APPLICATION OF CONVEX OPTIMIZATION IN MOTION PLANNING FOR SELF-DRIVING VEHICLES

Owen Chang, Jihoon Kim, Zhuowen Zou

University of California San Diego, La Jolla, CA 92093, USA

1. TASK ASSIGNMENT

OC identified the task and provided the primal-dual formulation of the problem, JK reviewed related studies and ran experiments under different conditions, ZZ derived the mathematical formulation and implemented codes, and CKC supervised the students.

2. INTRODUCTION

With the continuous advancement in self-driving technology, self-driving cars have become ever more safe, accessible, efficient, and convenient [1]. Challenges and solutions are generated at every level of the decision-making hierarchy - route planning, behavioral specification, motion planning, and local feedback control. In particular, motion planning receives instruction from the behavioral layer and accomplish local navigation by predicting and selecting a continuous path through the environment, sending instructions to the control layer that executes the expected motion of the vehicle.

The spatiotemporal locality of the problem and the immediate effect that the vehicle may impose on the passengers has made motion planning an extremely complex yet popular problem. Indeed, in translating the selected behavior into a path or trajectory that can be tracked by the controller, the motion planning system needs to consider (1) dynamical feasibility for the vehicle, (2) passenger's comfort, and (3) collisions avoidance with obstacles detected by the on-board sensors. As a result, convex optimization becomes a good candidate for finding solutions that respect the constraint and optimize passenger experience by punishing uncomfortable maneuvers.

The motion planning problem involves the appropriate formulation of collision avoidance constraints, which are known to be non-convex and computationally intractable. One solution unconstrained optimization and its pioneering method is the artificial potential field (APF) by Khatib [2]. APF considers the movement of the ego vehicle/object in the planning space as a type of force motion in the virtual force field and has been improved and still used in many applications including unmanned aerial vehicle (UAV) [3]. In APF, the ego vehicle approaches to the target point under the composite influence of both attractive force and repulsive forces [4]. While APF is simple and produces smooth trajectories it can easily converge on the local minimum before it reaches the target when the distance between the target and obstacle is small. To overcome these limitations, constraints were added. The constrained optimizations for motion planning and collision avoidance problems fall into two classes, point-mass (PM) model and full-dimensional (FD) model, based on how controlled object is modeled [5]. In PM, the obstacles are represented as either polyhedron, where mixed-integer optimization is used ,or ellipsoid, where avoidance constraint is formulated as a smooth non-convex constraint solved by nonlinear programming solvers. In FD, the commonly adopted method is sequential quadratic programming, where a non-convex optimization problem is solved in iterative steps by constructing a convex sub-problem [6]. For example, Lampariello et al. [6] included inequality constrained of minimal distance between polytopes to be positive and Schulman et al. introduced a notion of signed distance, a sequential linearization technique [7]. Zhang et al. represented the controlled object and the obstacles as convex sets and converted the collision avoidance constraints as smooth non-convex constraints by reformulating the distance-function between two convex sets using strong duality [5]. Sequential Convex Programming (SCP) is also getting more attention by linearizing all non-convex elements to solve the convex problems in a local neighborhood of the region, where accurate linearization is achieved [8]. Successive Convexification (SCvx) [9] and Guaranteed Sequential Trajectory Optimization (GusTO) [10] are two most widely used SCP algorithms. Alternating direction method of multipliers (ADMM) [11] solved the optimization problems by breaking the original problem into many small sub-problems leveraging dual decomposition and augmented Lagrangian method [12, 13]. Recently, ADMM was adopted in operator splitting solver [14] for quadratic programs (OSQP) [15] to allow optimization even with no positive definiteness (PD) of the objective function or linear independence of the constraint functions.

In this paper, we introduce the convex optimization modeling techniques of self-driving vehicles and experimented with different objective functions. The goals is to create a sequence of vehicle control solution to enable self-driving to conform to road situation[1] while minimizing the discomfort caused by evasive deceleration using convex optimization

[16]. The performance of the method would be evaluated in CommonRoad (CR) benchmark scenarios [17]. We adopted ADMM based OSQP as it allows the optimization problem solving without the requirement of PD of the objective function. We compare this new approach, OSQP, with the relaxed condition to replace the original non PD matrix with PD matrix in the objective function. We experiment on the effect of relaxing condition and different values of velocity and jerk on the problem solution and the vehicle outcome.

Section 3 describes the convex optimization problem that models vehicle objectives and constraints. Section 4 describes the approach and implementation of the experiments. Sections 5 and 6 contains the results and relevant discussion.

3. PROBLEM STATEMENT

3.1. Primal

The longitudinal motion is modeled as $x = [s, v, a, j]^T$, where s is the longitudinal position, v is the velocity, a is the acceleration, and j is the jerk along a given reference path Γ. All components of vector x and the controller input u is defined as the rate of change respect to time in n-th order from the position s [18]:

$$v(t) = \frac{d}{dt}s(t) \tag{1}$$

$$a(t) = \frac{d}{dt}v(t) = \frac{d^2}{dt^2}s(t) \tag{2}$$

$$j(t) = \frac{d}{dt}a(t) = \frac{d^2}{dt^2}v(t) = \frac{d^3}{dt^3}s(t) \tag{3}$$

$$u(t) = \frac{d}{dt}j(t) = \frac{d^2}{dt^2}a(t) = \frac{d^3}{dt^3}v(t) = \frac{d^4}{dt^4}s(t) \tag{4}$$

The kinematic feasibility of the trajectory is constrained by

$$a_{min} \leq a(t) \leq a_{max} \tag{5}$$

$$v_{min} \leq v(t) \leq v_{max} \tag{6}$$

And the collision avoidance constraint are defined by obstacles blocking the reference path Γ, which is a function of time given moving obstacles such as other vehicles:

$$s_{min}^{(t)} \leq s(t) \leq s_{max}^{(t)} \tag{7}$$

The cost function we use aims to reach an expected velocity and punishing high accelerations and jerk with non-negative weights. In practice, this can translate to conforming to highway speed while favoring comfortable trajectories:

$$\begin{aligned}\mathcal{C}(x(t)) = \int_0^{t_h} & w_v(x^{(1)}(t) - v_{exp})^2 \\ & + w_a x^{(2)}(t)^2 + w_j x^{(3)}(t)^2 dt\end{aligned} \tag{8}$$

The input $u(t)$ and state $x(t)$ are subject to linear constraint,

$$u_{min} \leq u(t) \leq u_{max} \tag{9}$$

Our primal problem is the quadratic cost function over time horizon t_h:

$$\text{argmin}_u \int_0^{t_h} \mathcal{C}(x(t))dt \tag{10}$$

Subject to the above constraints.

In practice, the computations are carried out in a discretized fashion. We may therefore reformulate the variables as an array indexed by time steps t with dt the length of the time steps as opposed to a function of continuous time t. In particular, we reformulate the 4^{th} order constraints of Equation (4) via Taylor Expansion as a linear-time-invariant (LTI) system:

$$x_{t+1} = Ax_t + Bu_t, \ t = 0, \ \cdots, \ T-1 \tag{11}$$

Where T is the total number of timesteps and

$$A = \begin{bmatrix} 1 & \frac{dt}{1!} & \frac{dt^2}{2!} & \frac{dt^3}{3!} \\ 0 & 1 & \frac{dt}{1!} & \frac{dt^2}{2!} \\ 0 & 0 & 1 & \frac{dt}{1!} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ B = \begin{bmatrix} \frac{dt^4}{4!} \\ \frac{dt^3}{3!} \\ \frac{dt^2}{2!} \\ \frac{dt}{1!} \end{bmatrix}$$

Boundary constraints in Equation (5)-(7), and (9) are easily transformed. The total cost function at time t is discretized to

$$\begin{aligned}\mathcal{C}(x) &= \sum_{t=0}^{T} \mathcal{C}(x_t) \\ &= \sum_{t=0}^{T} (w_v(v - v_{exp})^2 + w_a a_t^2 + w_j j_t^2)dt \\ &= dt(w_v\|v - v_{ref}\|_2^2 + w_a\|a\|_2^2 + w_j\|j\|_2^2)\end{aligned}$$

After simplification, we are solving the convex optimization problem, for each timestep t,

$$\begin{aligned}\min_{u_t} \quad & (x_{t+1} - v_{ref})^T Q(x_{t+1} - v_{ref}) \\ \text{s.t.} \quad & x_{min}^{(t)} \leq x_t \leq x_{max}^{(t)} \\ & u_{min} \leq u_t \leq u_{max} \\ & x_{t+1} = Ax_t + Bu_t \\ & x_0 = x_{initial}\end{aligned}$$

where

$$v_{ref} = [0, v_{ref}, 0, 0]^T,\ c_{ref} \in R$$
$$x_{min}^{(t)} = [s_{min}^{(t)}, v_{min}, a_{min}, j_{min}]^T$$
$$x_{max}^{(t)} = [s_{max}^{(t)}, v_{max}, a_{max}, j_{max}]^T$$
$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & w_v & 0 & 0 \\ 0 & 0 & w_a & 0 \\ 0 & 0 & 0 & w_j \end{bmatrix}$$

3.2. Dual

The formulation for our cost function is time-dependent. To compute the current cost x_{t+1},we will need to know what the values of x_t are, which will require the values of x_{t-1} and so on. To avoid making the derivatives from running into a complicated recursion, we set x_0 equals to the initial state and $u_0 = 0$, and we start from computing the optimal u_1 and moving forward in time steps. This way we will be able to use x_t and u_{t-1} readily as it has already been computed in the previous iterations. Hence, at every iteration, we can set x_t and u_{t-1} to be a constant.

The Lagrangian is shown in the following equation and newly introduced A' and B' to incorporate our constraint on u_t, as well as appending u_t to x_t toward simpler notation:

$$\begin{aligned} L(x_{t+1}, \lambda_{max}, \lambda_{min}) = (x_{t+1} - v_{ref})^T Q (x_{t+1} - v_{ref}) \\ - \lambda_{min}^T (x_{t+1} - x_{min}^{(t)}) \\ + \lambda_{max}^T (x_{t+1} - x_{max}^{(t)}) \end{aligned} \tag{12}$$

where

$$x_{t+1} = A' x_t + B' u_t \tag{13}$$

$$\lambda_{max} = (\lambda_{max}^{(s)}, \lambda_{max}^{(v)}, \lambda_{max}^{(a)}, \lambda_{max}^{(j)}, \lambda_{max}^{(u)})^T \tag{14}$$

$$\lambda_{min} = (\lambda_{min}^{(s)}, \lambda_{min}^{(v)}, \lambda_{min}^{(a)}, \lambda_{min}^{(j)}, \lambda_{min}^{(u)})^T \tag{15}$$

$$A' = \begin{bmatrix} 1 & \frac{dt}{1!} & \frac{dt^2}{2!} & \frac{dt^3}{3!} & 0 \\ 0 & 1 & \frac{dt}{1!} & \frac{dt^2}{2!} & 0 \\ 0 & 0 & 1 & \frac{dt}{1!} & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B' = \begin{bmatrix} \frac{dt^4}{4!} \\ \frac{dt^3}{3!} \\ \frac{dt^2}{2!} \\ \frac{dt}{1!} \\ 0 \end{bmatrix} \tag{16}$$

Since $\frac{dx}{du_t} = B$,

$$\bigtriangledown L_{u_t} = 2Q(x_{t+1} - v_{ref}) \cdot B' - \lambda_{min}^T B' + \lambda_{max}^T B' = 0 \tag{17}$$

$$x_{t+1} = \frac{1}{2} Q^\dagger (\lambda_{max} - \lambda_{min}) - v_{ref} \tag{18}$$

Using eq. (16), we get:

$$u_t = (\frac{1}{2} Q^\dagger (\lambda_{max} - \lambda_{min}) - v_{ref} - A' x_t)/B' \tag{19}$$

Hence:

$$\begin{aligned} &g(\lambda_{max}, \lambda_{min}) = \\ &(\frac{1}{2} Q^\dagger (\lambda_{max} - \lambda_{min}) - v_{ref})^T (\frac{1}{2} (\lambda_{max} - \lambda_{min}) - v_{ref}) \\ &+ \lambda_{max}^T (x_{max}^t - \frac{1}{2} Q^\dagger (\lambda_{max} - \lambda_{min}) + v_{ref}) \\ &+ \lambda_{min}^T (\frac{1}{2} Q^\dagger (\lambda_{max} - \lambda_{min}) - v_{ref} - x_{min}^t) \end{aligned} \tag{20}$$

After simplification, our dual problem becomes:

$$\begin{aligned} \max_{\lambda_{min}, \lambda_{max}} g(\lambda_{max}, \lambda_{min}) = \\ (\frac{1}{4} - \frac{1}{2} v_{ref} - \frac{1}{2} (\lambda_{min} - \lambda_{max})^T) Q^\dagger (\lambda_{min} - \lambda_{max})^T \\ + \lambda_{max} (\frac{1}{2} v_{ref} + x_{max}^t) \\ - \lambda_{min} (\frac{1}{2} v_{ref} + x_{min}^t) \\ s.t.\ 0 \preccurlyeq \lambda_{max}, \lambda_{min} \end{aligned} \tag{21}$$

3.3. KKT Condition

For a solution to be optimal, Karush-Kuhn-Tucker (KKT) conditions have to be satisfied. The following are the four KKT conditions:

3.3.1. Primal feasibility

$$\begin{aligned} x_{min}^{(t)} - x \preccurlyeq 0 \\ x - x_{max}^{(t)} \preccurlyeq 0 \\ Ax_t + Bu_t - x_{t+1} = 0, t = 1, \cdots, T-1 \\ x_0 = x_{initial} \end{aligned} \tag{22}$$

3.3.2. Dual feasibility

$$\begin{aligned} 0 \preccurlyeq \lambda_{max} \\ 0 \preccurlyeq \lambda_{min} \end{aligned} \tag{23}$$

3.3.3. Complementary slackness

$$\begin{aligned} \lambda_{max}(x - x_{max}^t) = 0 \\ \lambda_{max}(x_{min}^t - x) = 0 \end{aligned} \tag{24}$$

3.3.4. Gradient of Lagrangian with respect to u_t

$$\bigtriangledown L_{u_t} = 2Q(x_{t+1} - v_{ref}) \cdot B' - \lambda_{min}^T B' + \lambda_{max}^T B' = 0 \tag{25}$$

4. APPROACHES

To evaluate the effectiveness of our method, we simulated our experiment using CommonRoad [19]. CommonRoad is a framework for Composable benchmarks for motion planning on roads are proposed so that numerical experiments are fully defined. Each experiment is composed of a vehicle model, a scenario, and a cost function. The vehicle model determines the ego-vehicle whose motions are to be planned. A scenario defines the structure of the road as well as the motions of the obstacles (e.g. other cars, blockage, etc.). And the cost function, as discussed in the previous section, implies the desired behavior of the ego-vehicle. For testing our experiment, we selected a few preset highway scenarios as well as designed our own. We used CVXPY[20] to formulate and solved our problem. Since we are only concerned with convex optimization on the linear speed, we delegated angular speed decision to the built-in Greedy-Best-First Search algorithm [19], whose choice depends only on collision.

In-vehicle trajectory planning, the weight matrix Q is not positive definite as it is a diagonal matrix with the first diagonal entry Q_{11} is zero. But to get the solution we wanted to experiment on relaxing this condition by adding a small non-zero value to entry Q_{11}. Then Q is positive definite and the problem reduces to the standard quadratic programming (QP), well-defined and easy to solve. We wanted to see how this change would impact the solution.

While the selection of weight is out of the scope of the optimization problem and is quite subjective, we want to observe the effects of each term in the optimization function, we have selected a subset of punishing weights w_v, w_a, w_j and plot the difference between their planned motion.

5. RESULTS

The goal was for the ego vehicle to reach and maintain the desired velocity, 31 meters per second (m/s) or 70 miles per hour (mph), in the presence of moving vehicles, leading and following, under constraints, while avoiding large acceleration and jerk that introduce discomfort to the passengers.

5.1. Case when Q is not PD

We set the weight matrix Q as nonpositive definite (PD) with the entry Q_{11} zero to reflect the realistic trajectory scenario. The ego vehicle (green) reached the desired velocity 31 m/s without collision to an obstacle, other moving vehicles, or curb-side. Figure 1 show how position, velocity, acceleration, and jerk changes along time point (x-axis). The ego vehicle is shown in green color and its position line does not cross over with neither the leading vehicle (red) or the following vehicle (blue). The magnitude of jerk, j, was large early but later, it decreased. Figure 2 depicts trajectory planning for each 8th time point.

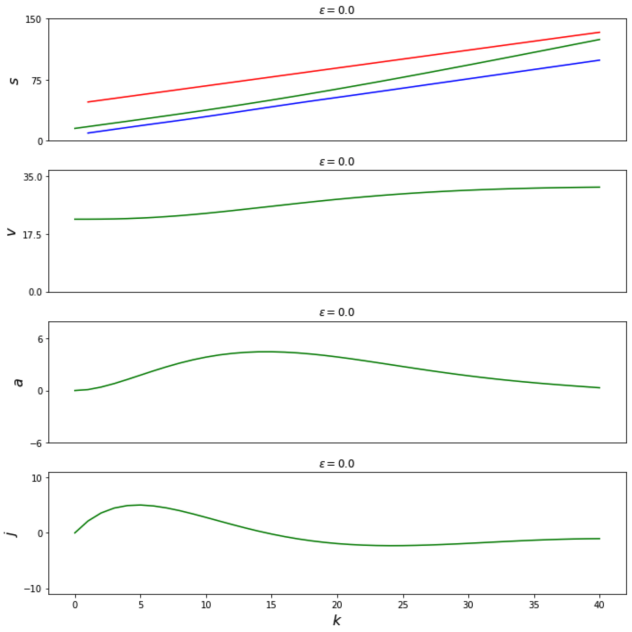


Fig. 1: The plot of optimization solution when Q is not PD. Vehicles are colored in green (ego-vehicle), red (static vehicle), and blue (leading and following vehicles).

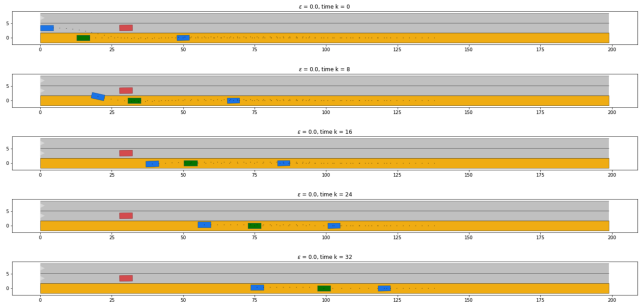


Fig. 2: The plot of trajectory planning when the weight matrix Q is not positive definite (PD) with $Q_{1,1}$ entry as $\epsilon = 0$. Vehicles are colored in green (ego-vehicle), red (leading vehicle ahead of the ego vehicle), and blue (following vehicle tailgating the ego vehicle). The Y-axes are s: longitudinal position, v: velocity, a : acceleration, j: jerk, k: discrete time point

5.2. Case when Q is PD with small perturbation

In most cases of real life vehicle trajectory planning, Q is not PD and we learned from CSE203 class that problems are

often relaxed for faster problem-solving. For this reason, we replaced the first diagonal entry Q_{11}'s current value of zero with a small non-zero value $= 0.05$ to make matrix Q_ϵ PD after perturbing by ϵ on non PD matrix Q. See the new matrix Q_ϵ below.

$$Q_\epsilon = \begin{bmatrix} \epsilon & 0 & 0 & 0 \\ 0 & w_v & 0 & 0 \\ 0 & 0 & w_a & 0 \\ 0 & 0 & 0 & w_j \end{bmatrix}$$

We wanted to test following two things:

1. Would our ego vehicle still get successful trajectory planning results with perturbed Q_ϵ by ϵ as in unperturbed Q?
2. How would solution values changes by perturbation by ϵ to Q?

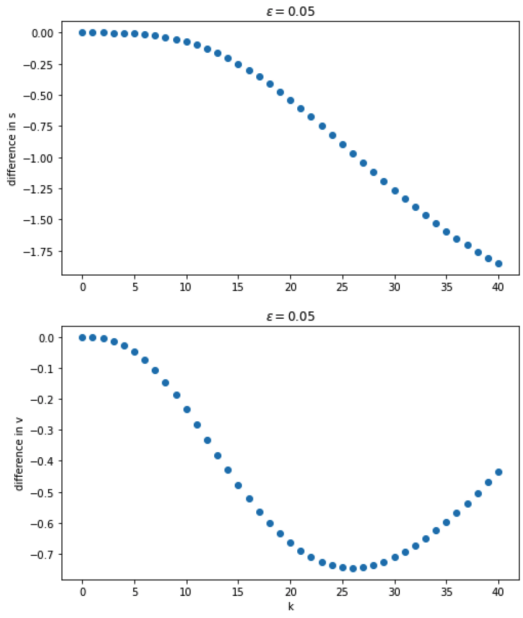


Fig. 3: Plot of difference in solutions when Q is PD with $\epsilon = 0.05$. The difference was computed by subtracting the solution of non-PD from that of PD.

Figure 3 shows the difference in results by subtracting the results of Q from Q_ϵ. The position was underestimated by adding ϵ and the final position was off by -1.75 m. The difference in velocity was also underestimated but the rate of change of the difference transitioned from negative to positive. Despite the notable difference after perturbation by ϵ, our ego vehicle (green color) still met the target reference velocity without collision (Figure 4). And the trajectory planning shows the ego vehicle (green color).

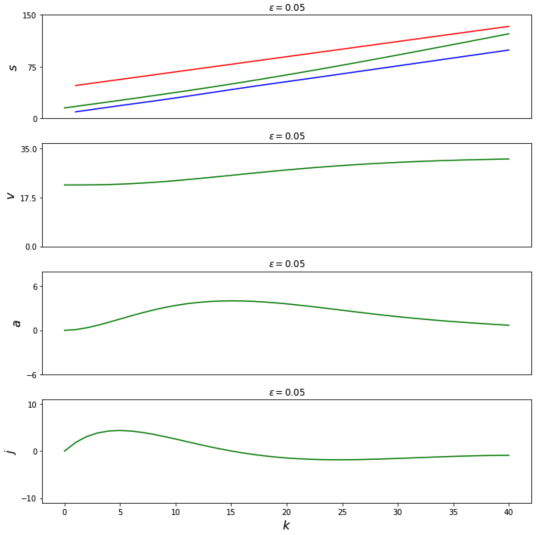


Fig. 4: The plot of optimization solution when Q is PD with $\epsilon = 0.05$. Vehicles are colored in green (ego-vehicle), red (static vehicle), and blue (leading and following vehicles).

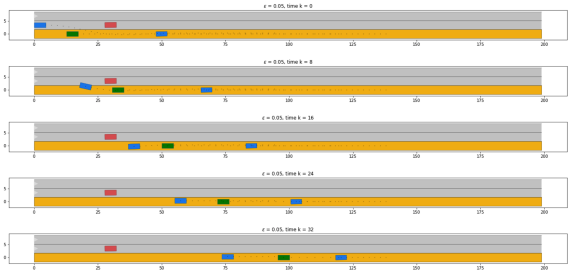


Fig. 5: The plot of trajectory planning when Q is PD with $\epsilon = 0.05$. Vehicles are colored in green (ego-vehicle), red (leading vehicle ahead of the ego vehicle), and blue (following vehicle tailgating the ego vehicle). Y-axes are s: longitudinal position, v: velocity, a : acceleration, j: jerk, k: discrete time point

5.3. Case when Q is PD with large perturbation

This time increased the perturbation size to $\epsilon = 1.0$. Matrix Q is again PD so problem becomes easier to solve. However, the trajectory planning ended up 25 off from that of the unperturbed scenario (Figure 6) and the ego vehicle collided with the following vehicle between time point 25 and 35 as shown in Figure 7 and Figure 8. These finding tell us that adding small perturbation is useful but not too much.

5.4. Effect of velocity punishment

Figure 9 visualizes the effects of velocity squared error term $w_v\|v - v_{ref}\|_2^2$ on motion planning. We fix the weight of acceleration ($w_a = 1$) and jerk ($w_a = 1$) punishment and conducted experiment on preset highway scenario USA_US101-

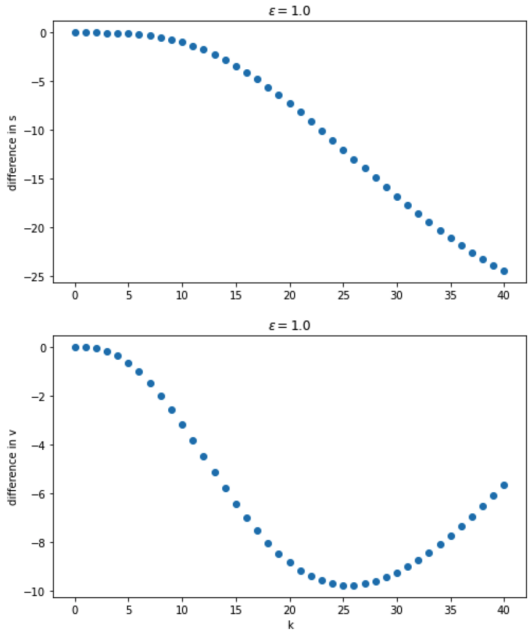


Fig. 6: The plot of difference in solutions when Q is PD with $\epsilon = 1.0$. The difference was computed by subtracting the solution of non-PD from that of PD.

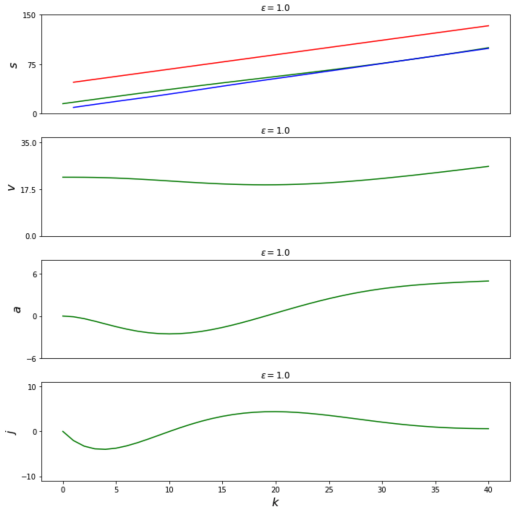


Fig. 7: The plot of optimization solution when Q is PD with $\epsilon = 1.0$. Vehicles are colored in green (ego-vehicle), red (static vehicle), and blue (leading and following vehicles).

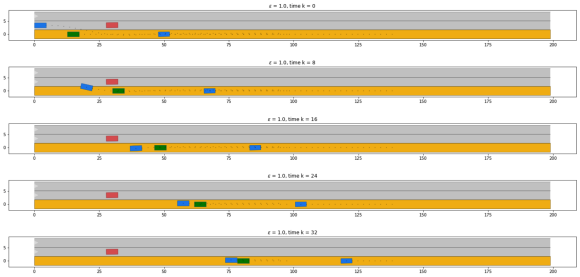


Fig. 8: Plot of trajectory planning when Q is PD with $\epsilon = 1.0$. Vehicles are colored in green (ego-vehicle), red (leading vehicle ahead of the ego vehicle), and blue (following vehicle tailgating the ego vehicle). Y-axes are s: longitudinal position, v: velocity, a : acceleration, j: jerk, k: discrete time point

11_4_T-1 with high ($w_v = 2$) and low ($w_v = 0.2$) velocity weight.

As shown in the figure, for the main part of the experiment, the ego vehicle accelerates to catch up with the vehicle in front of its lane. At the end, where the front vehicle is close the ego vehicle with high-velocity priority has chosen to change lane, while the one with low priority remains in its lane. The difference in the decision seems to conform to conventional driving behavior: if the driver intends to keep their speed, they chose to change lane when possible, and if the driver does not worry too much about it, they chose to stay in the lane and decelerates if necessary.

5.5. Effect of jerk punishment

Figure 10 visualizes the effects of jerk term $w_j \|j\|_2^2$ on motion planning. We fix the weight of velocity ($w_v = 1$) and acceleration ($w_a = 1$) and conducted experiment on preset highway scenario USA_US101-22_4_T-1 with different jerk weights.

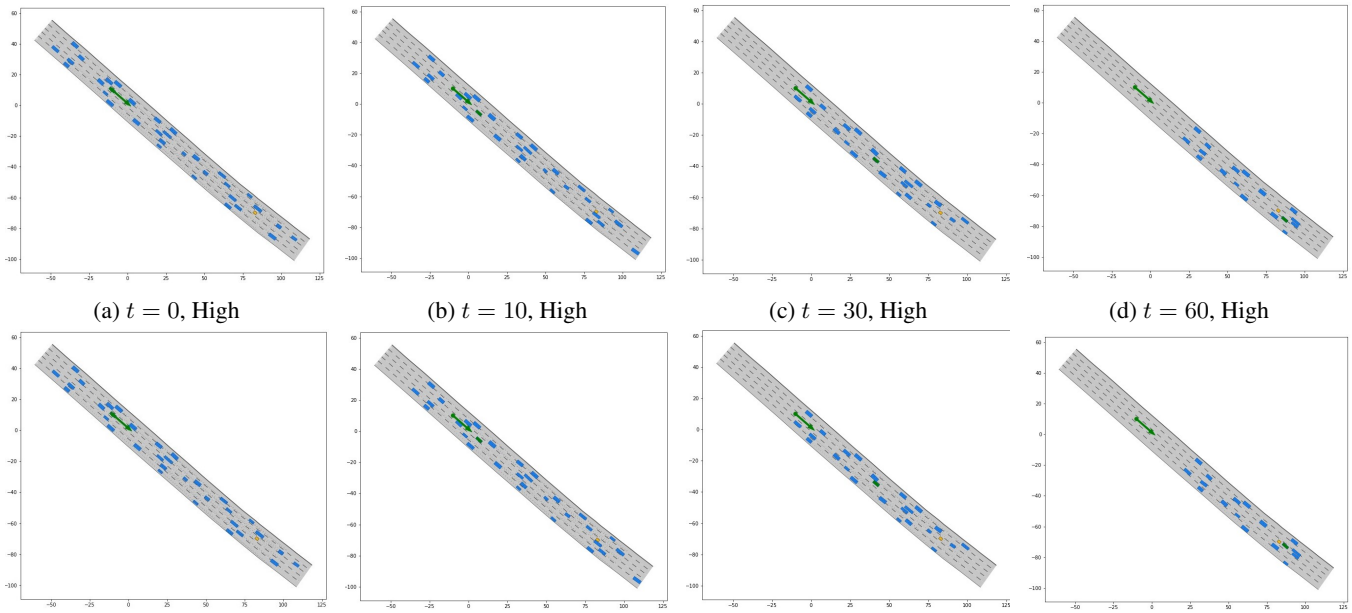
As shown in the figure, for cases with low jerk weight, the initial jerk of the vehicle is much steeper and is followed by a phase with negative jerk. In contrast, for cases with high jerk punishment, the increase in jerk is much lower in magnitude. This is

5.6. Computational Complexity

In the optimization problem, we used OSQP,[15] an operator splitting solver for quadratic programs. It is an algorithm that has no requirement on the problem data. In comparison with the interior-point method, which requires $O(\sqrt{n}log(1/\epsilon)$iterations to solve a problem of n numbers of variables with a precision of ϵ, OSQP is often 10 times faster than the interior-point method[15]. Its robustness and its efficiency make it a popular method for solving convex quadratic programs. In practice, our algorithms achieves an average runtime of 3.43×10^{-3} seconds over an optimization task of 60 time steps.

6. DISCUSSION

This paper applied convex optimization to the vehicle trajectory planning for self-driving. We demonstrated how the CSE 203 class materials can be applied to a real world problem,



(a) $t = 0$, High (b) $t = 10$, High (c) $t = 30$, High (d) $t = 60$, High

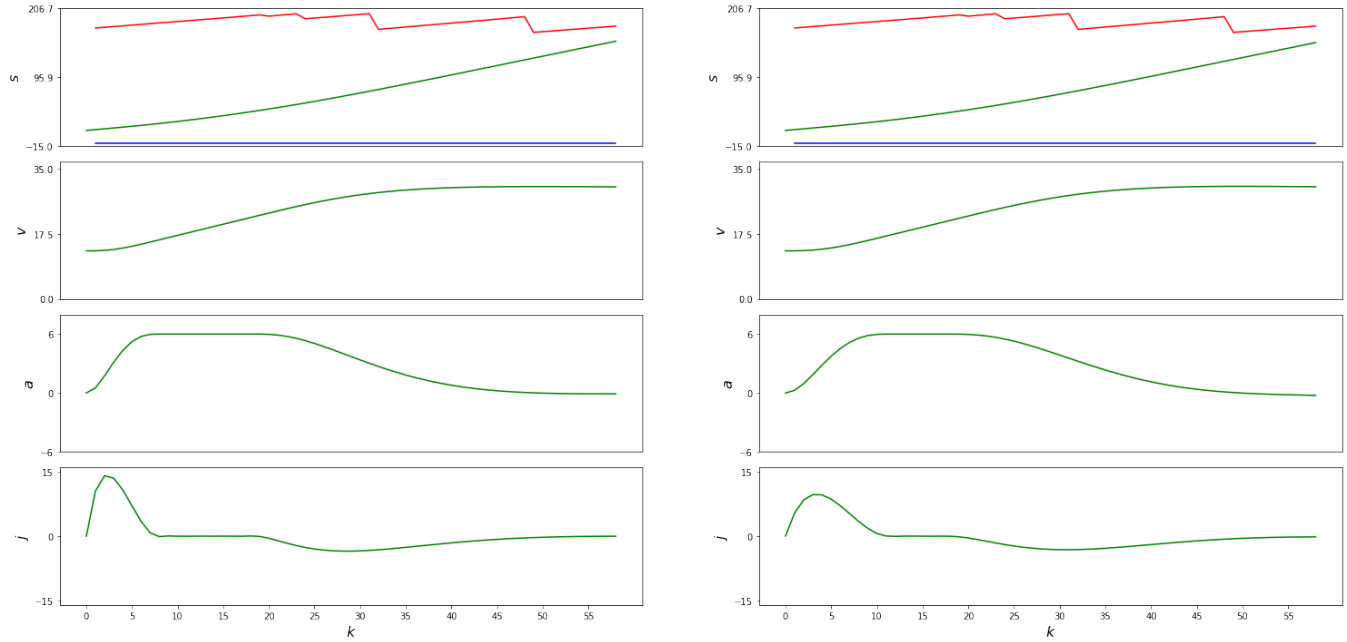
(e) $t = 0$, Low (f) $t = 10$, Low (g) $t = 30$, Low (h) $t = 60$, Low

Fig. 9: Planned motion of ego-vehicle with high and low-velocity punishment. The entire duration of the experiment is 60 timesteps, among which 4 are selected as shown above. As shown in the graph, the ego vehicle (green) accelerates to catch up with the vehicle in front of its lane. In particular, notice that to conform to the expected velocity, the ego vehicle with high-velocity priority has chosen to change lane ($t = 60$), while the one with low priority remains in its lane.

experimented to relax given condition, and inspect how different parameter changes impact on the numerical solution. Further, we observed the difference in numerical solution but the target of the vehicle movement, collision avoidance, is still achieved. Our solution created a sequence of vehicle controls to enable self-driving to avoid collisions while minimizing the discomfort caused by evasive deceleration using convex optimization. We adopted ADMM based OSQP because of it ability to solve the optimization problem without the requirement of PD condition of the objective function.

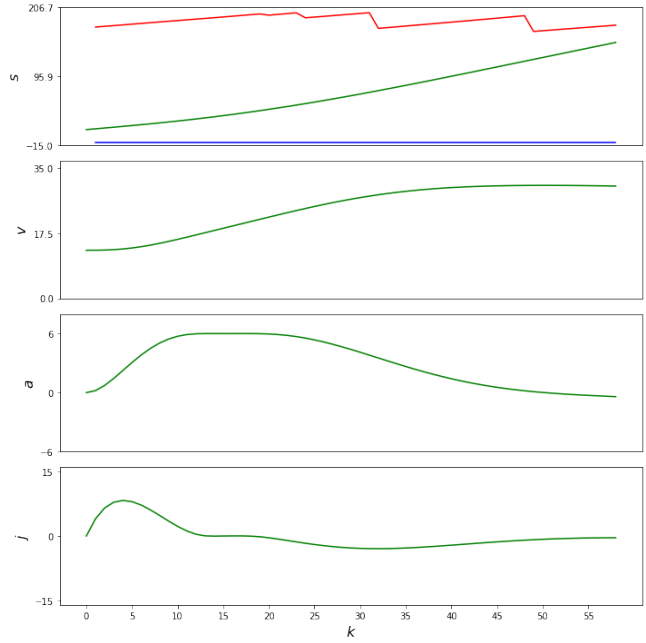
7. ACKNOWLEDGEMENTS

We thank Professor Chung-Kuan Cheng for teaching us the mechanism of duality and utility of convex optimization with passion and humbleness. We also thank TAs who diligently answered our questions and comments throughout the course with patience.

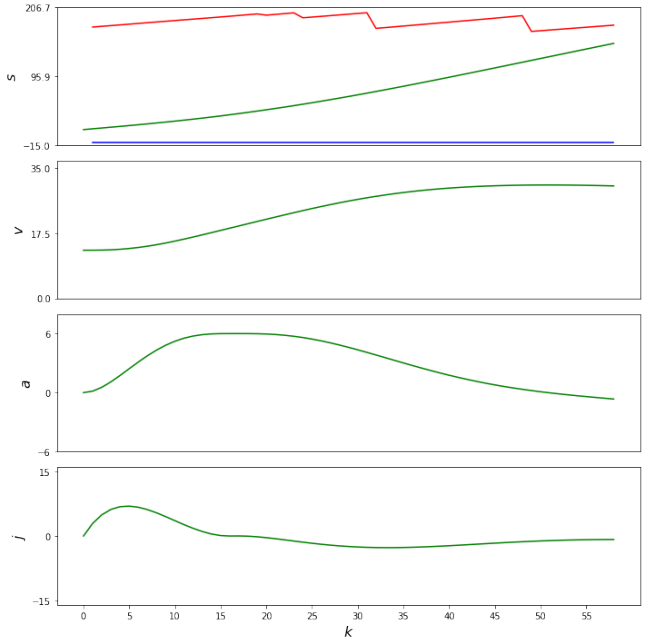


(a) $w_j = 0.1$

(b) $w_j = 0.5$



(c) $w_j = 1$



(d) $w_j = 2$

Fig. 10: The state vectors across time with varying jerk weights. Each subgraph depicts the value of the longitudinal position s, the velocity v, the acceleration a, and the jerk j across time given its respective jerk weight w_j. The red line and blue line for s refers to the constraints $s_{min}^{(t)}$ and $s_{max}^{(t)}$ derived from the other vehicles and is thus shared across graphs. For cases with low jerk weight e.g. (a) (b), the initial jerk of the vehicle is much steeper and is followed by a phase with negative jerk starting at around $t = 20$. In contrast, for cases with high jerk punishment e.g. (c) (d), the increase in jerk is much lower in magnitude.

8. REFERENCES

[1] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[2] Oussama Khatib. The potential field approach and operational space formulation in robot control. In *Adaptive and learning systems*, pages 367–377. Springer, 1986.

[3] Huahua Liu, Hugh HT Liu, Cheng Chi, Yawei Zhai, and Xingqun Zhan. Navigation information augmented artificial potential field algorithm for collision avoidance in uav formation flight. *Aerospace Systems*, 3(3):229–241, 2020.

[4] Jiayi Sun, Jun Tang, and Songyang Lao. Collision avoidance for cooperative uavs with optimized artificial potential field algorithm. *IEEE Access*, 5:18382–18390, 2017.

[5] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2020.

[6] Roberto Lampariello, Duy Nguyen-Tuong, Claudio Castellini, Gerd Hirzinger, and Jan Peters. Trajectory planning for optimal robot catching in real-time. In *2011 IEEE International Conference on Robotics and Automation*, pages 3719–3726. IEEE, 2011.

[7] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[8] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behcet Acikmese. Convex optimization for trajectory generation. *arXiv preprint arXiv:2106.09125*, 2021.

[9] Yuanqi Mao, Michael Szmuk, Xiangru Xu, and Behçet Açikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *arXiv preprint arXiv:1804.06539*, 2018.

[10] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. Gusto: guaranteed sequential trajectory optimization via sequential convex programming. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6741–6747. IEEE, 2019.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[12] Philip E Gill and Daniel P Robinson. A primal-dual augmented lagrangian. *Computational Optimization and Applications*, 51(1):1–25, 2012.

[13] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2014.

[14] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[15] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[16] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[17] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE, 2017.

[18] Christian Pek and Matthias Althoff. Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1447–1454. IEEE, 2018.

[19] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, 2017.

[20] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.