Model Predictive Control: Mini-Project

Mengze Tian, Yuzheng Zhu, Zhengming Zhu

[1]EPFL

mengze.tian@epfl.ch, yuzheng.zhu@epfl.ch, zhengming.zhu@epfl.ch

1. System Dynamics

In this project we model a car moving in 2D plane with a four-state kinematic. All the details could be found in project documents. To save space, the description of the system dynamics has been omitted here.

2. Linearization

2.1. The analytical expressions of $f_s(\mathbf{x_s}, \mathbf{u_s})$, A and B

$x_s = [0,\ 0,\ 0,\ V_s]^T$ and $u_s = [0,\ u_{T,s}]^T$.
From system dynamic, we know the slip angle $\beta = arctan(\frac{l_r tan(\delta)}{l_r + l_f})$. Since $\delta = 0$, we have $\beta = arctan(0) = 0$. With $x,\ y\ and\ \theta = 0$, we have

$$\begin{aligned} f_s(\mathbf{x_s}, \mathbf{u_s}) =& [Vcos(\theta+\beta),\ Vsin(\theta+\beta), \frac{V}{l_r}sin(\beta),\ \frac{F_{motor} - F_{drag} - F_{roll}}{m}]^T \\ =& [Vcos(\theta+\beta),\ Vsin(\theta+\beta), \frac{V}{l_r}sin(\beta),\ \frac{\frac{u_T P_{max}}{V} - \frac{1}{2}\rho C_d A_f V^2 - C_r mg}{m}]^T \\ =& [V_s,\ 0,\ 0,\ \frac{\frac{u_T P_{max}}{V_s} - \frac{1}{2}\rho C_d A_f V_s^2 - C_r mg}{m}]^T. \end{aligned} \tag{1}$$

For Matrix A,

$$\begin{aligned} \frac{\partial f(x,u)}{\partial x} &= \begin{bmatrix} 0 & 0 & -Vsin(\theta+\beta) & cos(\theta+\beta) \\ 0 & 0 & Vcos(\theta+\beta) & sin(\theta+\beta) \\ 0 & 0 & 0 & \frac{sin(\beta)}{l_r} \\ 0 & 0 & 0 & \frac{-u_{T,s}P_{max} - \rho C_d A_f V^3}{mV^2} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-u_{T,s}P_{max} - \rho C_d A_f V_s^3}{mV_s^2} \end{bmatrix}. \end{aligned} \tag{2}$$

For Matrix B,

$$\frac{\partial \beta}{\partial \delta} = \frac{\partial}{\partial \delta}(arctan(\frac{l_r tan(\delta)}{l_r + l_s})). \tag{3}$$

Set ratio $r = \frac{l_r}{l_r + l_s}$, We have

$$\frac{\partial \beta}{\partial \delta} = \frac{r * sec^2(\delta)}{1 + r^2 * tan^2(\delta)}. \tag{4}$$

Since $\delta = 0$, we obtain that

$$\frac{\partial \beta}{\partial \delta} = r. \tag{5}$$

Therefore, the matrix

$$\begin{aligned} B &= \frac{\partial f(x,u)}{\partial u} \\ &= \begin{bmatrix} -Vsin(\theta+\beta)\frac{\partial \beta}{\partial \delta} & 0 \\ -Vcos(\theta+\beta)\frac{\partial \beta}{\partial \delta} & 0 \\ \frac{V}{l_r}cos(\beta)\frac{\partial \beta}{\partial \delta} & 0 \\ 0 & \frac{P_{max}}{mV} \end{bmatrix}. \end{aligned} \tag{6}$$

Substitute $x_s,\ u_s$ in, We finally get

$$B = \begin{bmatrix} 0 & 0 \\ \frac{l_r}{l_r+l_s}V_s & 0 \\ \frac{1}{l_r+l_s}V_s & 0 \\ 0 & \frac{P_{max}}{mV_s} \end{bmatrix}. \tag{7}$$

2.2. Reason for independently decomposition feasibility

We operate the Todo 2.1 code and the output results are exactly same with the above analytical one. This decomposition is reasonable and makes sense.
From the mechanical perspective: Vehicles operating on highways tend to maintain straight-line motion. During turning, in order to ensure safety, most situations involve minimal changes in turning speed and turning angle. This means that the decomposition of vehicle motion into longitudinal and lateral components is appropriate. The system can be divided into two parts:

1. **Longitudinal Dynamics in x direction**: Governed by changes in velocity (V), which primarily affect the x value (straight-line driving). The throttle control variable u_T is the main input here, leading to changes in speed V, thereby influencing x.

2. **Lateral Dynamics in y direction**: Governed by the steering angle θ, which directly impacts the slip angle β, thereby adjusting the y value.

From a physical perspective:
- The longitudinal dynamics are mainly driven by the motor (input: u_T), which changes speed V.
- The lateral dynamics are mainly driven by the steering system (input: δ).

In terms of lateral motion, although V indirectly influences the lateral dynamics through β and θ, the effect is negligible under small turning angles. For small-angle maneuvers, this effect can be ignored. Similarly, over the short time scale where V remains nearly constant, the longitudinal dynamics are not significantly impacted by δ or β. Therefore, we can reasonably assume that the dynamics are decoupled.

This also shown in our linearized model. For example:

- $A(2,4)$ represents the effect of state V on the lateral dynamics, which is set to zero.
- $B(4,1)$ represents the influence of the input δ on updating V, which is also negligible and set to zero.

Thus, we can separate the dynamics into:
- The longitudinal subsystem (x, V) controlled by u_T.
- The lateral subsystem (y, θ) controlled by δ.

3. MPC

3.1. Design procedure that ensures recursive constraint satisfaction

In Part II, we completed the linearization, decomposition into and discretization of the system. In this part, we achieve recursive constraint satisfaction for vehicle motion by designing and programming MPC controllers for each subsystem.

We introduce the design procedure in the following content. All the details of design could be found in the code files. **Code is attached and plot code is inside the subclass MpcControl_lat.m**

We define *MpcControlBase.m* parent class file and *MpcControl_lon.m* and *MpcControl_lat.m* subclass file for each subsystem.

In parent class file, constructor method MpcControlBase receives subsysem, sample time and horizon length inputs. Then it finishes the discretization of subsystem and returns properties of it to initiate it.

The *get_u* method is also defined to find out MPC control input u given the reference r to track and the system initial state x_0. This method use *compute_steady_state_target* and *setup_controller* methods, which will be defined in subclass file, to find out the tracking steady state with reference and conduct the MPC controller to find out control input u.

In subclass file, the *compute_steady_state_target* method optimize target problem to find out the 'steady-state' corresponding to reference r:

$$\begin{aligned} \min \quad & u^2, && \text{(8a)} \\ \text{s.t.} \quad & x = f_{\_xs\_us} + A_d(x - x_s) + B_d(u - u_s), && \text{(8b)} \\ & ref = x, && \text{(8c)} \\ & u_{min} \leq u \leq u_{max}. && \text{(8d)} \end{aligned}$$

Noted that in longitudinal subsystem we only solve the problem on $x(2) = V$ since we have no cost and constraints on x-position and it also absolutely wouldn't have a steady state. (We only operate tracking the velocity and the reference constraint would also be added on V).The elements in target problem would be adjusted by adding corresponding index. Similarly, in lateral subsystem we only operate tracking on y-position and the reference constraint would also be added on y.

In method *setup_controller* we construct our MPC controller. We first introduce system properties (A, B, ...) from input system. Then we create SDP variable X and U to store state and input for each step in every open-loop control. Then, we build up an loop to adding input constraints, state constraints, system dynamic constraint for state and sum

up the objective cost function (finite horizon LQR form). We also added terminal constraint(if is not space R^2) and cost. Finally we solve this optimal problem and return the YALMIP optimizer object.

Up to now, we could get input u for each step in the close-loop system. The recursive constraints are satisfied since we added all of them inside of the optimal problem.

3.2. Choice of tuning parameters

For Q and R, we choose Q = eye(2) and R = 1 by tuning it and see the performance. This is a trade-off between input value and state tracking speed . Increasing Q and Q_f (for terminal cost) while decreasing R will accelerate the tracking speed to the reference but will result in more aggressive control inputs. Conversely, increasing R while decreasing Q and Q_f may slow down the tracking speed to the reference but will lead to relatively smoother control inputs, (before hitting the control input limit).

Our choice is a good one because we achieve the requirement that Settling time no more than 10 seconds when accelerating from $80km/h$ to $120km/h$ or 3 seconds doing a lane change (width of track is $3m$).

We choose N to be 12 and the H to be $N \times Ts$, since it's the smallest horizon that could making sure our tracking MPC problem is feasible while speeding up the computational process and saving computational cost.

3.3. Terminal invariant set for the lateral sub-system

To find out the maximal invariant set, we first compute LQR controller for unconstrained system and pre-define an set by state and input constraints. Then we do loop on it. We create a new set using system dynamic, intersect it with the original one to update it and repeat the above procedure until the former and later one are equal.

After finding it, we shift it by x_s, which is computed in method *compute_steady_state_target*, and make sure our final state lies in the shifted one.

The plot of terminal invariant set for the lateral subsystem is shown below.

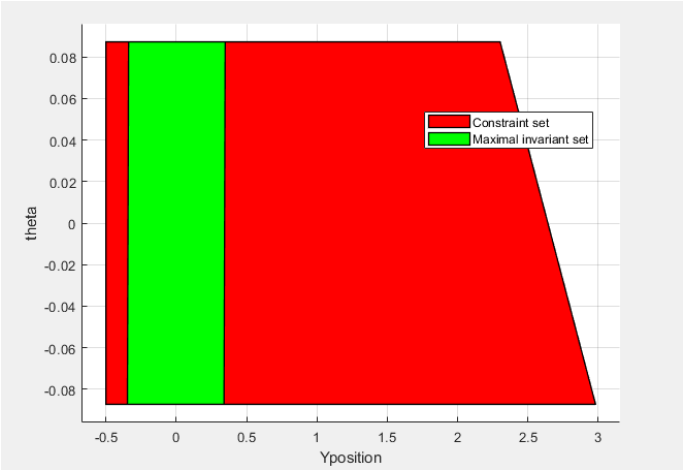

Figure 1. Maximal invariant set for lateral subsystem.

Note that the invariant set should be shifted by x_s before we add it as a terminal constraint to our state. i.e., the constraint changed from

$$F_f X \leq f_f, \tag{9}$$

to

$$F_f(X - X_{ref}) \leq f_f. \tag{10}$$

We only add terminal constraint in lateral subsystem MPC controller since there is no state constraints on x-position and velocity.

3.4. Closed-loop plots for each dimension of the system

The following figure shows the closed-loop plots of simulation. Finally we achieve all control requirements without violate contraints. There is no steady state error at tracking point, which is also our linearization point. However, there do exist a small steady state error in Velocity around $80m/s$ since we linearize our system at $120m/s$ and at $80m/s$ there is a linearization error introduced by the motor and drag term in the system dynamics. This problem will be solved in the following part 4.

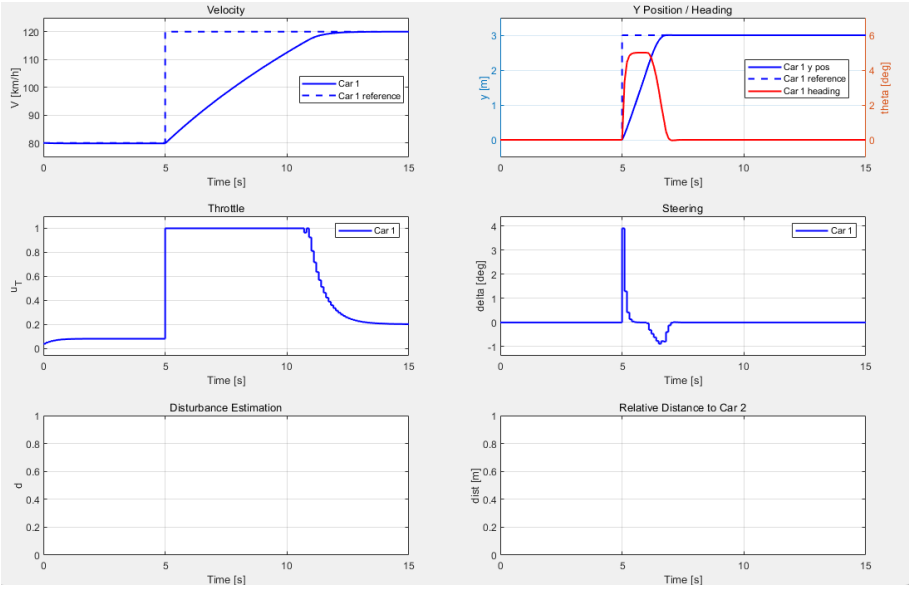

Figure 2. Closed-loop plots for each dimension of the system.

4. Offset-free Tracking

4.1. Disturbance estimator design

Due to the linearization error of the system, the dynamics of the system should be reformulated as:

$$x^+ = f_d(x_s, u_s) + A_d(x - x_s) + B_d(u - u_s) + \hat{B}_d d \tag{11}$$

There is a constant compensation $\hat{V_{est}}$, which can be calculated as follows:

$$\hat{V}_{est} = f_d(X_s, u_s) - A_d \times V_s - B_d \times u_s \tag{12}$$

Here we assumed $\hat{B}_d = B$ because the unknown disturbance matrix is the submatrix caused by input u_T, and $C_d = 0$. To design the dynamics of offset-free system, we should extend the state estimate to (V, d), the estimates of the state and disturbance can be calculated by:

$$\begin{bmatrix} \hat{V}_{k+1} \\ \hat{d}_{k+1} \end{bmatrix} = \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_k \\ \hat{d}_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k + \begin{bmatrix} L_V \\ L_d \end{bmatrix} \left(C\hat{V}_k + C_d\hat{d}_k - y_k \right) + \begin{bmatrix} \hat{V}_{est} \\ 0 \end{bmatrix} \tag{13}$$

.

And the error dynamics now is:

$$\begin{aligned} \begin{bmatrix} V_{k+1} - \hat{V}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} &= \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_k \\ d_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_k \\ &\quad - \begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{V}_k \\ \hat{d}_k \end{bmatrix} - \begin{bmatrix} B \\ 0 \end{bmatrix} u_k \\ &\quad - \begin{bmatrix} L_V \\ L_d \end{bmatrix} \left(C\hat{V}_k + C_d\hat{d}_k - CV_k - C_d d_k \right) \\ &= \left(\begin{bmatrix} A & B_d \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} L_V \\ L_d \end{bmatrix} \begin{bmatrix} C & C_d \end{bmatrix} \right) \begin{bmatrix} V_k - \hat{V}_k \\ d_k - \hat{d}_k \end{bmatrix}, \end{aligned} \tag{14}$$

where A, B, C should be submatrixes of the original dynamics of system because we only focus on tracking without velocity offset. And we placed the poles of the system $A + LC$ to $[0.5, 0.6]^T$ to obtain matrix L, which ensures a fast estimation.

4.2. Comparison between offset tracking and offset-free tracking

To show the impact of the state estimator on offset-free velocity tracking, We simulated the system from $x_0 = [0, 0, 0, 80/3.6]'$ with $ref = [3, 50/3.6]'$ for 15 seconds, where the reference step is delayed by 2 seconds. The results are shown in Fig 3 and Fig 4 respectively.

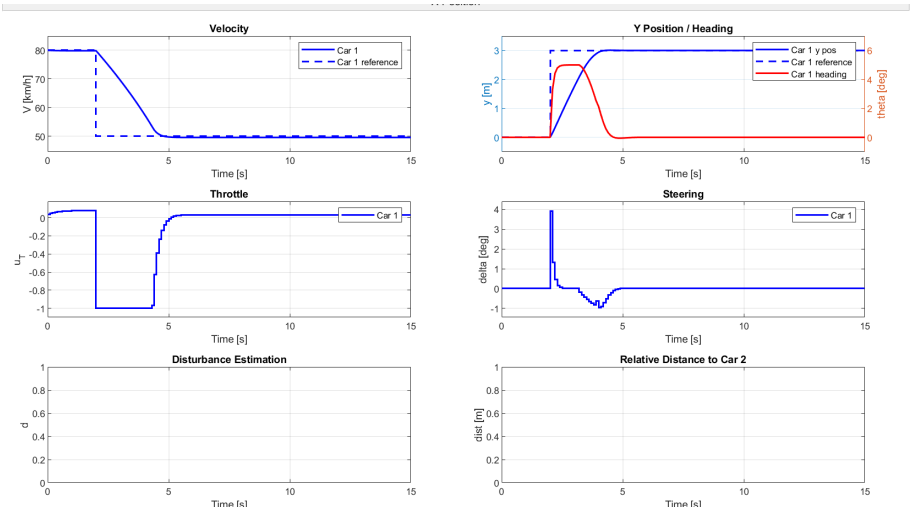

Figure 3. Without offset-free tracking.

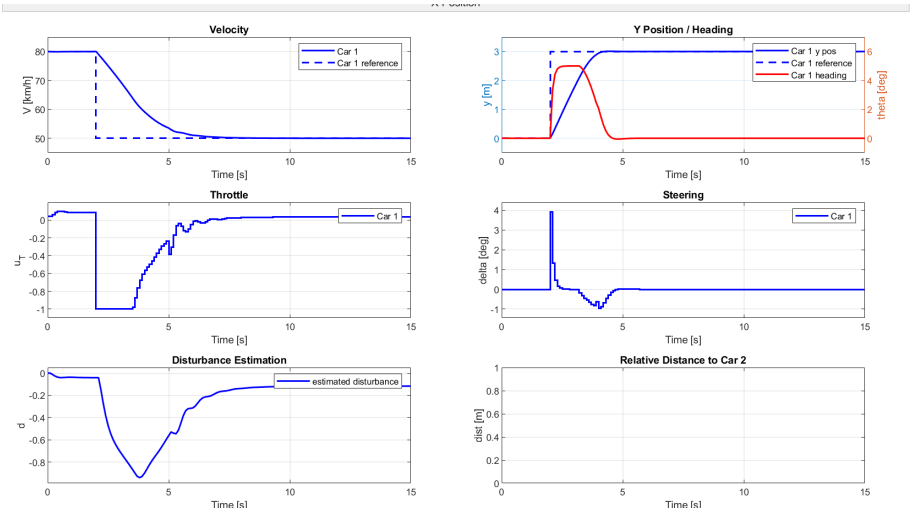

Figure 4. With offset-free tracking.

The velocity offset-free tracking can really compensate via adding disturbance estimator according to the Tab 1. We linearized system at $120m/s$, which is not the steady-state velocity.

Table 1. Comparison. We achieve constant disturbance elimination by our observer.

	Ref(m/s)	Tracking(m/s)
Offset	50	49.535
Offset-free	50	50.000

4.3. Different poles of $A + LC$

When placing different poles of $A + LC$ to $[0.3, 0.4]$, the performance is shown in Fig 5. We can see that the disturbance estimation is fluctuating, which leads to the velocity tracking fluctuate around $50m/s$ as well.

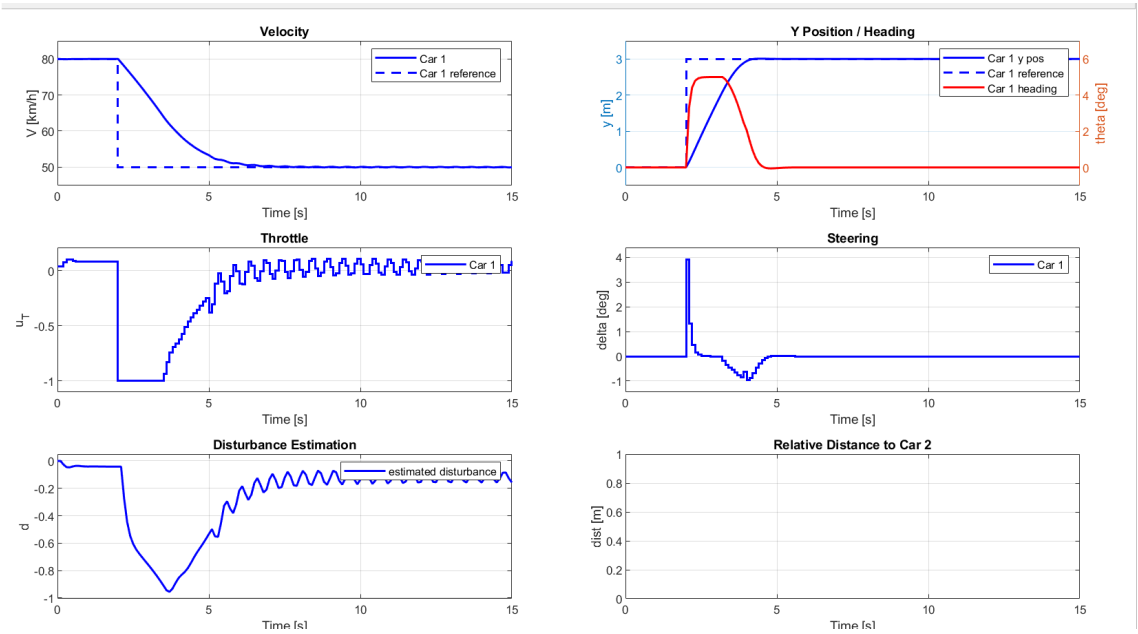

Figure 5. Oscillatory behavior caused by inappropriate poles.

5. Robust Tube MPC

5.1. Design Procedure and Tuning Parameters

5.1.1. Construction of the Maximal Invariant Set

Table 2. LQR Design Weights.

Parameter	Value
Q	0.1
R	1.0

We compute the maximal robust invariant set (mRPI) and tightened constraints. An LQR controller is designed using the feedback gain K, which is calculated with the state and input weight matrices shown in Tab 2. The relatively small value of Q ensures smooth and relaxed feedback, avoiding abrupt changes in the control input u for stable and comfortable operation. The tightened constraints are derived as:

$$\tilde{\mathcal{X}} = \mathcal{X} - \mathcal{E}, \quad \tilde{\mathcal{U}} = \mathcal{U} - K\mathcal{E}. \tag{15}$$

The constraints and invariant sets are visualized in Fig 6, and the results are saved for use in the robust MPC controller.

5.1.2. Controller Design

The robust Model Predictive Controller (MPC) is designed to regulate system dynamics under disturbances, ensuring constraint satisfaction and maintaining safety. The design of the objective function reflects a trade-off between tracking accuracy and control smoothness. The weights Q_x and Q_v penalize deviations in position and velocity, ensuring precise trajectory tracking. The penalty on changes in control input, represented by R, promotes smooth transitions, which are critical for system stability and avoiding aggressive control actions. The optimization problem is formulated as follows:

$$\min_{z,V} \quad \sum_{k=1}^{N-1} \left(z_{1,k}^\top Q_x z_{1,k} + z_{2,k}^\top Q_v z_{2,k} + \Delta V_k^\top R \Delta V_k \right) + z_N^\top P z_N, \tag{16a}$$

$$\text{s.t:} \quad H_e\left(x_{0,\text{other}} - x_0 - x_{\text{safe}} - z_0\right) \leq h_e, \tag{16b}$$

$$z_{k+1} = A z_k + B V_k, \quad k = 1, \ldots, N-1, \tag{16c}$$

$$H z_k \leq h, \quad k = 1, \ldots, N, \tag{16d}$$

$$F V_k \leq f, \quad k = 1, \ldots, N-1, \tag{16e}$$

$$F_f z_N \leq f_f. \tag{16f}$$

The optimization variables consist of z_k, which represents the nominal state at time step k, and V_k, denoting the nominal control input, with the smoothness of the control action enforced through the term $\Delta V_k = V_{k+1} - V_k$. The parameter values, $Q_x = 0.1$, $Q_v = 0.1$, and $R = 100$, are chosen to optimize the trade-off between trajectory tracking performance, input smoothness, and system stability.

5.2. Plots of Minimal Invariant Set and Terminal Set

The cruise control system is designed with carefully imposed constraints on velocity and position. As depicted in Fig 6B, the relative position and velocity between the ego vehicle and the lead vehicle are **constrained deliberately** within a predefined to ensure cruise control. Furthermore, the parameters are meticulously fine-tuned to ensure that the terminal set encompasses the origin.

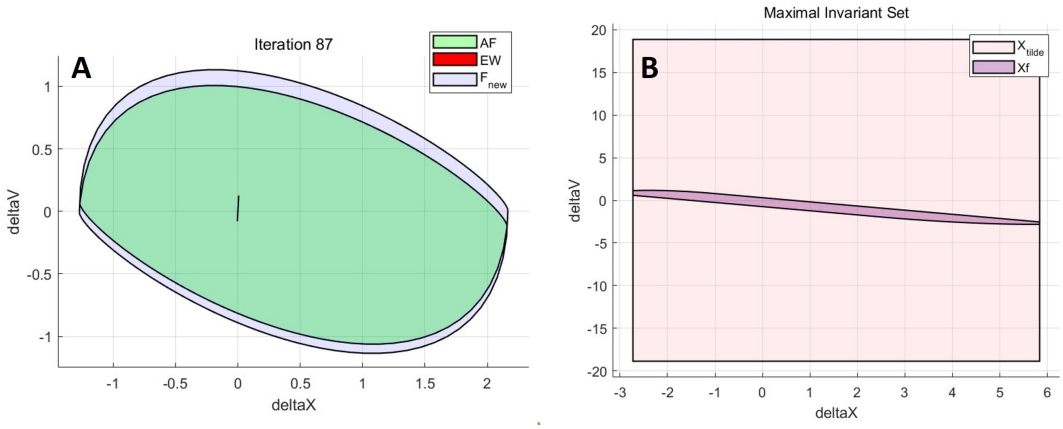

Figure 6. **(A) Maximal Invariant Set F_{new}.** EW represents the disturbance set, while AF denotes the propagation of all previous disturbances. (B)**Terminal Set $\mathcal{X}_f$.** $\tilde{\mathcal{X}}$ is obtained through set tightening operations.

5.3. Performance Evaluation

We evaluated the robust controller in two scenarios: one with the other vehicle following a constant velocity, and another adapts to the conditions of the leading vehicle to complete tracking. These tests demonstrate the controller's robustness to varying initial conditions and interactions between vehicles.

5.3.1. Easy Mode Tracking

As illustrated in Fig 7, we achieve easy mode tracking.

5.3.2. Hard Mode Tracking

As illustrated in Fig 8, we achieve hard mode tracking.

6. Nonlinear MPC

6.1. Nonlinear Controller

6.1.1. Design of Nonlinear System Dynamics

We model the vehicle's nonlinear dynamics and discretize them to facilitate the design of a nonlinear model predictive controller (NMPC). The dynamic model captures the key

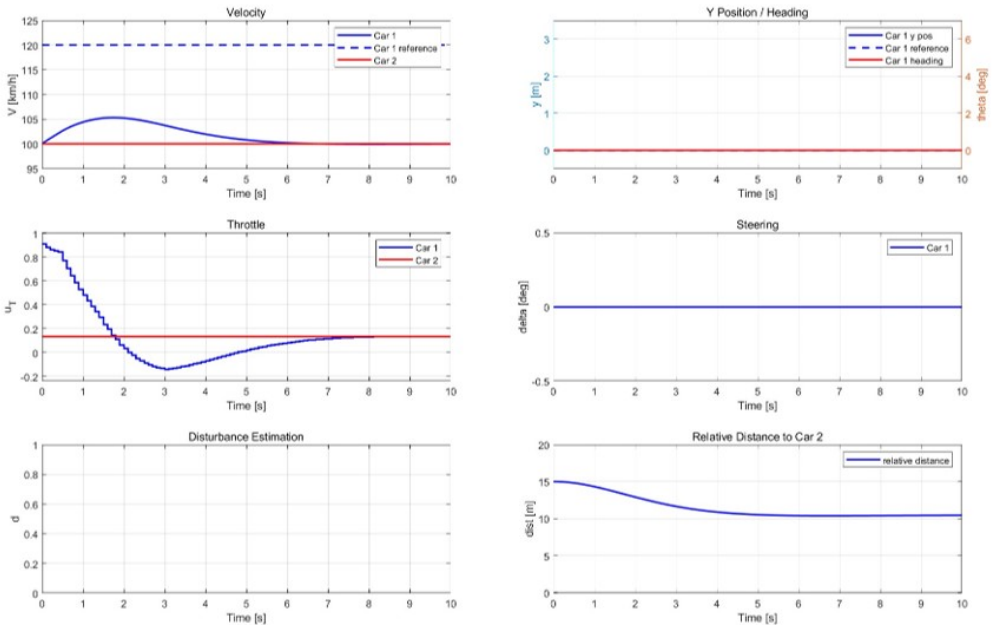

Figure 7. Controller performance in easy setting. The safe distance x_{safe} is set to 10 m to ensure safety.

lateral and longitudinal behaviors of the vehicle, as well as external forces acting on it. The continuous-time dynamics are represented as:

$$\dot{x} = V\cos(\theta + \beta), \tag{17a}$$

$$\dot{y} = V\sin(\theta + \beta), \tag{17b}$$

$$\dot{\theta} = \frac{V\sin(\beta)}{\ell_r}, \tag{17c}$$

$$\dot{V} = \frac{F_{\text{motor}} - F_{\text{drag}} - F_{\text{roll}}}{m}. \tag{17d}$$

To enable the implementation of NMPC, the continuous-time dynamics are discretized using the fourth-order Runge-Kutta (RK4) method. This discretization ensures numerical stability while preserving the nonlinear characteristics of the system. The discretized state transition equation is given by:

$$s_{k+1} = s_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{18}$$

where k_1, k_2, k_3, and k_4 are intermediate evaluations of the dynamic function. This approach allows for accurate modeling of the vehicle dynamics and sets the foundation for subsequent optimization and control design.

6.1.2. Optimization Formulation

The nonlinear model predictive control (NMPC) problem is formulated to minimize a cost function while ensuring compliance with system dynamics and constraints. The

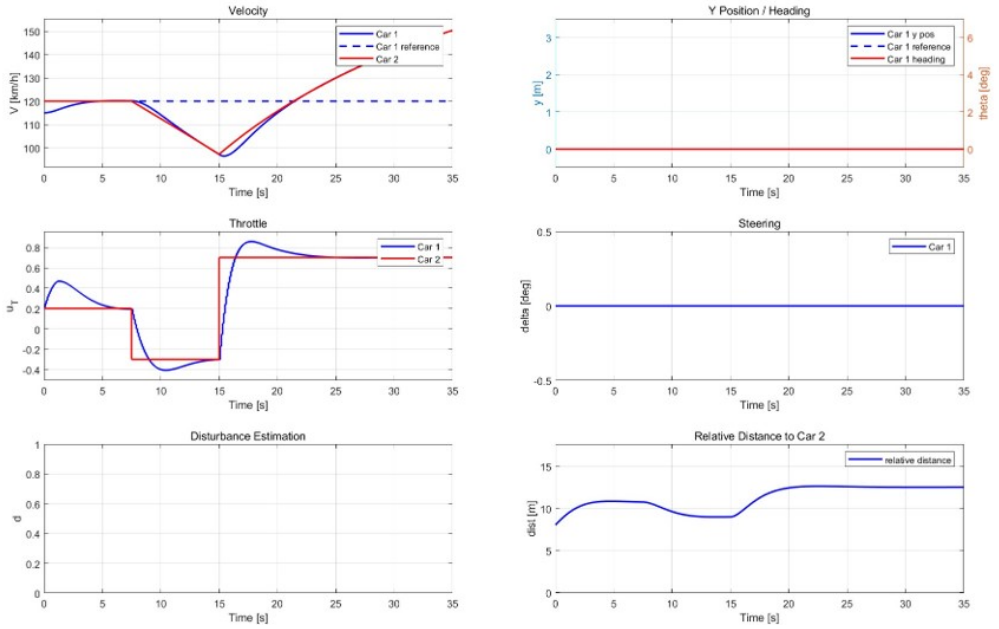

Figure 8. Controller performance in hard setting. We take passenger ride comfort into account by adjusting the MPC parameters to ensure smooth velocity transitions of the vehicle.

objective function is defined as:

$$\min_{X,U} \quad \sum_{k=1}^{N-1} \left(Q_y(y_k - y_{\text{ref}})^2 + Q_v(V_k - V_{\text{ref}})^2 + U_k^\top R U_k\right) + Q_y(y_N - y_{\text{ref}})^2 + Q_v(V_N - V_{\text{ref}})^2, \tag{19a}$$

$$\text{s.t:} \quad X_{k+1} = f_{\text{discrete}}(X_k, U_k), \quad k = 1, \ldots, N-1, \tag{19b}$$

$$X_{\text{lb}} \leq X_k \leq X_{\text{ub}}, \quad k = 1, \ldots, N, \tag{19c}$$

$$U_{\text{lb}} \leq U_k \leq U_{\text{ub}}, \quad k = 1, \ldots, N-1. \tag{19d}$$

where Q_y and Q_v are the weights for position and velocity tracking errors, R is the weight for control effort, y_{ref} and V_{ref} are the reference values for position and velocity, and U_k represents the control inputs. Parameters are tuned and set to 1.

6.1.3. Steady-State Tracking Error Analysis

Nonlinear MPC theoretically eliminates steady-state errors. Its dynamic model avoids linearization, thereby eliminating the inaccuracies associated with linearized models.

6.1.4. Performance Evaluation

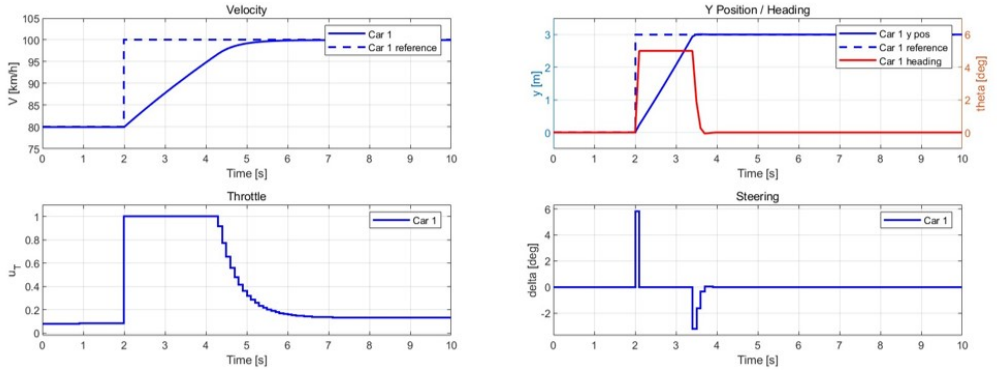

Figure 9. Controller performance in easy setting.

6.2. Overtaking

To achieve overtaking, we formulate hierarchical optimization problems to ensure safe maneuvering. The first optimization problem determines the reference value for y during overtaking by incorporating safety constraints. The second problem tracks the reference, following the approach outlined in Section 6.1.2.

6.2.1. Optimization Formulation

To ensure safe overtaking, we firstly formulate an optimization problem to determine the reference value for the lateral position y of the ego vehicle. The optimization minimizes deviations from the initial reference value y_{initial} while incorporating safety constraints based on elliptical safety zones. The problem is formulated as:

$$\min_{y_{\text{ref}}} \quad (y_{\text{ref}} - y_{\text{initial}})^2, \tag{20a}$$

$$\text{s.t.} \quad (\mathbf{p} - \mathbf{p}_{\text{other}})^\top \mathbf{H} (\mathbf{p} - \mathbf{p}_{\text{other}}) \geq 1, \tag{20b}$$

$$y_{\text{lb}} \leq y_{\text{ref}} \leq y_{\text{ub}}, \tag{20c}$$

where y_{ref} is the optimized lateral reference value, y_{initial} is the initial reference, and $\mathbf{p}$ represents position of the ego vehicle. Specifically, the safety constraint ensures that the relative position between each ellipse point and the other vehicle satisfies a minimum distance defined by the quadratic form $\mathbf{H}$. Additionally, the optimization enforces boundary constraints to restrict y_{ref} within the lane limits $[y_{\text{lb}}, y_{\text{ub}}]$.

The ellipsoidal collision avoidance constraint incorporates a safety factor of 1.3 to account for potential uncertainties. The semi-major axis a and semi-minor axis b are determined as $a = 1.3 \times 4.3\,\text{m}$ and $b = 1.3 \times 1.3\,\text{m}$, where 1.3 represents the safety factor.

6.2.2. Performance Evaluation

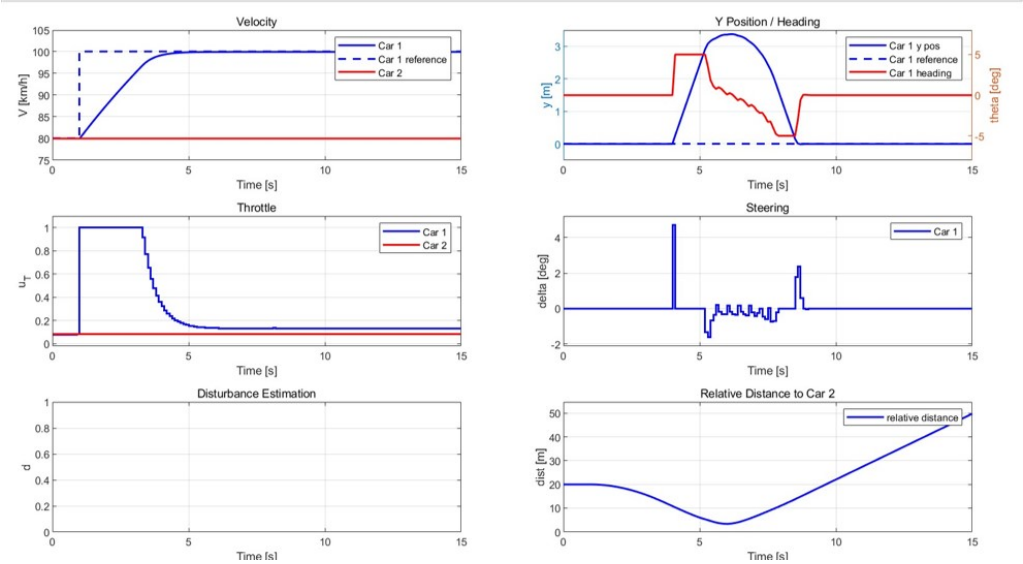

Figure 10. Controller performance in hard setting. We achieved lane-changing overtaking and velocity tracking while ensuring safety.