# ESE 599 Masters Independent Study
# Learning Model Predictive Control on F1tenth Race Car

Yuwei Wang[1], Baihong Zeng[2], Instructor: Rahul Mangharam[3]
yuwwang[1], baihongz[2], rahulm[3]@seas.upenn.edu

***Abstract*—Achieving minimal lap time offers great advantage in wining a car race. In this project, we aim to minimize the lap time in an online manner and an algorithm based on Learning Model Predictive Control (LMPC) [1] is implemented to do online iterative trajectory optimization which provides a minimal lap time. Both kinematic and dynamics models are used as system identification strategies. This algorithm is written in C++ and the MPC is solved using OSQP solver. Afterwards, it is tested on F1tenth RC car platform and the simulation result shows that the lap time decreases and converges to a minimum lap time racing policy.**

## I. Introduction

In this project, our goal is to have a minimum lap time within the physically limitations, which usually requires tracking an optimal trajectory and pushing all vehicle components into their physical limits. People are trying to achieve this goal using different strategies and they will be discussed in section I-A and we try to solve this problem based on Model Predictive Controller (MPC).

As an advanced optimization-based control strategy, MPC has achieved remarkable success in recent decades. It is widely used for trajectory tracking as it has advantage over time-varying Linear Quadratic Regulator (LQR) in that it takes into account constraints on states and inputs. It can also be used as an online trajectory planner by tailoring the cost and constraints to suit specific applications. The limitation of MPC on the other hand is that it only gives local optimality as it only solves a finite-horizon problem with a terminal cost that approximates the infinite-horizon cost. However, we would like to obtain a racing policy that is globally optimal in the sense of minimum lap time. Therefore, we adopted the learning MPC strategy which formulates the racing problem as an iterative control task. The controller is not based on a pre-computed racing line and it learns from previous trajectories which minimizes the lap time. The car learns to improve its racing policy iteratively online as it completes more and more laps, until the policy converges to a minimum lap time. In particular, the closed-loop trajectories at each lap are stored and used to systematically update the controller for the next lap.

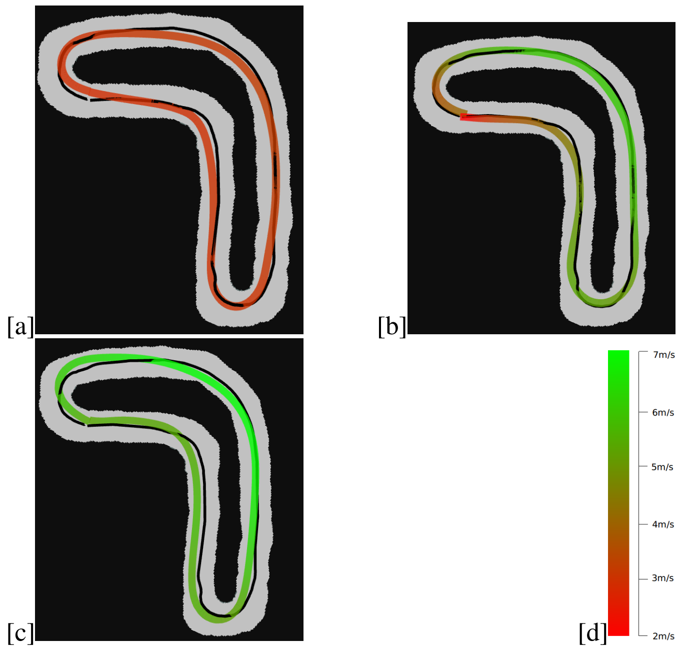


Fig. 1. Trajectory evolution and the corresponding speed profile at lap 3 [a], 8 [b], 20 [c], respectively.

Initial Sample Safe Set is collected using a path following controller. Starting from Lap 3, control policies are generated by the MPC described in the upcoming section. Every time a lap is completed, a cost-to-go value defined as the remaining time to complete the lap is updated for each collected sample. Stability and recursive feasibility of MPC is gauranteed by enforcing the terminal state to land within a convex hull of a selected subset of collected samples from previous laps. The MPC manages to minimize the cost-to-go for the terminal state at each run, and is solved with OSQP solver at 20Hz.

Two authors have equally contribution to this of the two authors.

### A. Related Work

Some methods are proposed about how to generate optimal trajectory using different machine learning and optimization techniques including Bayesian Optimization and Covariance Matrix Adaptation Evolution Strategy [2] [3]. However, these methods are not complete in fulfilling our goal since they

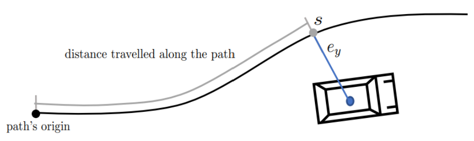


Fig. 2. Progress of a car along centerline $s$ [1]

didn't complete the part on how to track trajectories perfectly.

Model Predictive Contouring Control [4] is a variant of MPC designed for autonomous racing. The proposed method combines both planning and tracking in one nonlinear optimization problem following the ideas of contouring control.

Ugo Rosolia's paper: Learning How to Autonomously Race a Car: a Predictive Control Approach [1] talks about how to formulate the LMPC problem, giving us the foundation of implementing this approach. This project is inspired and based on this paper and his previous work.

## II. OBJECTIVE

This algorithm should be able to minimize lap time when the outer environment stays the same (i.e. fixed shape, no dynamic obstacles). The objectives are summarized as below:

- Obtain a dynamically feasible trajectory which offers minimum lap time.
- Be able to converge with the presence of static obstacles.

## III. METHODOLOGY

The state and control input vectors are stated as:

$$\mathbf{x} = [x, y, \psi, v_x, \dot{\psi}, \beta]^T \text{ and } \mathbf{u} = [a, \delta]^T \tag{1}$$

where $x$, $y$ are position in world coordinate, $\psi$ is yaw angle, $v_x$ is velocity in car's x-direction, $\dot{\psi}$ is yaw rate, and $\beta$ is the slip angle. $a$ and $\delta$ are acceleration and steering commands, respectively. $s$ is a feature representing the progress along the centerline track as shown in Figure 2.

### A. Controller Design

This part is similar to what Rosolia states in [1]. Historical data from previous laps are used to construct a convex terminal set $\mathbf{D}_l^j(x)$ and terminal cost $\mathbf{J}_l^j(x)\lambda$. Afterwards, we exploit these quantities to design the controller.

### B. Stored Data

One iteration is defined as one successful lap. We store both the states $\mathbf{x}$ and its corresponding control inputs $\mathbf{u}$. This is also called closed-loop trajectories.

$$\begin{aligned}\mathbf{u}^j &= \left[u_0^j, \ldots, u_{T^j}^j\right] \\ \mathbf{x}^j &= \left[x_0^j, \ldots, x_{T^j}^j\right]\end{aligned} \tag{2}$$

Superscript $j$ denotes $j$th iteration. $T^j$ denotes the time at which the closed-loop system reached the terminal set, i.e. $x_{T^j} \in \mathcal{X}_{\mathcal{F}}$, where $\mathcal{X}_{\mathcal{F}}$ represents the states beyond finishing line [1].

### C. Local Convex Safe Set

This part is based on the same section in Rosolia's paper [1] with more explanation. A local convex safe set is a set of states around a local point $x$. It is constructed using a subset of the stored data points. We want to find the K-nearest neighbors with respect to the progress of the selected local point. $D_l^j(x)$ represents the K-nearest neighbors to $x$ from the $l$th to the $j$th iteration are collected as shown in Figure 3.

$$D_l^j(x) = \left[x_{t_1^{l,*}}^l, \ldots, x_{t_K^{l,*}}^l, \ldots, x_{t_1^{j,*}}^j, \ldots, x_{t_K^{j,*}}^j\right] \tag{3}$$

which is used to define the local convex safe set around $x$.

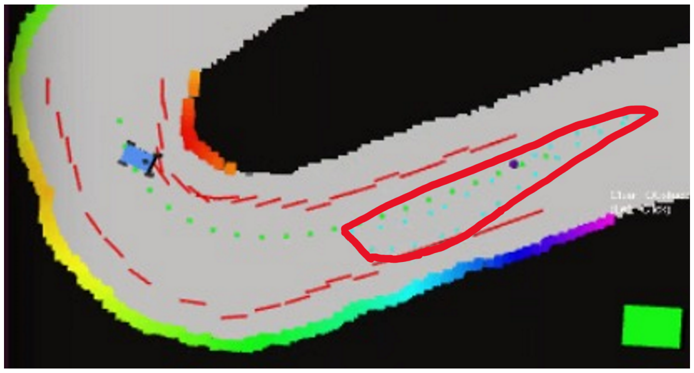


Fig. 3. local convex safe set $D_l^j(x)$ shown as light blue dots, which is the K-nearest points around $x$ from the $l$th to the $j$th iteration. Red Polygon is the corresponding convex hull $\mathcal{CL}_l^j(x)$

$$\begin{aligned}\mathcal{CL}_l^j(x) = \Big\{&\bar{x} \in R^n : \exists \boldsymbol{\lambda} \in R^{K(j-l+1)} \\ &\boldsymbol{\lambda} \geq 0, 1\boldsymbol{\lambda} = 1, D_l^j(x)\boldsymbol{\lambda} = \bar{x}\Big\}\end{aligned} \tag{4}$$

$\mathcal{CL}_l^j(x)$ shown in Figure 3 is a linear combination of $D_l^j(x)$, meaning $\mathcal{CL}_l^j(x)$ is the convex hull of $D_l^j(x)$. $\bar{x}$ represents a new state generated through the linear combination and $\bar{x}$ is within $\mathcal{CL}_l^j(x)$. Then, we can construct an approximation to the minimum cost-to-go $J_l^j(\bar{x}, x)$ over the local convex safe set $\mathcal{CL}_l^j(x)$ around $x$ using a subset of stored data.

$$\begin{aligned}J_l^j(\bar{x}, x) = &\min_{\boldsymbol{\lambda}} \mathbf{J}_l^j(x)\boldsymbol{\lambda} \\ &\text{s.t} \quad \boldsymbol{\lambda} \geq 0, 1\boldsymbol{\lambda} = 1, D_l^j(x)\boldsymbol{\lambda} = \bar{x}\end{aligned} \tag{5}$$

where $\boldsymbol{\lambda} \in R^{k(j-l)}, 1$ is a row vector of ones and the row vector

$$\mathbf{J}_l^j(x) = \Big[J^l_{t_1'^*\to T^l}\left(x^l_{t_1',*}\right),\ldots,J^l_{t_M^{l,*}\to T^l}\left(x^l_{t_M^{l,*}}\right),\ldots \\ J^j_{t_1^{j,*}\to T^j}\left(x^j_{t_1^{j,*}}\right),\ldots,J^j_{t_M^{j,*}\to T^j}\left(x^j_{t_M^{j,*}}\right)\Big) \tag{6}$$

collects the cost-to-go associated with the K-nearest neighbors to $x$ from $l$th to $j$th iteration. The cost-to-go $J^j_{t\to T^j}\left(x^j_t\right) = T^j - t$ represents the time to drive the vehicle from $x^j_t$ to the finish line along the $j$th trajectory. We underline that the cost-to-go is computed after completion of the $j$th iteration [1].

### D. LMPC Formulation Design

MPC design is shown in Equ 7 and Equ 8. At each time $t$ of the $j$th iteration, the controller solves this finite time optimal control problem. Rosolia [1] only considers cost-to-go $J$, i.e. time is the only factor in cost, modeling it as a minimum time optimization problem. We consider two more aspects into the objective function formulation:

- Control effort $u$. Control effort should be minimized to save energy / fuel during a race as well as to achieve smooth maneuvers.
- Slack variable $s_k, s_t$. Slack variable should be minimized so that the soft constraints violation is minimum.
- Cost-to-go $J$. In this problem, cost-to-go is equivalent to time, meaning lap time is minimized.

$$J^{\mathrm{lmpc},j}_{t\to t+N}\left(x^j_t, z^j_t\right) = \min_{\mathbf{U}^j_t,\boldsymbol{\lambda}^j_t} \sum_{k=t}^{t+N-1} \mathbf{u}^{j\,T}_k \mathbf{R}\mathbf{u}^j_k + \mathbf{s_k}^T\mathbf{q_s}\mathbf{s_k} \\ +\mathbf{s_t}^T\mathbf{q_s}\mathbf{s_t} + \mathbf{J}^{j-1}_l\left(z^j_t\right)\boldsymbol{\lambda}^j_t \tag{7}$$

$$\text{s.t. } x^j_{t|t} = x^j_t \tag{8a}$$
$$x^j_{k+1|t} = A^j_{k|t}x^j_{k|t} + B^j_{k|t}u^j_{k|t} + C^j_{k|t} \tag{8b}$$
$$F_x x_{k|t} < b_x \tag{8c}$$
$$\boldsymbol{\lambda}^j_t \geq 0, 1\boldsymbol{\lambda}^j_t = 1, D^{j-1}_l\left(z^j_t\right)\boldsymbol{\lambda}^j_t = x^j_{t+N|t} \tag{8d}$$
$$x^j_{k|t} \in \mathcal{X}, u^j_{k|t} \in \mathcal{U} \tag{8e}$$
$$\forall k = t,\cdots,t+N-1 \tag{8f}$$
$$s_k \geq 0, s_t \geq 0 \tag{8g}$$

where $\mathbf{U}^j_t = \left[u^j_{t|t},\ldots,u^j_{t+N-1|t}\right] \in R^{d\times N}, \boldsymbol{\lambda}^j_t \in R^{(j-l+1)K}$.

In this project, we used OSQP-Eigen as our solver and all the MPC problem has to be casted to a standard QP problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^THx + q^Tx \\ \text{subject to} & lower \leq A_c x \leq upper \end{array} \tag{9}$$

The formulation example can be found in [5]. We reformulated Equ 7 and 8 into the standard format and they are explained in the following sections. The matrix form of $H$, $q$, $lower$, $upper$, $A_c$, $x$ are available in Appendix $A$ to $F$.

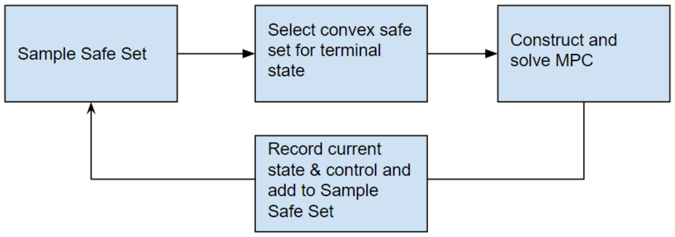

Sample Safe Set
Select convex safe set for terminal state
Construct and solve MPC
Record current state & control and add to Sample Safe Set

Fig. 4. LMPC Workflow

### E. Objective Function

*1) Quadratic Cost:* In Equ 7, control input $u$ are considered since we also want to reduce the energy used. We also want to minimize the violation of the soft constraints that position slack variable $s_k$ and terminal set slack variable $s_t$ are considered as part of the quadratic cost as well.

*2) Linear Cost:* Equ 5 is the cost-to-go representing the time needed to go from terminal state to goal. It is grouped into Equ 7.

### F. Constraints

In the above finite horizon optimization problem, Equ 8a, 8b, 8c, 8e represent the state, dynamics, boundary, input constraints. While Equ 8d enforces $x^j_{t+N|t}$ into the local safe set $\mathcal{CL}^j_l(x)$. Optimal solution to Equ 7 at time $t$ of the $j$th iteration is used to compute $z^j_t$ in Equ 8d:

$$z^j_t = \begin{cases} x^{j-1}_N & \text{If } t = 0 \\ x^j_{t+N|t-1} & \text{Otherwise} \end{cases} \tag{10}$$

The above vector $z^j_t$ represents a candidate terminal state for the planned trajectory of the LMPC at time $t$. First, we initialize the candidate terminal state $z^j_0$ using the $(j-1)$th trajectory. Afterwards, we update the vector $z^j_t$ using the terminal state solved in previous time step $x^j_{t+N|t-1}$.

Finally, we apply the first element of $u^j_t = u^{j,*}_{t|t}$ to the system in Equ 8b. This problem is repeated at time $t+1$, based on the new state $x_{t+1|t+1} = x^j_{t+1}$ [1].

*1) Dynamics Constraints from Eq 8b, code section: get_linearized_dynamics(), solve_MPC():* The nonlinear vehicle dynamics we used is described in details in section IV. In order to formulate the constraints on dynamics as part of the QP, at each run we linearize the dynamics around states and inputs obtained from previous timestep's MPC solution into a Affine-Time-Varying Model.

*2) Position Constraints and its Relaxation from Equ 8c, code section: Track.h, solve_MPC():* As from the paper, to make sure $X$ is a convex set and every state $x_t$ is within the set, we need half-space constraint $F_x x < b_x$ as shown below:

$$\begin{aligned} x_t &\in \mathcal{X} = \{x \in R^n : F_x x \leq b_x\} \\ u_t &\in \mathcal{U} = \left\{u \in R^d : F_u u \leq b_u\right\}, \forall t \geq 0 \end{aligned} \tag{11}$$

Our approach constraints every position $[x, y]$, i.e. $x_{t0}, x_{t1}$ between the boundary of the track, while the $b_x$ of other elements of the states are set as $\infty$. We approximate centerline of the track using cubic spline, for each $x^j_{k-1}$, we find the closest centerline point, and its tangent line. By moving this tangent line parallelly until one of the parallel line intersect with the track boundary, we find the width of the track at this specific progress $s$. Then we construct the position constraint for $x^j_k$ from the closest center point of $x^j_{k-1}$ using this approach. This approach will be detailed in the coming paragraphs and is illustrated in Figure 5.

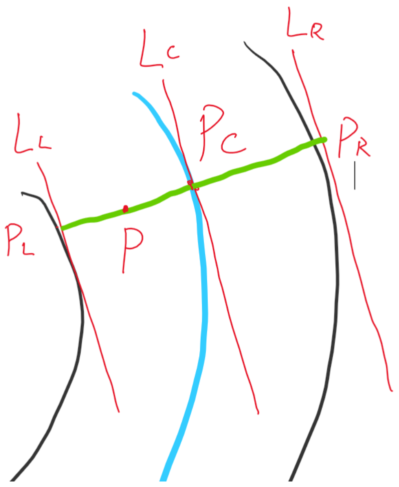


Fig. 5. Position Constraint for one state, black lines are track boundaries, $P$ is the position of $x^j_{k-1}$, where $P_c$ is the closest centerline point of $P$.

Centerline of the track $L_c$ is approximated using two cubic splines:

$$\begin{aligned} X(s) &= as^3 + bs^2 + cs + d \\ Y(s) &= as^3 + bs^2 + cs + d \end{aligned} \tag{12}$$

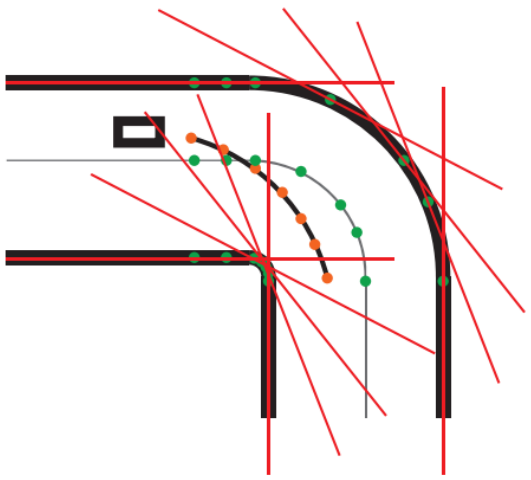


Fig. 6. Resulting halfspaces (red lines) of the whole horizon, approximating the track. The green points on the center line correspond to the closest point on the discrete center line, for each point in the horizon. The points on the borders are the projection of the center line points on the borders [4].

And the first derivatives of the above parameter functions are:

$$\begin{aligned} \frac{dx}{ds} &= 3as^2 + 2bs + c \\ \frac{dy}{ds} &= 3as^2 + 2bs + c \end{aligned} \tag{13}$$

Normally, $L_c$ can be expressed as $y(x)$:

$$y = kx + b \tag{14}$$

Where $k = \frac{dy}{dx}$. The other way to express a line is:

$$ax + by + c = 0 \tag{15}$$

By multiplying $\frac{dx}{ds}$ on each term, $L_c$ in equ. 15 form turns into:

$$L_c : -\frac{dy}{ds}x + \frac{dy}{ds}y = b \cdot \frac{dx}{ds} \tag{16}$$

The green line perpendicular to $L_c$ is:

$$\frac{dy}{ds}x - \frac{dx}{ds}y = b_1 \tag{17}$$

By propagating along green line, the intersecting point $(P_L, P_R)$ between track boundaries and $L_L, L_R$ is found, where $L_L, L_R$ are having same slope as $L_c$. They have the form:

$$\begin{aligned} L_L &: -\frac{dy}{ds}x + \frac{dy}{ds}y = c_1 \\ L_R &: -\frac{dy}{ds}x + \frac{dy}{ds}y = c_2 \end{aligned} \tag{18}$$

With that, we can construct our position constraint which clamps the position within boundary:

$$\begin{aligned}\min\left(c_1, c_2\right) \leq -\frac{dy}{ds}x + \frac{dy}{ds}y \\ -\frac{dy}{ds}x + \frac{dy}{ds}y \leq \max\left(c_1, c_2\right)\end{aligned} \tag{19}$$

Each state has one such constraint. There are $N$ such position constraints corresponding to $N$ $x_k$ states as shown in Figure 6. In the case of sharp turns, such hard constraint may result in infeasible solution of MPC. Thus, we decided to bring in slack variable $s_k$ to relax this constraint, where $s_k \geq 0$. Equ 22 is transformed into:

$$\begin{aligned}\min\left(c_1, c_2\right) \leq -\frac{dy}{ds}x + \frac{dy}{ds}y + s_k \leq \infty \\ -\infty \leq -\frac{dy}{ds}x + \frac{dy}{ds}y - s_k \leq \max\left(c_1, c_2\right)\end{aligned} \tag{20}$$

$\infty$ and $-\infty$ are just used to fill up *lower* and *upper* matrix.

*3) Control Input Constraints from Equ 8e, code section: solve_MPC():*

$$\begin{aligned}dec_{max} \leq a \leq acc_{max} \\ -steer_{max} \leq \delta \leq steer_{max}\end{aligned} \tag{21}$$

*4) Speed Constraints from Equ 8e, code section: solve_MPC():*

$$-speed_{max} \leq v \leq speed_{max} \tag{22}$$

*5) Slack Variable Constraints from Equ 8g, code section: solve_MPC():*

$$\begin{aligned}0 \leq s_k \leq \infty \\ 0 \leq s_t \leq \infty\end{aligned} \tag{23}$$

*6) Lambda Constraints from Equ 8d, code section: solve_MPC():*

$$0 \leq \boldsymbol{\lambda}_t^j \leq \infty \tag{24}$$

*7) Terminal State Constraints from Equ 8d, code section: solve_MPC():* Similar to the section III-F2, the terminal state constraints are relaxed as well. The original hard constraint is:

$$D_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j = x_{t+N|t}^j \tag{25}$$

With the introduction of slack variable $s_t$, this constraint is relaxed to:

$$-s_k \leq D_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j - x_{t+N|t}^j \leq s_k \tag{26}$$

To comply with the format of OSQP, Equ 26 is formatted as:

$$\begin{aligned}0 \leq s_k + D_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j - x_{t+N|t}^j \leq \infty \\ -\infty \leq -s_k + D_l^{j-1}\left(z_t^j\right)\boldsymbol{\lambda}_t^j - x_{t+N|t}^j \leq 0\end{aligned} \tag{27}$$

*8) Sum of Lambda Constraints from Equ 8d:* An requirement for $\mathcal{CL}_l^j(x)$ to be the convex hull of $D_l^j(x)$.

$$1 \leq \mathbf{1}\boldsymbol{\lambda}_t^j \leq 1 \tag{28}$$

## IV. Vehicle Model

### A. Kinematic Model

The kinematic single-track models a road vehicle with only two wheels, where the front and rear wheel pairs are each lumped into one wheel. The kinematic single-track model further does not consider any tire slip, so that the velocity vector $v$ at the center of the rear axle is always aligned with the link between the front and rear wheel [6]. The kinematic dynamics is as follows:

$$\dot{x} = vcos(\psi), \dot{y} = vsin(\psi), \dot{v} = a, \dot{\psi} = \frac{vtan(\delta)}{l} \tag{29}$$

### B. Dynamic Model

Since the kinematic single-track model does not consider tire slip, important effects such as understeer or oversteer are not considered. When the vehicle is driving close to its physical capabilities, those effects are dominant. The extension on top of kinematic model is a dynamic model known as the bicycle model. Perform planning of evasive maneuvers closer to physical limits requires the bicycle model [6]. We also consider the load transfer of the vehicle due to longitudinal acceleration $a$ such that the vertical forces on the front and rear axis $F_{z,f}$ and $F_{z,r}$ become

$$F_{z,f} = \frac{mgl_r - mah_{cg}}{l_r + l_f}, \quad F_{z,r} = \frac{mgl_f + mah_{cg}}{l_r + l_f} \tag{30}$$

These forces are inserted into the derivation of the equations for the slip angle (at the center of gravity) and the yaw rate. The dynamics is given by the following equations:

$$\begin{aligned}\dot{x} =& vcos(\psi + \beta), \quad \dot{y} = vsin(\psi + \beta), \quad \dot{\psi} = \dot{\psi}, \quad \dot{v} = a, \\ \ddot{\psi} =& \frac{\mu m}{I_z\left(l_r + l_f\right)}\left(l_f C_{S,f}\left(gl_r - ah_{cg}\right)\delta\right. \\ &+ \left(l_r C_{S,r}\left(gl_f + ah_{cg}\right) - l_f C_{S,f}\left(gl_r - ah_{cg}\right)\right)\beta \\ &\left.-\left(l_f^2 C_{S,f}\left(gl_r - ah_{cg}\right) + l_r^2 C_{S,r}\left(gl_f + ah_{cg}\right)\right)\frac{\dot{\psi}}{v}\right), \\ \dot{\beta} =& \frac{\mu}{v\left(l_r + l_f\right)}\left(C_{S,f}\left(gl_r - ah_{cg}\right)\delta - \left(C_{S,r}\left(gl_f + ah_{cg}\right) + \right.\right. \\ & C_{S,f}\left(gl_r - a_{\text{long}}h_{cg}\right)\beta + \left(C_{S,r}\left(gl_f + ah_{cg}\right)l_r\right. \\ & - C_{S,f}\left(gl_r - ah_{cg}\right)l_f\frac{\dot{\psi}}{\eta} - \dot{\psi}\end{aligned}$$

| | vehicle parameter |
|---|---|
| name | symbol |
| vehicle length | $l$ |
| vehicle width | $w$ |
| total vehicle mass | $m$ |
| moment of inertia about z axis | $I_z$ |
| distance from COG to front axle | $l_f$ |
| distance from COG to rear axle | $l_r$ |
| COG height of total mass | $h_{cg}$ |
| cornering stiffness coefficient | $C_{S,f}$ |
| friction coefficient | $\mu$ |

## V. Results

### A. Testing Simulator Platform

The F1tenth AV platform is equipped with all the essential components we need to validate, test, and employ any new algorithms and hardware. It is a 1/10th scale car powered by a LiPo battery and it communicates with joystick through Wi-Fi. A 2D Hokuyo Lidar provides orientation and distance information of its surrounding obstacles and a Vedder Electronic Speed Controller (VESC) precisely controls the speed. All computation are performed on a NVIDIA Jetson TX2 board. Different kinds of camera is optional for better perception ability. Our original plan is to test the algorithm on a real F1tenth race car but due to COVID-19 outbreak, the accessibility of real car is limited so we can only perform experiment on the corresponding high fidelity simulator.

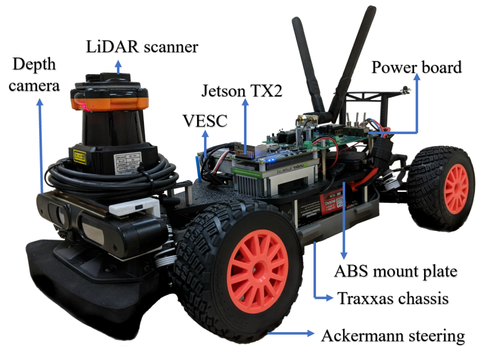


Fig. 7. F1tenth Platform Overview

### B. Experiment Result

All simulator and car parameters are set to the default values in F1tenth simulator repository [7]. We stored two initial laps data with 1m/s running pure pursuit in advance to start the learning process. The initial lap time is 44.5s and the converged lap time is 7.5s which is shown in Table I.

When the cost of acceleration being high, the car tends to run with a steady speed, which limits its top speed at straight line and the average speed is low. Figure 8 shows the result where the learning rate is slower that it converges to 9.1s at 72 iters. When the cost of acceleration is high, the car is more willingly to accelerate and decelerate, meaning it drives more aggressively. The car achieved a lower lap time (7.4s) and high learning rate (convergence at 20 iters) as shown in Figure 9.

A comparison between initial naive pure pursuit centerline trajectory and optimized trajectory is shown in Figure 10. The optimized trajectory are really using the whole width of the track as it intersects with the inflated track boundary (blue dots). Also, it accelerates after the corners and decelerate before entering the corners, which is a common skills for professional human drivers. Moreover, it reaches the capped top speed (7m/s) of the simulator, meaning the learning has achieved a great result.

TABLE I
Performance in Simulation

| | Lap time: $s$ |
|---|---|
| LMPC with acceleration penalty $R_0 = 1.5$ | 8.4 |
| LMPC with acceleration penalty $R_0 = 0.1$ | 7.5 |

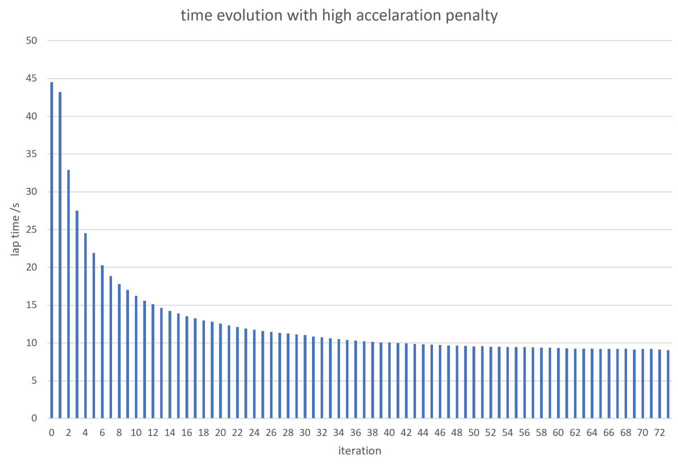


Fig. 8. Convergence of lap time with high acceleration penalty $R_0 = 1.5$

## VI. Conclusions

A Learning Model Predictive Controller (LMPC) for autonomous racing is implemented. The proposed control framework uses historical data to construct safe sets and the corresponding objective function. Multiple constraints are considered. Controller is systematically updated when a lap is completed, as a result the LMPC learns from experience to safely drive the vehicle at the limit of handling. We demonstrated the effectiveness of the proposed strategy on the F1tenth platform simulator by reducing the lap time from 44.5s to 7.5s. Experimental results show that the controller learns

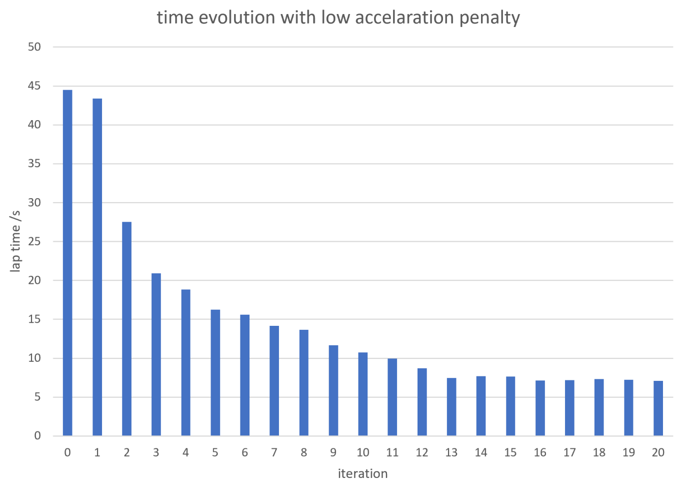


Fig. 9. Convergence of lap time with low acceleration penalty $R_0 = 0.1$

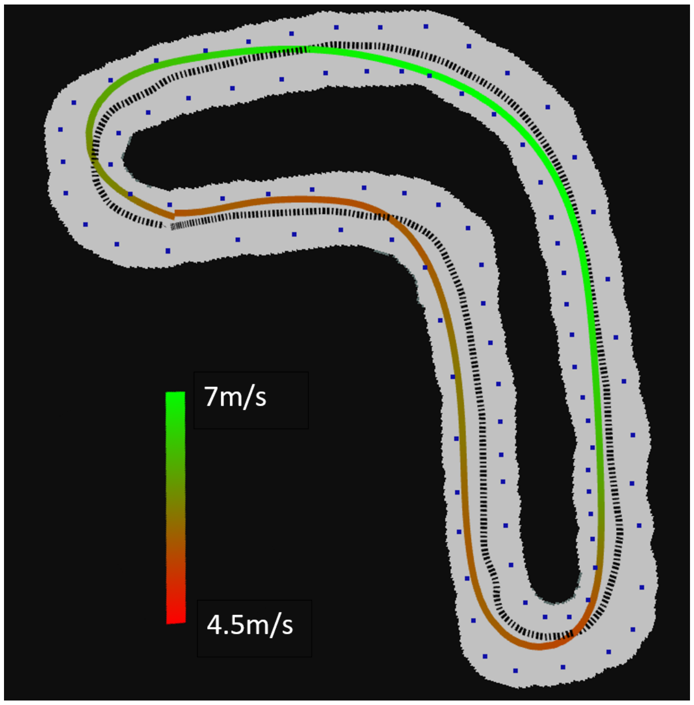


Fig. 10. Comparison between initial naive trajectory (black dot line) and optimized trajectory (colored line), with blue dot as inflated track during calculation

to drive the vehicle aggressively, which minimizes the lap time. The overall goal of this project has achieved and more testing on the real car are expected when the outer environment permits.

## APPENDIX

### A. Decision Variables: $x$

### B. Hessian Positive Definite Matrix: $H$

### C. Gradient Matrix: $q$

### D. Constraint Matrix: $A_c$

### E. lower bound matrix:$lower$

### F. upper bound matrix:$upper$

## ACKNOWLEDGMENT

We would like to thank Professor Rahul Mangharam and members in mLAB including Hongrui Zheng, Matthew O'Kelly, Kuk Jang for their expert advice on the project. Also, PostDoc Ugo Rosolia from UC Berkeley gave us help along the way as well.

## REFERENCES

[1] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, 2019.

[2] A. Jain and M. Morari, "Computing the racing line using bayesian optimization," *arXiv preprint arXiv:2002.04794*, 2020.

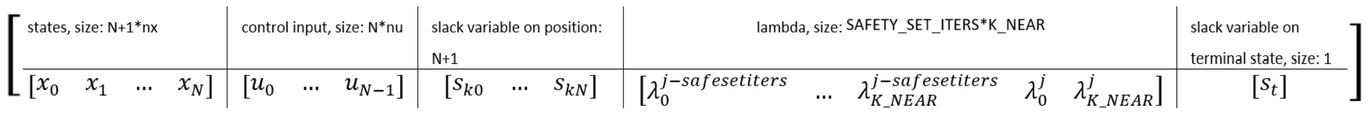
[3] M. O'Kelly, H. Zheng, A. Jain, J. Auckley, and K. Luong, "Tech report: Tunercar: A superoptimization toolchain for autonomous racing."

[4] M. M. A. Liniger, A. Domahidi, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, pp. 628–647, 2017.

[5] "Osqp-eigen." [Online]. Available: https://robotology.github.io/osqp-eigen/doxygen/doc/html/index.html

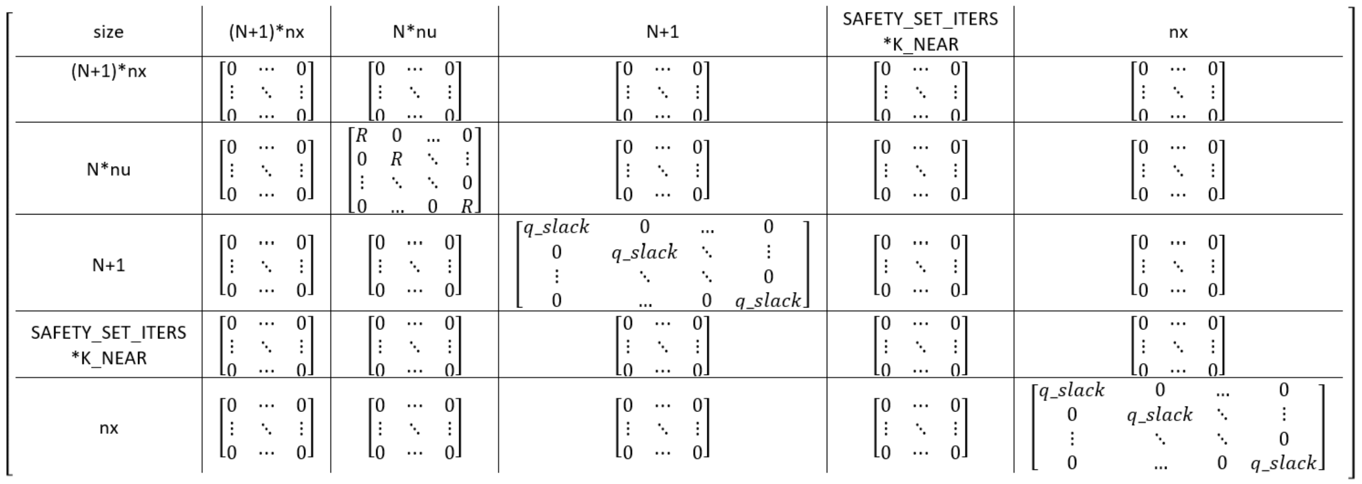
[6] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.

[7] mLAB, "F1tenth simulator repository," Dec. 2019. [Online]. Available: https://github.com/mlab-upenn/f110-fall2019-skeletons



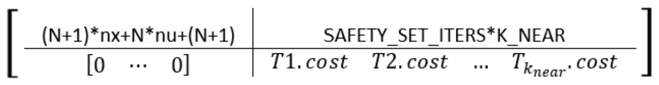
| states, size: N+1*nx | control input, size: N*nu | slack variable on position: N+1 | lambda, size: SAFETY_SET_ITERS*K_NEAR | slack variable on terminal state, size: 1 |
|---|---|---|---|---|
| $[x_0 \quad x_1 \quad \dots \quad x_N]$ | $[u_0 \quad \dots \quad u_{N-1}]$ | $[s_{k0} \quad \dots \quad s_{kN}]$ | $[\lambda_0^{j-safesetiters} \quad \dots \quad \lambda_{K\_NEAR}^{j-safesetiters} \quad \lambda_0^j \quad \lambda_{K\_NEAR}^j]$ | $[s_t]$ |

Fig. 11. Decision Variables, $x$



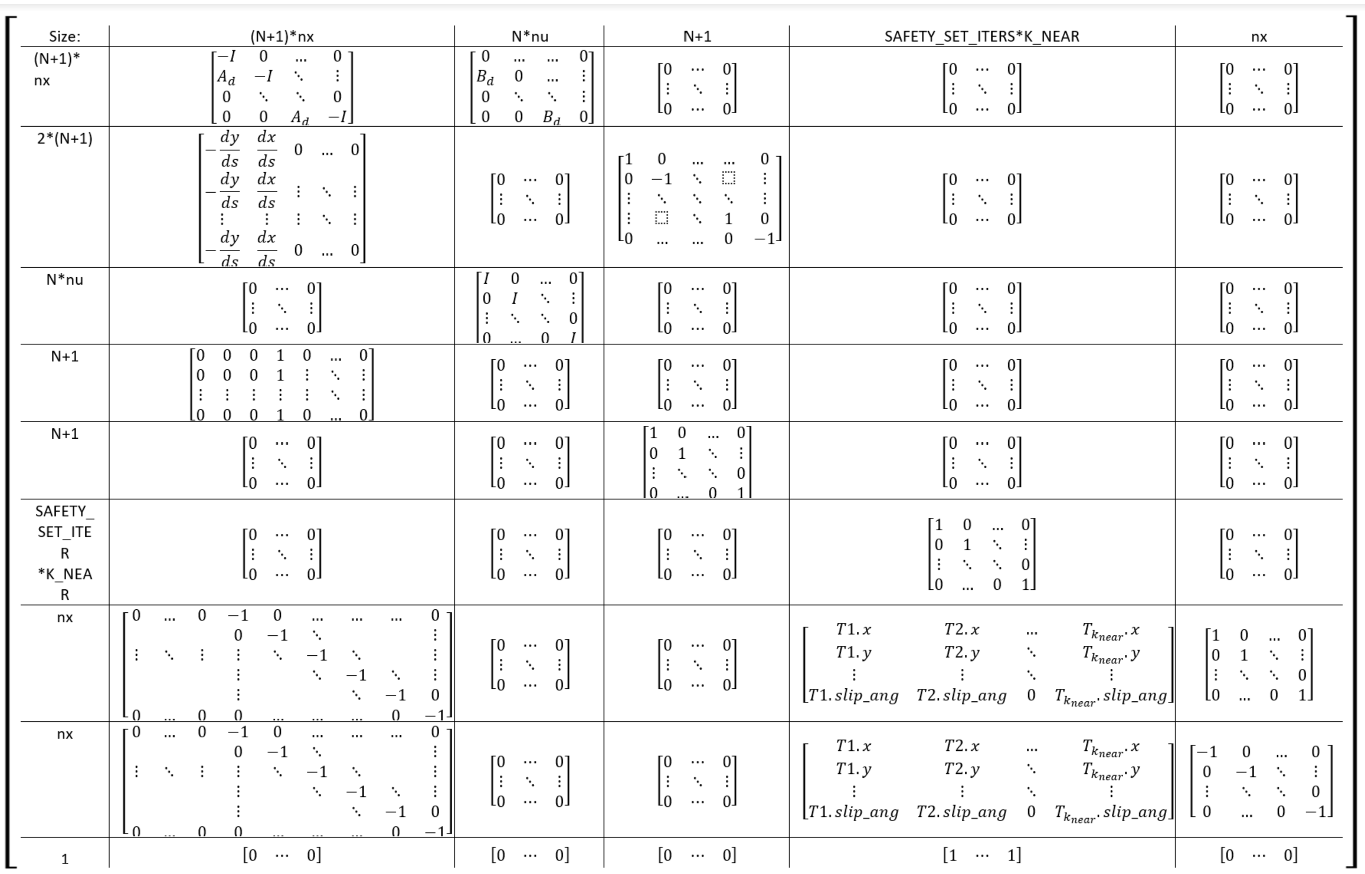
| size | (N+1)*nx | N*nu | N+1 | SAFETY_SET_ITERS *K_NEAR | nx |
|---|---|---|---|---|---|
| (N+1)*nx | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| N*nu | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}R & 0 & \dots & 0\\ 0 & R & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & R\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}q\_slack & 0 & \dots & 0\\ 0 & q\_slack & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & q\_slack\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| SAFETY_SET_ITERS *K_NEAR | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| nx | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}q\_slack & 0 & \dots & 0\\ 0 & q\_slack & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & q\_slack\end{bmatrix}$ |

Fig. 12. Hessian Matrix $H$ and its submatrix size



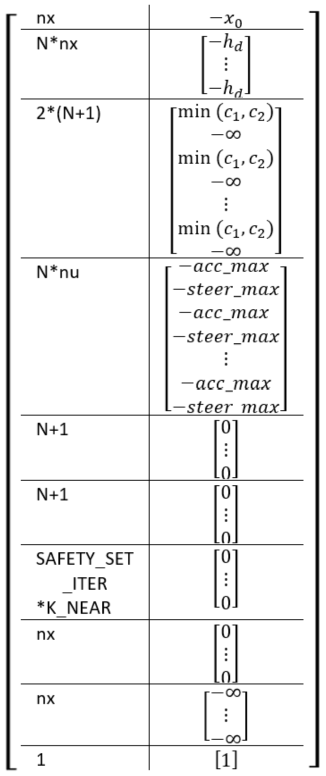
| (N+1)*nx+N*nu+(N+1) | SAFETY_SET_ITERS*K_NEAR |
|---|---|
| $[0 \quad \cdots \quad 0]$ | $T1.cost \quad T2.cost \quad \dots \quad T_{k_{near}}.cost$ |

Fig. 13. Gradient Matrix $q$ and its submatrix size



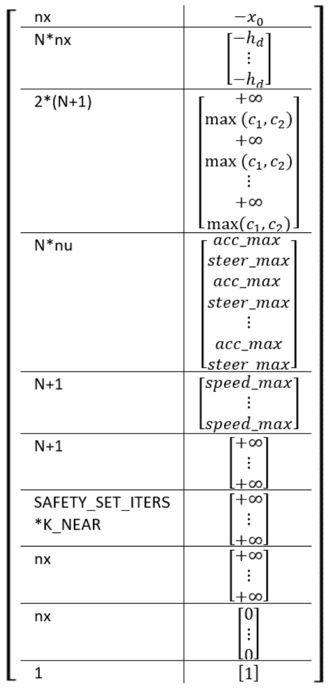
| Size: | (N+1)*nx | N*nu | N+1 | SAFETY_SET_ITERS*K_NEAR | nx |
|---|---|---|---|---|---|
| (N+1)*nx | $\begin{bmatrix}-I & 0 & \dots & 0\\ A_d & -I & \ddots & \vdots\\ 0 & \ddots & \ddots & 0\\ 0 & 0 & A_d & -I\end{bmatrix}$ | $\begin{bmatrix}0 & \dots & \dots & 0\\ B_d & 0 & \dots & \vdots\\ 0 & \ddots & \ddots & \vdots\\ 0 & 0 & B_d & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| 2*(N+1) | $\begin{bmatrix}-\frac{dy}{ds} & \frac{dx}{ds} & 0 & \dots & 0\\ -\frac{dy}{ds} & \frac{dx}{ds} & \vdots & \ddots & \vdots\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ -\frac{dy}{ds} & \frac{dx}{ds} & 0 & \dots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}1 & 0 & \dots & \dots & 0\\ 0 & -1 & \ddots & \square & \vdots\\ \vdots & \ddots & \ddots & \ddots & \vdots\\ \vdots & \square & \ddots & 1 & 0\\ 0 & \dots & \dots & 0 & -1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| N*nu | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}I & 0 & \dots & 0\\ 0 & I & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & I\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}0 & 0 & 0 & 1 & 0 & \dots & 0\\ 0 & 0 & 0 & 1 & \vdots & \ddots & \vdots\\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 1 & 0 & \dots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & 1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| SAFETY_SET_ITER*K_NEAR | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & 1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ |
| nx | $\begin{bmatrix}0 & \dots & 0 & -1 & 0 & \dots & \dots & \dots & 0\\ & & & 0 & -1 & \ddots & & & \vdots\\ \vdots & \ddots & \vdots & \vdots & \ddots & -1 & \ddots & & \vdots\\ & & & \vdots & & \ddots & -1 & \ddots & \vdots\\ & & & \vdots & & & \ddots & -1 & 0\\ 0 & \dots & 0 & 0 & \dots & \dots & \dots & 0 & -1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}T1.x & T2.x & \dots & T_{k_{near}}.x\\ T1.y & T2.y & \ddots & T_{k_{near}}.y\\ \vdots & \vdots & \ddots & \vdots\\ T1.slip\_ang & T2.slip\_ang & 0 & T_{k_{near}}.slip\_ang\end{bmatrix}$ | $\begin{bmatrix}1 & 0 & \dots & 0\\ 0 & 1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & 1\end{bmatrix}$ |
| nx | $\begin{bmatrix}0 & \dots & 0 & -1 & 0 & \dots & \dots & \dots & 0\\ & & & 0 & -1 & \ddots & & & \vdots\\ \vdots & \ddots & \vdots & \vdots & \ddots & -1 & \ddots & & \vdots\\ & & & \vdots & & \ddots & -1 & \ddots & \vdots\\ & & & \vdots & & & \ddots & -1 & 0\\ 0 & \dots & 0 & 0 & \dots & \dots & \dots & 0 & -1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}T1.x & T2.x & \dots & T_{k_{near}}.x\\ T1.y & T2.y & \ddots & T_{k_{near}}.y\\ \vdots & \vdots & \ddots & \vdots\\ T1.slip\_ang & T2.slip\_ang & 0 & T_{k_{near}}.slip\_ang\end{bmatrix}$ | $\begin{bmatrix}-1 & 0 & \dots & 0\\ 0 & -1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \dots & 0 & -1\end{bmatrix}$ |
| 1 | $\begin{bmatrix}0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\end{bmatrix}$ | $\begin{bmatrix}1 & \cdots & 1\end{bmatrix}$ | $\begin{bmatrix}0 & \cdots & 0\end{bmatrix}$ |

Fig. 14. Constraint Matrix $A_c$ and its submatrix size



| | |
|---|---|
| nx | $-x_0$ |
| N*nx | $\begin{bmatrix}-h_d\\ \vdots\\ -h_d\end{bmatrix}$ |
| 2*(N+1) | $\begin{bmatrix}\min(c_1,c_2)\\ -\infty\\ \min(c_1,c_2)\\ -\infty\\ \vdots\\ \min(c_1,c_2)\\ -\infty\end{bmatrix}$ |
| N*nu | $\begin{bmatrix}-acc\_max\\ -steer\_max\\ -acc\_max\\ -steer\_max\\ \vdots\\ -acc\_max\\ -steer\_max\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$ |
| SAFETY_SET _ITER *K_NEAR | $\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$ |
| nx | $\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$ |
| nx | $\begin{bmatrix}-\infty\\ \vdots\\ -\infty\end{bmatrix}$ |
| 1 | $[1]$ |

Fig. 15. lower bound matrix *lower* and its submatrix size



| | |
|---|---|
| nx | $-x_0$ |
| N*nx | $\begin{bmatrix}-h_d\\ \vdots\\ -h_d\end{bmatrix}$ |
| 2*(N+1) | $\begin{bmatrix}+\infty\\ \max(c_1,c_2)\\ +\infty\\ \max(c_1,c_2)\\ \vdots\\ +\infty\\ \max(c_1,c_2)\end{bmatrix}$ |
| N*nu | $\begin{bmatrix}acc\_max\\ steer\_max\\ acc\_max\\ steer\_max\\ \vdots\\ acc\_max\\ steer\_max\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}speed\_max\\ \vdots\\ speed\_max\end{bmatrix}$ |
| N+1 | $\begin{bmatrix}+\infty\\ \vdots\\ +\infty\end{bmatrix}$ |
| SAFETY_SET_ITERS *K_NEAR | $\begin{bmatrix}+\infty\\ \vdots\\ +\infty\end{bmatrix}$ |
| nx | $\begin{bmatrix}+\infty\\ \vdots\\ +\infty\end{bmatrix}$ |
| nx | $\begin{bmatrix}0\\ \vdots\\ 0\end{bmatrix}$ |
| 1 | $[1]$ |

Fig. 16. upper bound matrix *upper* and its submatrix size