宇杰 B07611039

Problem 1: Hand-written Part

The equivalent QCQP is

$$\begin{array}{rl} \text{minimize}_{\tilde{x}\in\mathbf{R}^2, w\in\mathbf{R}} & w \\ \text{subject to} & f_1(\tilde{x}) - w \le 0 \\ & f_2(\tilde{x}) - w \le 0 \\ & f_3(\tilde{x}) - w \le 0, \end{array}$$

where $f_k(\tilde{x}) = \frac{1}{2}(\tilde{x} - y_k)^T P_k(\tilde{x} - y_k) + r_k$.

1.(a). Find dom f and derive $\nabla f(x)$ and $\nabla^2 f(x)$

Since $f(x) = tf_0(x) + \phi(x)$, **dom** $f = \{x \in \mathbf{R}^3 | f_k'(x) < 0 \ \forall k = 1, 2, 3\}$.
Let $F_i(x) = f_i(\tilde{x}) - w$, we have

$$\begin{aligned} \nabla f(x) &= t\nabla f_0(x) + \nabla\phi(x) \\ &= t \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \textstyle\sum_{i=1}^{3} \frac{1}{-F_i(x)} \nabla F_i(x) \end{aligned}$$

and

$$\begin{aligned} \nabla^2 f(x) &= t\nabla^2 f_0(x) + \nabla^2\phi(x) \\ &= \textstyle\sum_{i=1}^{3} \frac{1}{F_i(x)^2} \nabla F_i(x) \nabla F_i(x)^T + \sum_{i=1}^{3} \frac{1}{-F_i(x)} \nabla F_i(x). \end{aligned}$$

1.(b). Write a python function which takes inputs x, and t, and evaluates the function f at the point x, as well as the Gradient and the Hessian.

Please check my submitted code file named as my_objective_with_log_barrier.py for details. A function called *my_objective_with_log_barrier* derives the value of $f(x)$, gradient and Hessian of f.

Problem 2: Programming Part

2.(a). Newton Method Implementation

I follow the steps mentioned in the assignment and write a program called my_ objective_ with_ log_ barrier.py to implement Newton Method, and the results when $\mu = 20, 200, 2000, 20000$ are represented as below. **The submitted code and reference tables may answer the problem (c) to (k) and problem (m).**
Note that the sub-question numbers may be different with the assignment. Thus, I write the problem description after the sub-question number to ease TA's correct, thanks!

The number of outer iterations	1	2	3	4	5	6	7	8
The number of Newton steps	5	5	5	5	5	5	5	5
t	1	20	400	8000	1.6e+05	3.2e+06	6.4e+7	1.28e+9
$f(x)$	4.5419	120.9379	2.3259e+3	4.631e+4	9.2589e+5	1.8517e07	3.3035e+8	7.4070e+9
The number of total Newton steps	5	10	15	20	25	30	35	40
$x^*(t)$	(-0.2717, 0.2806, 8.1685)	(-0.4682, 0.2655, 5.8882)	(-0.4858, 0.2636, 5.7917)	(-0.4867, 0.2635, 5.7870)	(-0.4867, 0.2634, 5.7867)	(-0.4867, 0.2635, 5.7867)	(-0.4867, 0.2635, 5.7867)	(-0.4867, 0.2635, 5.7867)

Table 1: The results when $\mu = 20$.

The number of outer iterations	1	2	3	4	5
The number of Newton steps	5	8	6	6	6
t	1	200	4e+4	8e+6	1.6e+9
$f(x)$	4.5420	1.1672e+3	2.3149e+5	4.6294e+7	0.2587e+9
The number of total Newton steps	5	13	19	25	31
$x^*(t)$	(-0.2718, 0.2806, 8.1685)	(-0.4848, 0.2637, 5.7967)	(-0.4867, 0.2635, 5.7867)	(-0.4867, 0.2635, 5.7867)	(-0.4867, 0.2635, 5.7867)

Table 2: The results when $\mu = 200$.

The number of outer iterations	1	2	3	4
The number of Newton steps	5	9	8	9
t	1	200	4e+6	8e+8
$f(x)$	4.5419	1.1587e+4	2.3147e+7	4.6293e+10
The number of total Newton steps	5	14	22	31
$x^*(t)$	(-0.2718, 0.2806, 8.1685)	(-0.4865, 0.2635, 5.7877)	(-0.4867, 0.2635, 5.7867)	(-0.4867, 0.2635, 5.7867)

Table 3: The results when $\mu = 2000$.

The number of outer iterations	1	2	3
The number of Newton steps	5	12	9
t	1	2e+4	4e+8
$f(x)$	4.5420	1.1587e+5	2.3147e+9
The number of total Newton steps	5	17	26
$x^*(t)$	(-0.2718, 0.2806, 8.1685)	(-0.4867, 0.2635, 5.7868)	(-0.4867, 0.2635, 5.7867)

Table 4: The results when $\mu = 20000$.

2.(b). Make a comment on the relationship between the boundary and the central path

The number of active inequality constraints is 0 since $f_k(x^*(t)) < 0\ \forall k \in \{1, 2, 3\}$.

2.(c). Compare results

We can evidently find that as μ increases, the number of outer iterations decreases. Similar relationship occurs on the optimal values and the optimal points.

2.(d). Which μ used the lowest total number of Newton iterations

It can be shown from the table 1 to table 4 that $\mu = 20000$ used the lowest total number of Newton iterations. It used 26 steps at total.

2.(e). Use the cvx toolbox to solve the same problem

The CVX program is submitted and named as cvx_program.py. The results derived by CVX are that optimal value is about 5.78667, and the optimal solution $x^* = (-0.48678, 0.26347)$.
Compared with the results derived by Newton Method, we find that as μ increases, the corresponding optimal value and the optimal solution are more closed to the results derived by CVX.