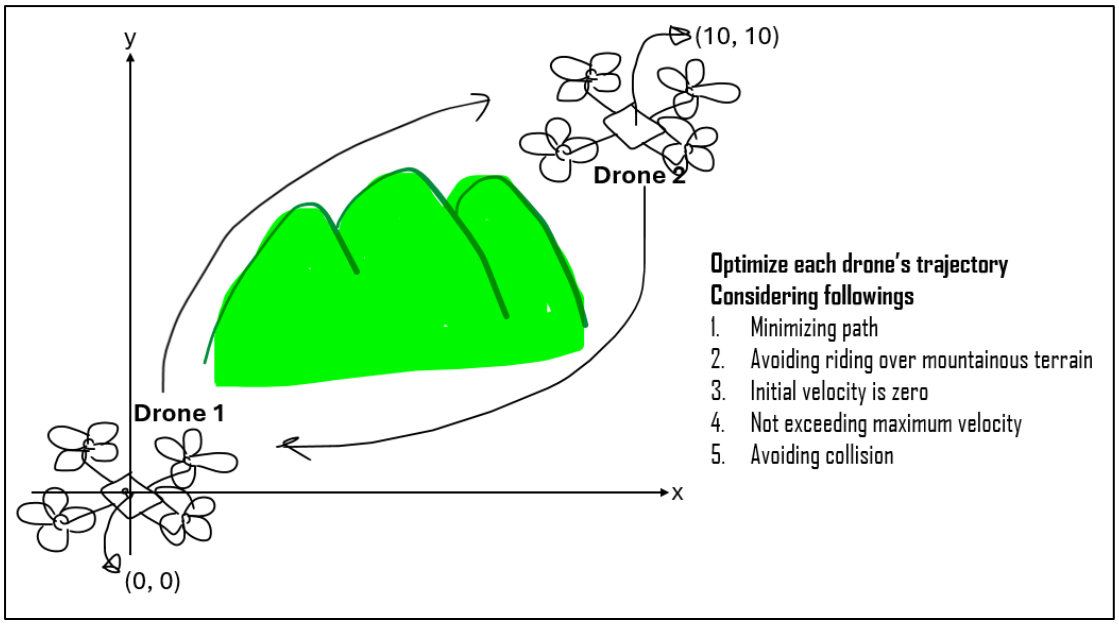
Revised version after due date of submission

MECH 579: Numerical Optimization
Department of Mechanical Engineering, McGill University
Assignment #5: Constrained Optimization: Optimize the Path of a Delivery Drone
5th. December, 2025

Name : Chiyoung Kwon
ID : 261258263
Department : Mechanical Engineering
Program : PhD



You are working for a large delivery company and are working building algorithms to design the path of the delivery drones. You are to develop an optimization problem that minimizes the cost of the path of the drone x(t) and the length of the path. The domain that the drone will travel through will be [0, 0] × [10, 10] with a total time of T = 15. The drone will have zero velocity at the beginning, and have a maximum speed of vmax = 1.5 units/s.

Consider a drone that needs to travel across Montreal, simulated by the following cost equation,

$$C(x, y) = \frac{1}{(x-5)^2 + (y-5)^2 + 1}. \tag{1}$$

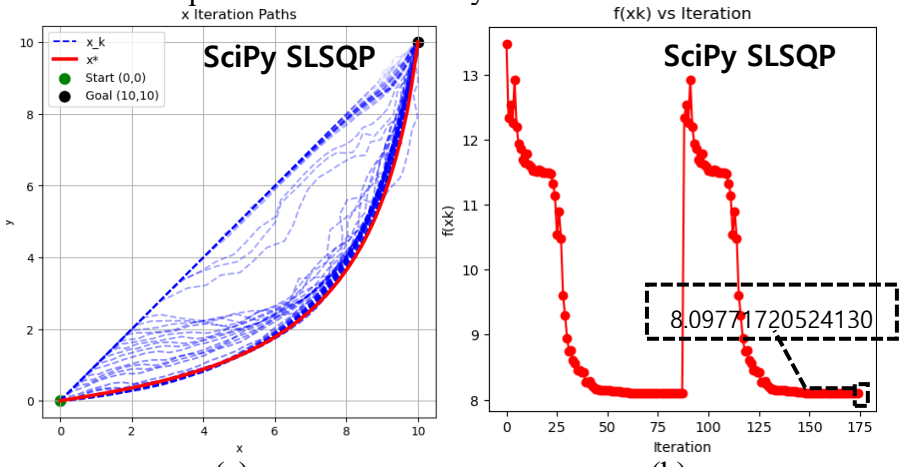
The optimization problem statement is defined as follows,

$$
\begin{aligned}
\min_{\mathbf{r}} \quad & \sum_{i=1}^{N} C(x_i, y_i) + \sum_{i=2}^{N} [(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2] \\
\text{with respect to} \quad & \mathbf{r} = x(t)\hat{i} + y(t)\hat{j} \\
\text{s.t.} \quad & (x(t_1), y(t_1)) = (0, 0) \\
& (x(t_N), y(t_N)) = (10, 10) \\
& \mathbf{V} = u(t_1)\hat{i} + v(t_1)\hat{j} = 0 \\
& \mathbf{V}_i^2 \leq v_{\max}^2 \\
& 0 \leq x_i \leq 10 \\
& 0 \leq y_i \leq 10,
\end{aligned} \tag{2}
$$

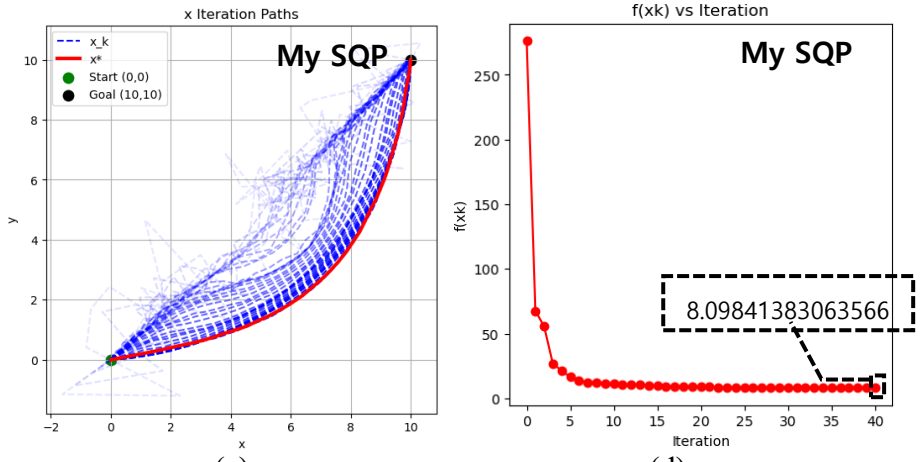
where $\mathbf{r}$ is the position of a drone in x and y coordinates and the velocity in each direction is defined as $u_i = u(t_i) = \frac{x_{i+1} - x_i}{dt}$ and similarly for v_i.

1. Solve the optimization problem with the **`scipy`** optimize package with $N = 45$,

   (a) Show the drone's path around the cost equation [Use a contour plot for the cost equation, scatter for the drone's location], including the initial guess. You can choose to either use the default finite-difference approach to compute the derivatives or provide the exact analytical derivatives.



(a) (b)



(c) (d)

Figure 1. (a) The trajectory of drone during the optimization and (b) the cost function's value at each iteration using SLSQP solver of SciPy. (c) The trajectory of drone during the optimization and (d) the cost function's value at each iteration using solver of SQP(based on augmented Lagrangian method and line search method) that I wrote.

The result of Figure 1 is taken from **`scipy.optimize.minimize('SLSQP')`** solver. I set the options maxiter set to 1000, gtol set to 1e-8. The initial trajectory is picked up as one having equal interval between each adjacent points.
The final iteration is 175 and the value of f(x*) is 8.097717205241308.
The reason why I used SLSQP solver is that TRUST-CONSTR solver didn't converge to the optimal solution such has the cost value of 8.097717205241308(maxiter/gtol were set as same with those of SLSQP and its final value of 1000th iteration was 10.472747723161383).
I also compared its result to that of SQP(based on augmented Lagrangian method and line search method) solver I wrote by myself. Even if its initial trajectory was picked up using random number, which is different from the case of SciPy SLSQP(because the initial trajectory having equal interval did not converge to the optimal trajectory in my SQP solver), it converged almost optimal trajectory.
Its cost value is 8.098413830635666 at 40th iteration which was set as maximum iteration number. It's expected to take so long for convergence since I wrote my SQP code using pure Python and NumPy not relying on any other compiled language.

(b) plot the convergence of the gradient of the Lagrangian with respect to the number of iterations.

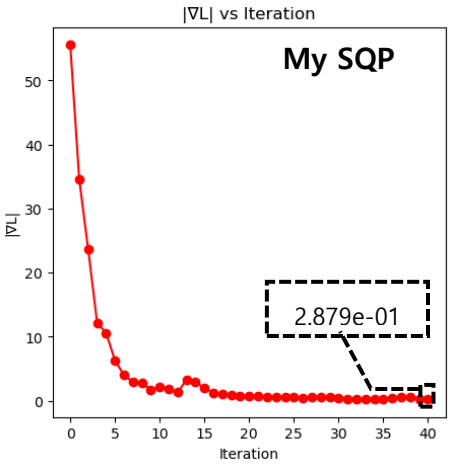


Figure 2. Gradient of Lagrangian function of my SQP solver.

As it is known, SciPy SLSQP does not provide the log of Lagrange multipliers. Instead of it, I could check it from my SQP solver.
Its final value was 0.2879131699800309 at 40th iteration.

(c) plot the value of the constraints as a function of iterations and provide a table listing the final constraint values.

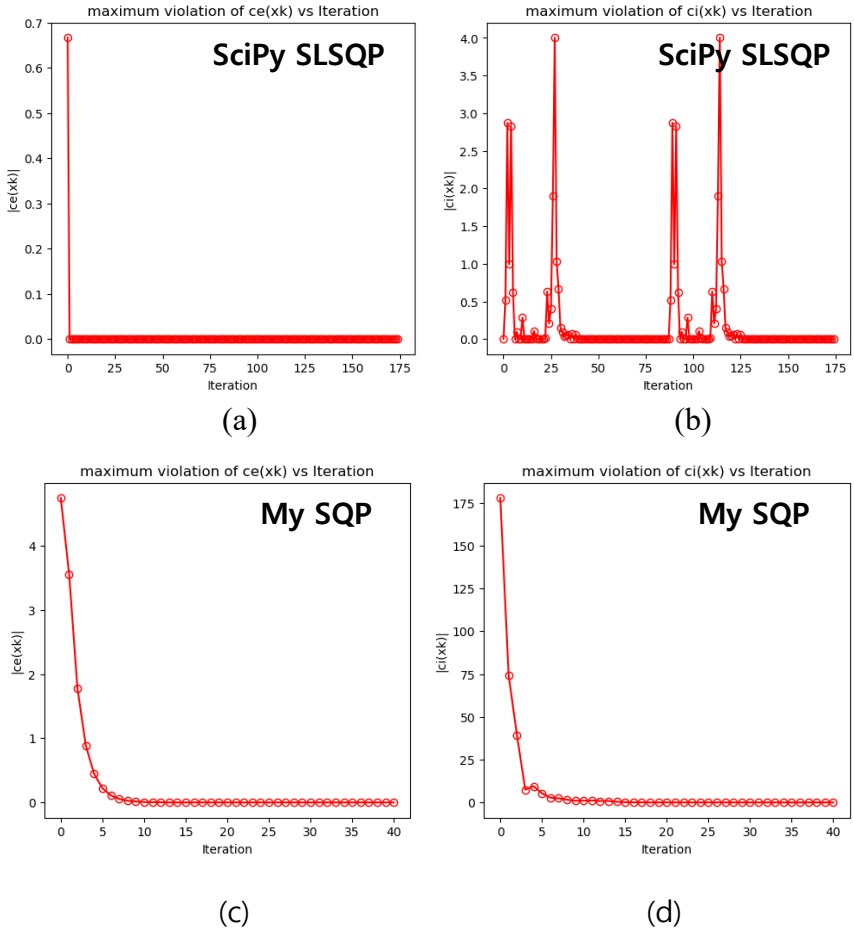


Figure 3. The maximum violation of equality constraint(a) and inequality constraint(b) in SciPy SLSQP. The maximum violation of equality constraint(c) and inequality constraint(d) in my SQP solver.

	SciPy SLSQP	My SQP solver
Maximum equality constraint at last iteration.	1.7357682434263162 e-32	4.506215912545634 e-06
Maximum inequality constraint at last iteration.	4.699722333394624 e-33	2.354211821555949 e-06

Table 1. The maximum violation of constraints in each solver at each last iteration .

2. Change the cost equation now to

$$C(x, y) = cos^2(x)cos^2(y) \tag{3}$$

to simulate a mountainous terrain. Solve the optimization problem again. Repeat steps (i), (ii), and (iii) from part (b).

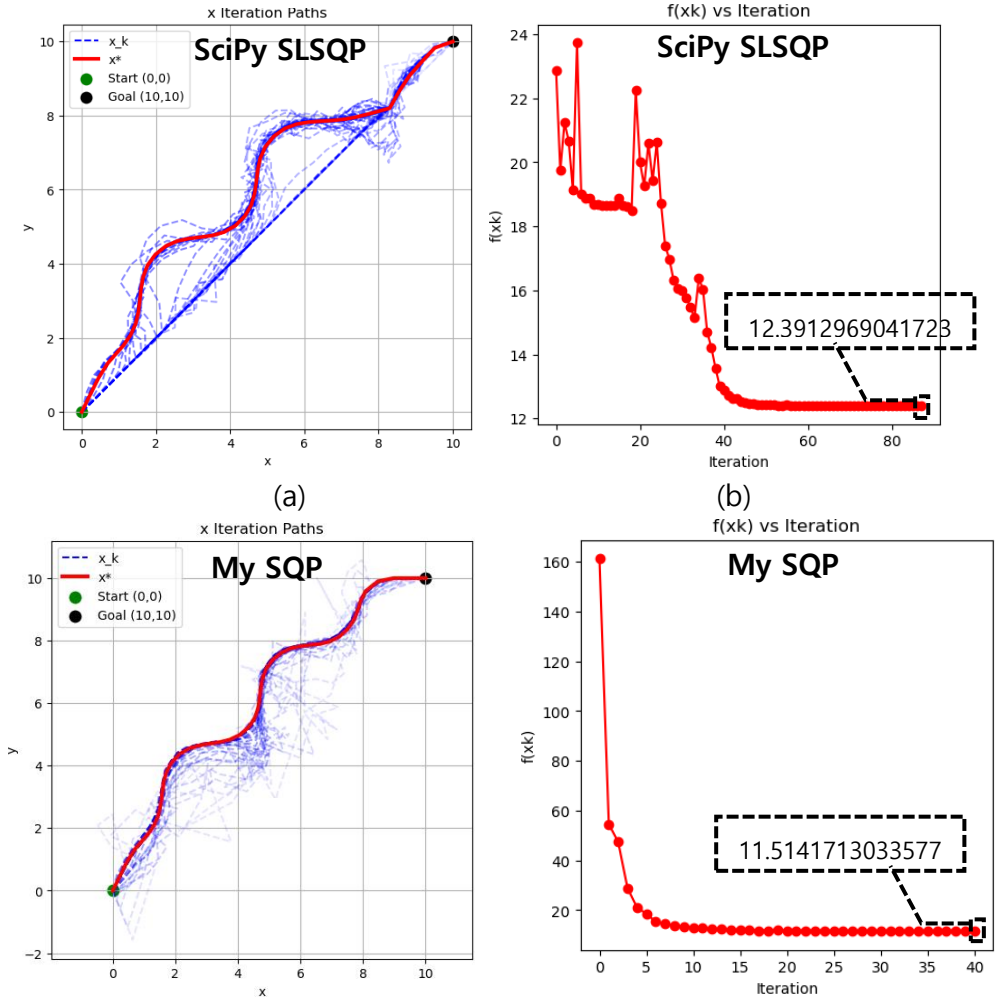
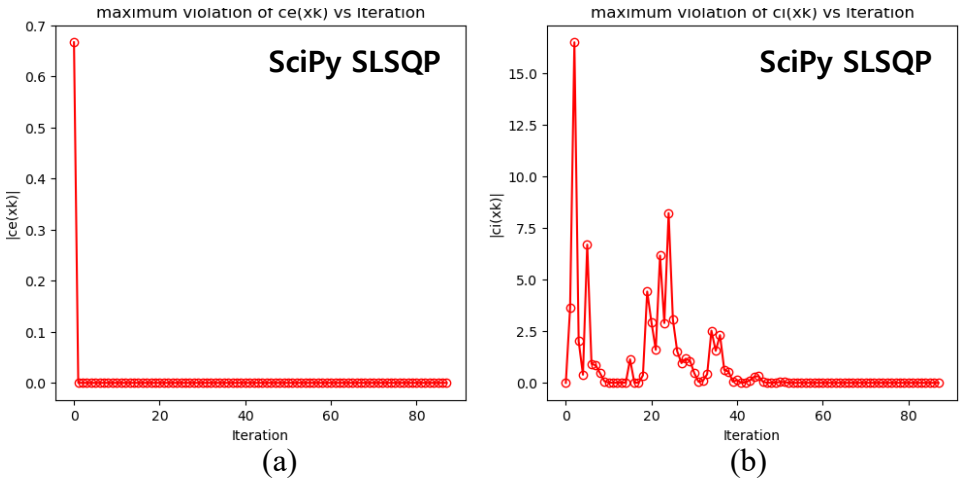
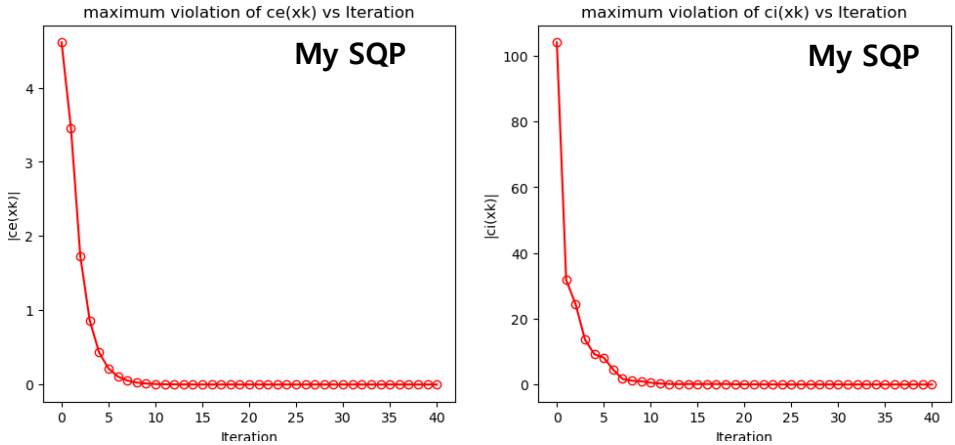


Figure 4. (a) The trajectory of drone during the optimization and (b) the cost function's value at each iteration using SLSQP solver of SciPy. (c) The trajectory of drone during the optimization and (d) the cost function's value at each iteration using my SQP solver.





(c) (d)

Figure 5. The maximum violation of equality constraint(a) and inequality constraint(b) in SciPy SLSQP. The maximum violation of equality constraint(c) and inequality constraint(d) in my SQP solver

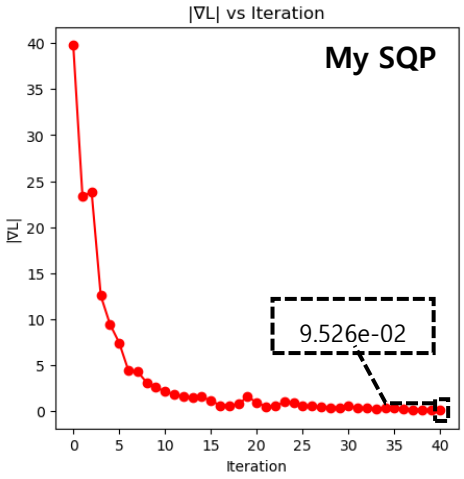


Figure 6. Gradient of Lagrangian function of my SQP solver.

	SciPy SLSQP	My SQP solver
Maximum equality constraint at last iteration.	3.748760151519011e-31	4.000508996071517e-06
Maximum inequality constraint at last iteration.	3.1771905106836584e-09	3.207603087576061 e-04

Table 2. The maximum violation of constraints in each solver at each last iteration.

3. Now add an additional drone that travels from (10, 10) to (0, 0). Introduce a new constraint that guarantees that drones will avoid collisions. Solve this problem, showing the drones' path around the cost equation. [Hint: you might find list comprehension useful]

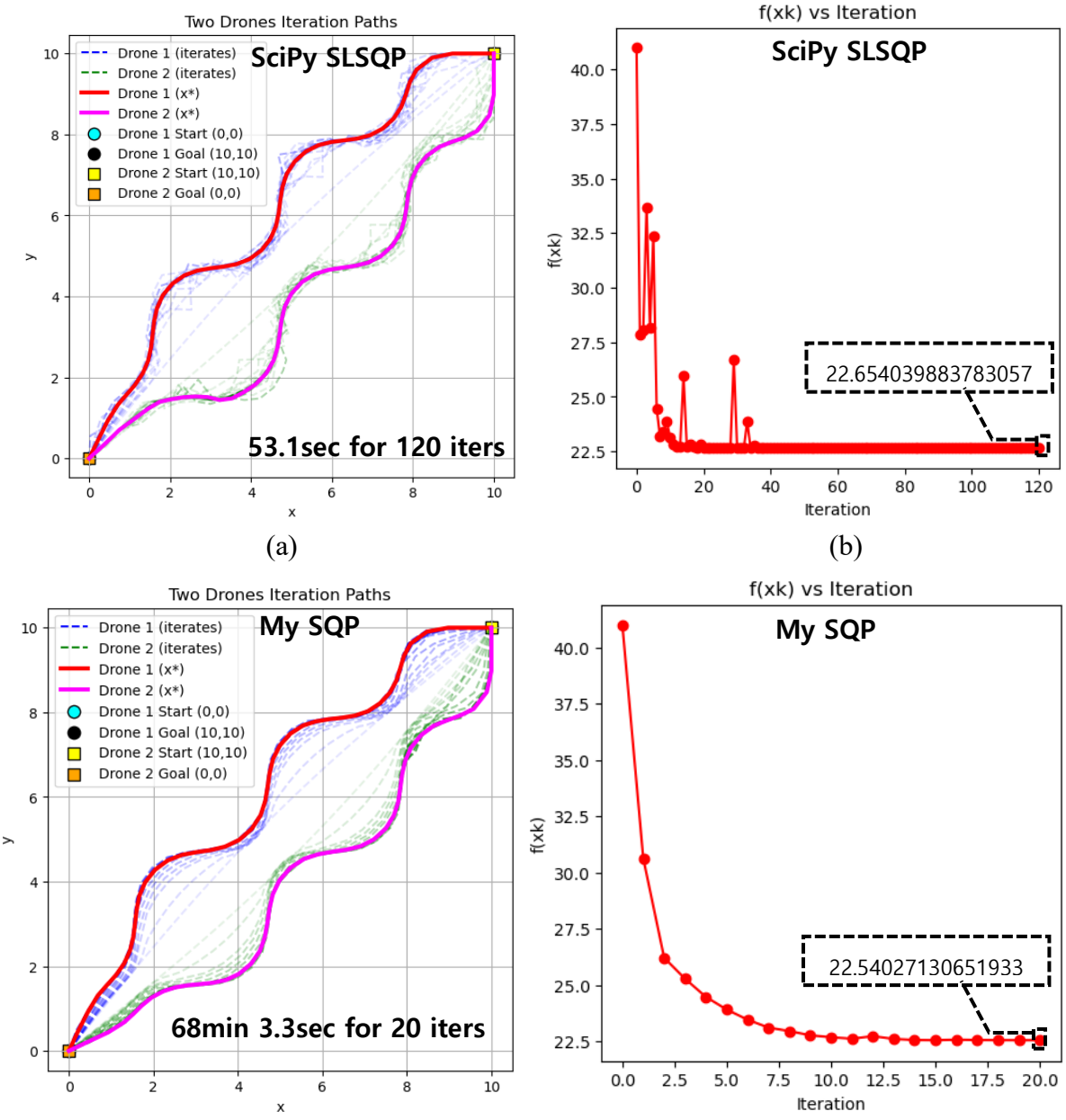
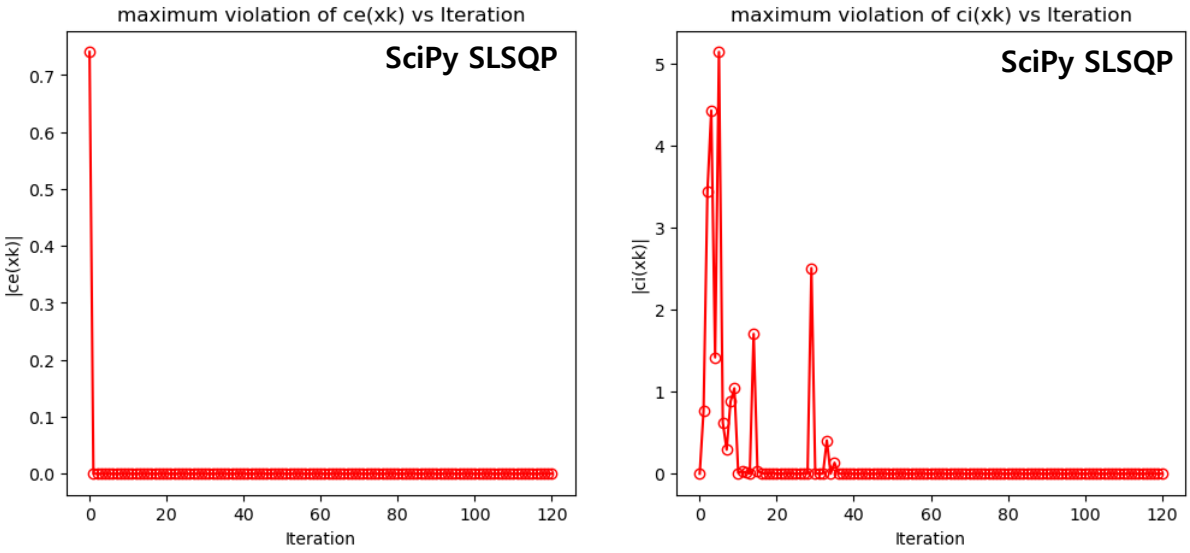
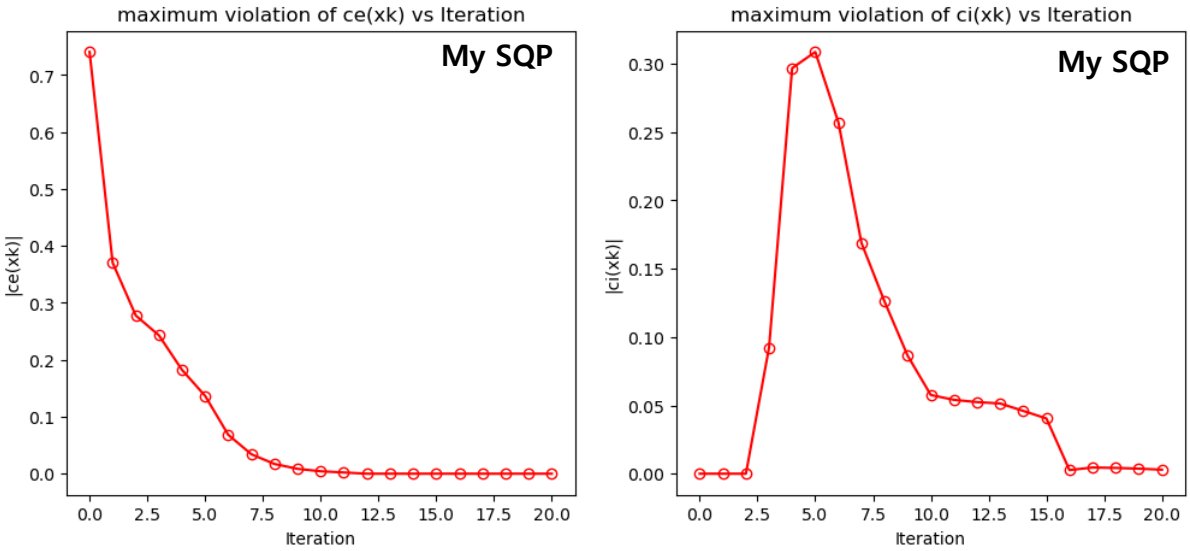


Figure 7. (a) The trajectory of drone during the optimization and (b) the cost function's value at each iteration using SLSQP solver of SciPy. (c) The trajectory of drone during the optimization and (d) the cost function's value at each iteration using my SQP solver.



(a) (b)



(c) (d)

Figure 8. The maximum violation of equality constraint(a) and inequality constraint(b) in SciPy SLSQP. The maximum violation of equality constraint(c) and inequality constraint(d) in my SQP solver.

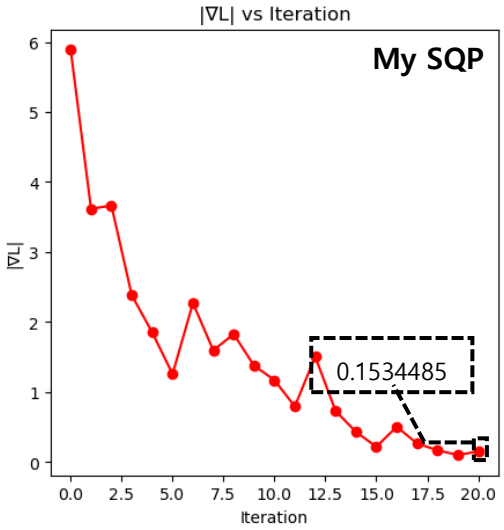


Figure 9. Gradient of Lagrangian function of my SQP solver.

	SciPy SLSQP	My SQP solver
Maximum equality constraint at last iteration.	1.9307586178347258e-22	2.1541872900456643e-06
Maximum inequality constraint at last iteration.	2.5863133856773857e-10	0.002952826600984171

Table 3. The maximum violation of constraints in each solver at each last iteration.