Model Predictive Control

ME-425 — Fall 2024

MPC Mini-Project

Cruise Controller for a Car on a Highway

Authors:

Yo-Shiun Cheng (386249)
Jiwon You (394716)

Professor:

Collin Jones

Contents

Introduction

During this project for the course ME-425: Model Predictive Control (MPC) instructed by Dr. Colin Jones, we focus on developing advanced controllers for an autonomous vehicle to perform precise lane-keeping, overtaking, and velocity tracking tasks. Our goal is to implement robust MPC algorithms into the provided simulation environment, ensuring optimal performance in diverse driving scenarios.

The project begins with a deep dive into vehicle dynamics, enabling us to construct accurate models and predict system responses under various control inputs. Next, we linearize the dynamics to design subsystem-specific MPC controllers for both longitudinal and lateral control. These controllers are fine-tuned to track references such as velocity and lane position under constraints, such as steering limits and collision avoidance.

Building on this foundation, we implement a sophisticated Nonlinear MPC (NMPC) controller, allowing the vehicle to handle more complex scenarios, including overtaking maneuvers and interactions with other moving vehicles. This controller incorporates collision-avoidance constraints, represented as dynamic ellipsoidal boundaries, ensuring safe overtaking without compromising trajectory accuracy.

1 Deliverable 2.1: Analytical Linearization of the Car Model

1.1 Nonlinear Dynamics

The car dynamics are modeled using the following state equations:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} V\cos(\theta+\beta) \\ V\sin(\theta+\beta) \\ \frac{V}{l_r}\sin(\beta) \\ \frac{u_T P_{\max}}{V} - \frac{1}{2}\rho C_d A_f V^2 - C_r m g \end{bmatrix} \tag{1}$$

where β, the kinematic slip angle is defined as follow,

$$\beta = \tan^{-1}\left(\frac{l_r \tan(\delta)}{l_r + l_f}\right) \tag{2}$$

Here:

- x, y: Longitudinal and lateral positions,
- θ: Yaw angle,
- V: Velocity,
- δ: Steering angle,
- u_T: Normalized throttle input,
- l_r, l_f: Distances from the center of mass to the rear and front axles,
- $P_{\max}$: Maximum motor power,
- ρ: Air density,
- C_d: Drag coefficient,
- A_f: Frontal area,
- C_r: Rolling resistance coefficient,
- m: Vehicle mass,
- g: Gravitational acceleration.

1.2 Linearization

To linearize the system around the steady-state operating point:

$$\mathbf{x}_s = \begin{bmatrix} 0 & 0 & 0 & V_s \end{bmatrix}^\top, \tag{3}$$

$$\mathbf{u}_s = \begin{bmatrix} 0 & u_{T,s} \end{bmatrix}^\top, \tag{4}$$

we compute the Jacobian matrices:

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_s,\mathbf{u}_s}, \tag{5}$$

$$B = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Big|_{\mathbf{x}_s,\mathbf{u}_s}. \tag{6}$$

The system is linearized as: ($\mathbf{f}(\mathbf{x}_s, \mathbf{u}_s)$, A, and B are computed with MATLAB)

$$\mathbf{f}(\mathbf{x},\mathbf{u}) \approx \mathbf{f}(\mathbf{x}_s,\mathbf{u}_s) + A(\mathbf{x}-\mathbf{x}_s) + B(\mathbf{u}-\mathbf{u}_s), \tag{7}$$

$$\mathbf{f}(\mathbf{x}_s,\mathbf{u}_s) = \begin{bmatrix} V_s \\ 0 \\ 0 \\ -\frac{1}{m}(C_r g m - \frac{P_{\max} u_{T,s}}{V_s} + \frac{A_f C_d V_s^2 \rho}{2}) \end{bmatrix}, \tag{8}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{V_s^2 m}(A_f C_d \rho V_s^3 + P_{\max} u_{T,s}) \end{bmatrix}, \tag{9}$$

$$B = \begin{bmatrix} 0 & 0 \\ \frac{V_s l_r}{l_f + l_r} & 0 \\ \frac{V_s}{l_f + l_r} & 0 \\ 0 & \frac{P_{\max}}{V_s m} \end{bmatrix}. \tag{10}$$

2 Deliverable 2.2: Separation into Independent Subsystems

In this deliverable, we analyze the car's linearized dynamics around steady cruising motion and show that the longitudinal (lon) and lateral (lat) dynamics can be treated as independent subsystems. This separation simplifies control design and leverages the decoupled nature of the system dynamics.

2.1 Analysis of the Linearized Dynamics

The car's state-space representation after linearization around a steady state with velocity $V_s = \frac{120}{3.6}$ m/s is given by the following matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1.0000 \\ 0 & 0 & 33.3333 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.0244 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 20.0000 & 0 \\ 12.8205 & 0 \\ 0 & 1.6667 \end{bmatrix}. \tag{11}$$

From the structure of the A and B matrices, we observe:

- The normalized throttle u_T influences the velocity V, which in turn affects the longitudinal position x.

- The steering angle influences the lateral position y and the yaw angle θ, and θ further influences y.
- There is no direct coupling between (x, V, u_T) and $(y, \theta, \text{steering angle})$.

Thus, when the system is linearized around steady cruising motion, the longitudinal and lateral subsystems are decoupled, allowing us to split them into independent dynamics.

2.2 Verification of Decomposition

We verify that the decomposition into longitudinal and lateral subsystems preserves the dynamics information by reconstructing the overall state derivative from the decomposed subsystems. The decomposition involves the following:

- **Longitudinal indices:** $x_{\text{lon}} = [x, V]$, $u_{\text{lon}} = u_T$.
- **Lateral indices:** $x_{\text{lat}} = [y, \theta]$, $u_{\text{lat}} = \text{steering angle}$.

Given the perturbed state $\mathbf{X}$ and input $\mathbf{U}$, the state derivative is computed as:

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{x}_s, \mathbf{u}_s) + \mathbf{A}(\mathbf{X} - \mathbf{x}_s) + \mathbf{B}(\mathbf{U} - \mathbf{u}_s), \tag{12}$$

$$\dot{\mathbf{X}}_{\text{lon}} = \mathbf{A}_{\text{lon}}(\mathbf{X}_{\text{lon}} - \mathbf{x}_s^{\text{lon}}) + \mathbf{B}_{\text{lon}}(\mathbf{U}_{\text{lon}} - \mathbf{u}_s^{\text{lon}}), \tag{13}$$

$$\dot{\mathbf{X}}_{\text{lat}} = \mathbf{A}_{\text{lat}}(\mathbf{X}_{\text{lat}} - \mathbf{x}_s^{\text{lat}}) + \mathbf{B}_{\text{lat}}(\mathbf{U}_{\text{lat}} - \mathbf{u}_s^{\text{lat}}). \tag{14}$$

The total state derivative is reconstructed by combining the longitudinal and lateral dynamics:

$$\dot{\mathbf{X}}_{\text{composed}} = \begin{bmatrix} \dot{\mathbf{X}}_{\text{lon}} \\ \dot{\mathbf{X}}_{\text{lat}} \end{bmatrix} + \mathbf{f}(\mathbf{x}_s, \mathbf{u}_s). \tag{15}$$

The results are:

$$\dot{\mathbf{X}}_{\text{Original}} = \begin{bmatrix} 33.3333 \\ 0 \\ 0 \\ 0.1667 \end{bmatrix}, \quad \dot{\mathbf{X}}_{\text{Composed}} = \begin{bmatrix} 33.3333 \\ 0 \\ 0 \\ 0.1667 \end{bmatrix}.$$

The two results are identical, confirming the correctness of the decomposition.

3 Deliverable 3.1: MPC Controllers for each Subsystems

In the previous section, we linearized the car system and decomposed it into 2 independent subsystems, namely the lateral and longitudinal systems. We will now design two controllers based on Linear MPC.

3.1 MPC Controller for Lateral Subsystem

The lateral subsystem focuses on controlling the lateral position y and the heading angle θ of the vehicle using the steering input δ. the primary objective of the lateral controller are to ensure recursive satisfaction of constraints while maintaining a smooth response and accurate reference tracking.

3.1.1 Problem Formulation

The lateral subsystem's dynamics are described by the discretized state-space model:

$$\mathbf{x_{k+1}} = A_d\mathbf{x_k} + B_d\mathbf{u_k}, \tag{16}$$

where:

- $\mathbf{x_k} = [y_k, \theta_k]^\top$: Lateral states (y position and heading angle),
- $\mathbf{u_k} = \delta_k$: Steering angle,
- A_d: Discrete-time state matrix,
- B_d: Discrete-time input matrix.

For the practical implementation of the MPC controller we have to split the ideal infinite horizon problem into two sub-problems to provide realistic computational cost. From step 0 to N we will formulate the Finite Horizon MPC with states and inputs constraints. Then, from step N to infinity we will drop the constraints and solve the optimization problem to find a control law and its associated invariant terminal set. The cost function is designed to minimize the deviation from the reference state $\mathbf{x_{ref}}$ and input $\mathbf{u_{ref}}$, and is given as:

$$J = \sum_{k=1}^{N-1} \big((\mathbf{x_k} - \mathbf{x_{ref}})^\top Q(\mathbf{x_k} - \mathbf{x_{ref}}) + (\mathbf{u_k} - \mathbf{u_{ref}})^\top R(\mathbf{u_k} - \mathbf{u_{ref}})\big) + ((\mathbf{x_N} - \mathbf{x_{ref}})^\top Q_f(\mathbf{x_N} - \mathbf{x_{ref}}), \tag{17}$$

where:

- Q: State weight matrix,
- R: Input weight matrix,
- Q_f: Terminal cost matrix is the solution of the Discrete Algebraic Riccati equation (DARE)

3.1.2 Constraints

The constraints include:

- **State constraints:**
$$-0.5 \leq y_k \leq 3.5, \quad -5^\circ \leq \theta_k \leq 5^\circ, \tag{18}$$
- **Input constraints:**
$$-30^\circ \leq \delta_k \leq 30^\circ, \tag{19}$$
- **Terminal constraints ensuring recursive feasibility:**
$$\mathbf{x_N} \in \mathcal{X}_f, \tag{20}$$
where $\mathcal{X}_f$ is the terminal invariant set.

3.1.3 Terminal Invariant Set

To ensure stability and recursive feasibility, a terminal invariant set $\mathcal{X}_f$ is computed. The terminal cost matrix Q_f is obtained from the solution of DARE, and the corresponding terminal feedback gain K is used to compute the closed-loop dynamics:

$$A_K = A_d + B_d K. \tag{21}$$

The invariant set is calculated iteratively using the algorithm shown in Figure 1 which finds the maximal invariant set based on the state and input constraints and on the following property: A set C is a control invariant set if and only if $C \subseteq$ *pre(C)*

Conceptual Algorithm to Compute Invariant Set

$\Omega_0 \leftarrow \mathbb{X}$
loop
  $\Omega_{i+1} \leftarrow \mathrm{pre}(\Omega_i) \cap \Omega_i$
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{O}_\infty = \Omega_i$
  end if
end loop

Figure 1: Algorithm that computes a control invariant set.

We obtain the following terminal invariant set given our constraints and cost matrices as shown in Figure 2.

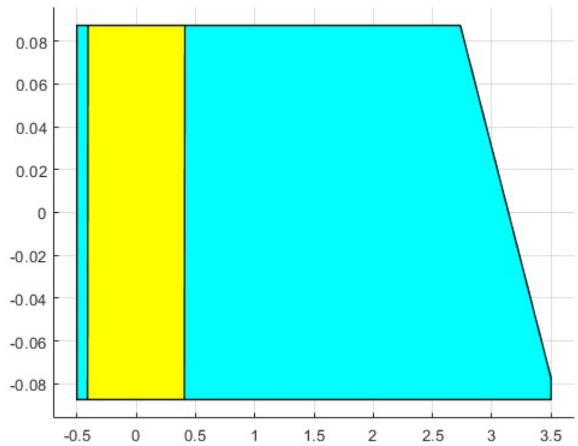


Figure 2: Terminal invariant set (yellow) of lateral subsystem, where the cyan part is the polytopic constraints.

3.1.4 MPC Tuning Parameters

MPC controllers need the tuning of the following parameters: the horizon length H, the matrix Q and the matrix R which are both involved in the cost function, and the terminal cost matrix Q_f.

1. **Tuning Q and R:** Initially we have $Q = \mathrm{diag}(1, 1)$ and $R = 1$, and it works just fine as it satisfied the goal specified in the project description (the settling time is no more 3 seconds when doing a lane change). So we just kept them.

2. **Tuning Q_f:** Q_f is obtained using the LQR solution from DARE using the function `dlqr(A,B,Q,R)` in Matlab.

3.2 MPC Controller for Longitudinal Subsystem

The longitudinal controller is designed to track the velocity reference V_{ref} using the normalized throttle input u_T while ensuring that the vehicle maintains stability and operates within the defined constraints.

3.2.1 Problem Formulation

The longitudinal subsystem's dynamics are described by the discretized state-space model:

$$\mathbf{x_{k+1}} = A_d \mathbf{x_k} + B_d \mathbf{u_k}, \tag{22}$$

where:

- $\mathbf{x_k} = [x_k, V_k]^\top$: Longitudinal states (x position and velocity),
- $\mathbf{u_k} = u_T$: Normalized throttle input.

The cost function is formulated to penalize deviation of the state $\mathbf{x}$ from the reference velocity V_{ref} and minimize excessive throttle usage:

$$J = \sum_{k=1}^{N-1} \left((\mathbf{x_k} - [0; V_{ref}])^\top Q (\mathbf{x_k} - [0; V_{ref}]) + (\mathbf{u_k} - \mathbf{u_{ref}})^\top R (\mathbf{u_k} - \mathbf{u_{ref}}) \right) + ((\mathbf{x_N} - \mathbf{x_{ref}})^\top Q_f (\mathbf{x_N} - \mathbf{x_{ref}}). \tag{23}$$

3.2.2 Constraints

The constraints include:

- **Input constraints:**

$$-1 \leq u_T \leq 1, \tag{24}$$

3.2.3 MPC Tuning Parameters

MPC controllers need the tuning of the following parameters:the matrix Q and the matrix R, and the terminal cost matrix Q_f.

1. **Tuning Q and R:** Initially we have $Q = \text{diag}(0, 1)$ and $R = 1$ as we do not have to penalize x position, and it works also pretty well as it satisfied the goal specified in the project description (the settling time is no more than 10 seconds when accelerating from 80 to 120 km/h). So we kept them.
2. **Tuning Q_f:** We use $Q_f = Q$ due to the reason that the computation of terminal invariant set is uneeded.

3.3 Horizon Length and Simulation Results

We then applied both lateral and longitudinal controllers to the system and chose a suitable horizon length H to simulate it.

3.3.1 Tuning Horizon Length H

The prediction horizon H significantly affects both the region of attraction and the convergence speed. If the chosen value of H is too small, the optimization problem may become infeasible, as the controller cannot compute a solution that reaches the terminal set within the limited number of steps. Conversely, selecting a larger H increases computational complexity, leading to higher processing costs. After testing, we have $H = 5$ as there's no obvious performance increase while having a longer horizon length.

3.3.2 Simulation Results

Figure 3 shows that the car is able to perform a lane change successfully while respecting all the constraints applied to the system. There is no steady state error because the reference velocity is exactly the linearization velocity.

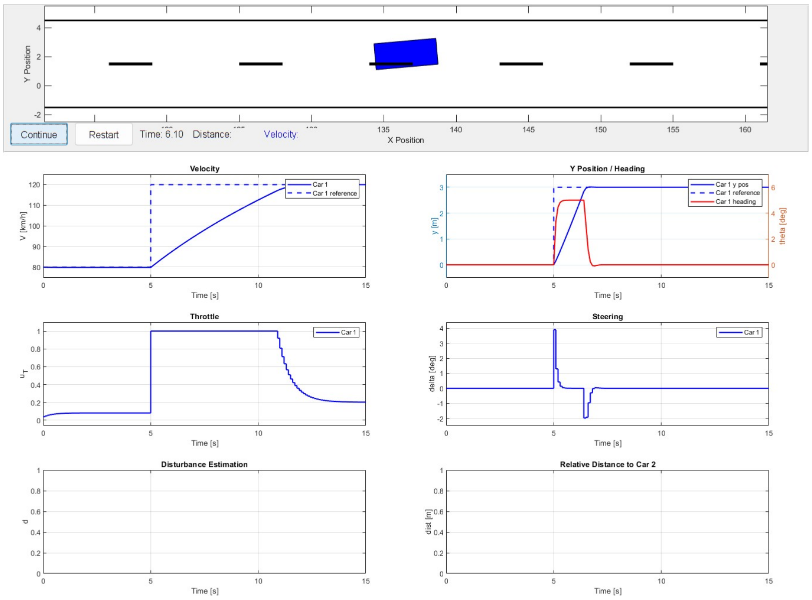


Figure 3: Visualization of the simulation results of the Linear MPC controller

4 Deliverable 4.1: Offset-free tracking

4.1 Design Procedure

The system is linearized around a chosen steady state (x_s, u_s). Tracking any other velocity which is not the linearization steady state will induce linearization error. We introduce a constant unknown disturbance d to account for the linearization error.

$$\mathbf{x}^+ = f_d(x_s, u_s) + A_d(\mathbf{x} - \mathbf{x_s}) + B_d(\mathbf{u} - \mathbf{u_s}) + \hat{B}_d d \tag{25}$$

Then we define a reduced system of velocity V and disturbance d. We also add a Luenberger observer.

$$\hat{z}^+ = f_d(z_s, u_s) + \hat{A}(\hat{z} - z_s) + \hat{B}(u - u_s) + L(\hat{C}\hat{z} - y) \tag{26}$$

where z $= \begin{bmatrix} V \\ d \end{bmatrix}$ and y is measurement of longitudinal speed V. Longitudinal position x is ignored.

4.2 Choice of tuning parameters

In addition to Q,R and horizon length H, we must tune the observer gain L. Increasing the gain L, i.e having small eigenvalues for $A + LC$, makes the observer more sensitive to high frequency. On the other hand, decreasing the gain L, i.e having large eigenvalues for $A + LC$, makes the observer slow. Given that our disturbance is constant, i.e extremely low frequency, there is no need for a fast observer. Therefore, we chose the eigenvalue of $[\mathbf{0.7}, \mathbf{0.9}]$.
Choice of Q,R and H is identical to section 3.

4.3 Plots and analysis

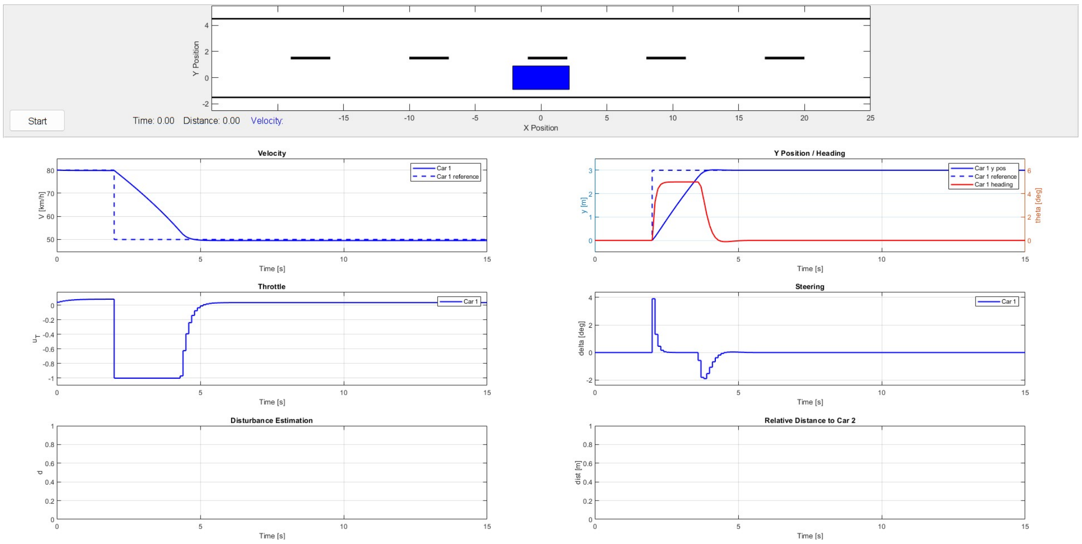


Figure 4: Simulation without estimator

In Figure 4 we have simulation result without estimator.

- The car is linearized around 120km/h but is tracking 80km/h and 50km/h, which induces offset due to linearization error
- In the plot one observes that there is steady state error in both 80km/h and 50km/h

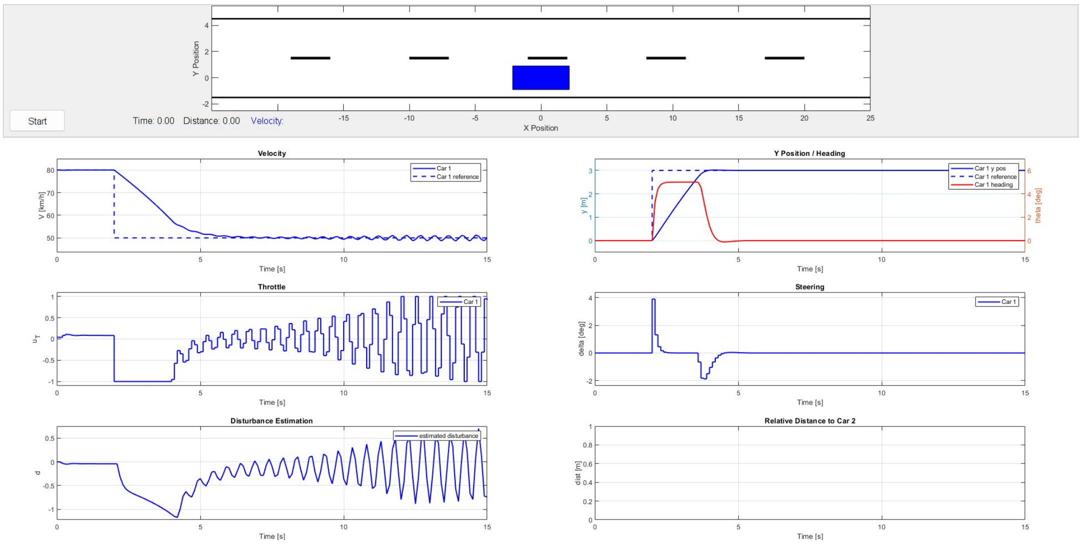


Figure 5: Simulation with estimator, eigenvalue 0.2 0.4

In Figure 5 we have simulation result with estimator.

- The eigenvalues are set to $[\mathbf{0.2}, \mathbf{0.4}]$, which makes the observer fast and sensitive to high frequency noise.
- Estimated disturbance is diverging and velocity fails to converge to reference speed, due to poor tuning of observer gain.

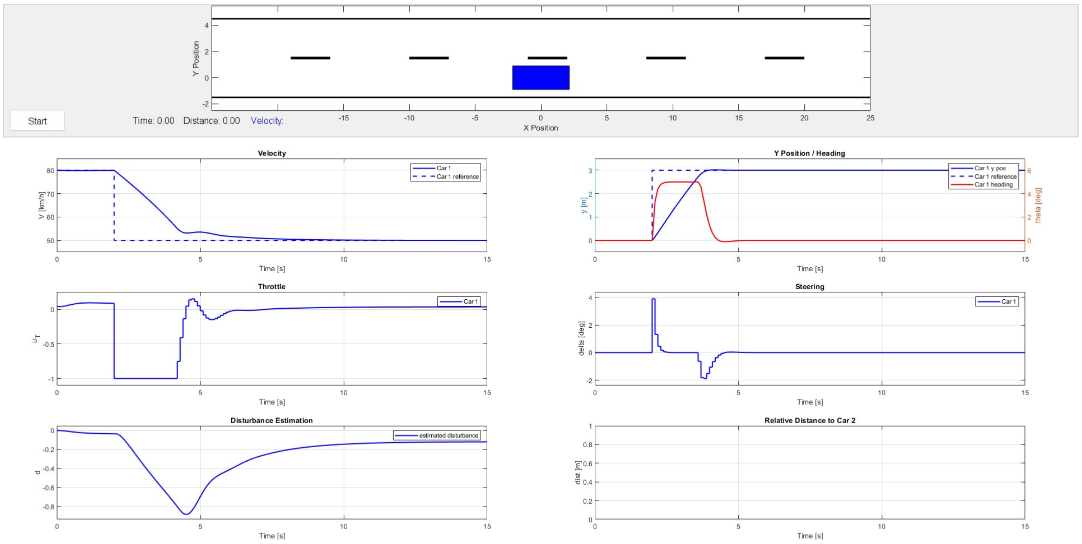


Figure 6: Simulation with estimator, eigenvalue 0.7 0.9

In Figure 6 we have simulation result with estimator.

- The eigenvalues are set to $[\mathbf{0.7}, \mathbf{0.9}]$, which makes the observer slow and stable
- Unlike Figure 4, the system is able to track 80km/h and 50km/h without offset. This is because we are estimating virtual disturbance to compensate for linearization error

- Recall that the car is linearized around 120km/h. Therefore, linearization error will be larger at 50km/h than 80km/h. This is reflected in estimated disturbance plot. At 80km/h disturbance converges to $\mathbf{-0.03}$, and to a much larger $\mathbf{-0.11}$ at 50km/h.

5 Deliverable 5.1 : Tube MPC

The goal of this controller is to track reference velocity while keeping a minimum distance from the lead car.

5.1 Problem Formulation

We introduce delta dynamics.

$$\Delta^+ = A_d\Delta - B_d u + B_d \tilde{u_T} \tag{27}$$

where $\Delta = \begin{bmatrix}\Delta_x \\ \Delta_V\end{bmatrix} = \begin{bmatrix}\tilde{x} - x - x_{safe} \\ \tilde{V} - V\end{bmatrix}$. It must satisfy $\Delta_x \geq 6 - x_{safe}$, where x_{safe} is a design choice.
We shall treat $B_d\tilde{u_T}$ as disturbance $w \in W$. Since $\tilde{u_T} \in [u_{Ts} - 0.5, u_{Ts} + 0.5] = U_{TS}$, we have $W = B_d * U_{TS}$.
First we find an LQR controller K for $\Delta^+ = A_d\Delta - B_d u + w$. Choose Q = diag([1,1]) and R=1 for simplicity.
Minimal invariant set $\mathcal{E}$ is found by adding(in the sense of Minkowski sum) $(A_d - B_dK)^iW$ until convergence, given in Figure 7.

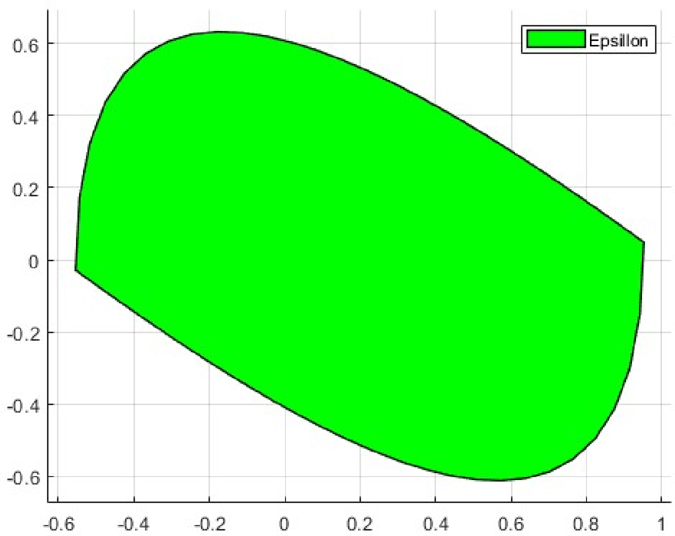


Figure 7: Minimal invariant set $\mathcal{E}$

Then we compute state constraint sets. Original constraint for Δ is $\Delta_x \geq 6 - x_{safe}$, which we will denote by X. Then for tube center z, we have $z \in X \ominus \mathcal{E}$. For terminal set X_f, we compute the maximal control invariant set with LQR gain K from above.

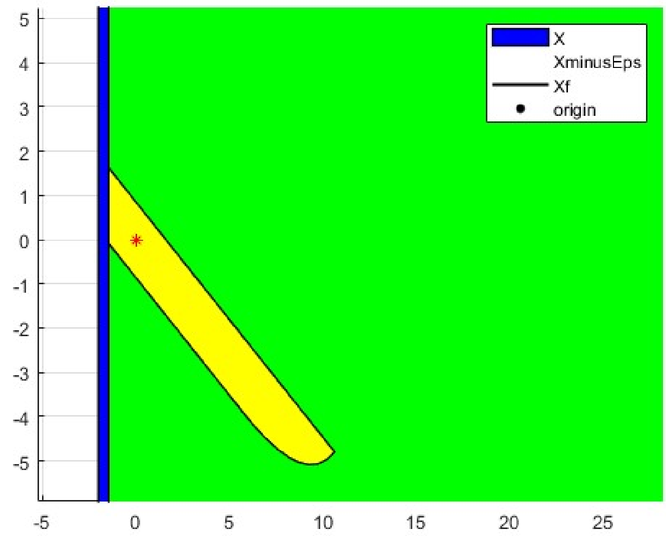


Figure 8: $X, X \ominus \mathcal{E}, X_f$

. From Figure 8 we observe that XminusEps, colored green, is indeed X tightened by $\mathcal{E}$. Yellow terminal set X_f contains the origin.

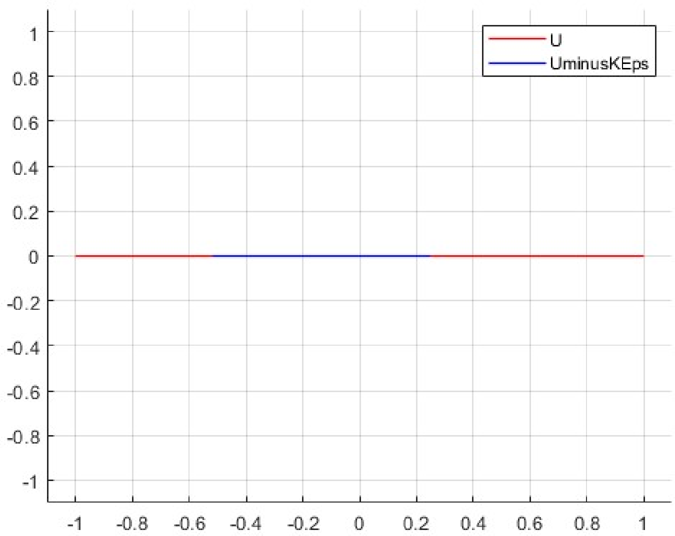


Figure 9: $U, U \ominus K\mathcal{E}$

. In Figure 9, it is shown approximately that nominal control $v \in [-0.5, 0.2]$ while actual control $u \in [-1, 1]$. Nominal control v is tighter, because the remaining authority is saved to compensate for unknown disturbance. While we are using Δ as state, we want V to track reference velocity, $V \to V_{ref}$.

$$z_V \approx \Delta_V = \tilde{V} - V \tag{28}$$

$$V - V_{ref} \approx \tilde{V} - z_V - V_{ref} \to 0 \tag{29}$$

Therefore, MPC cost is implemented as $I = \Sigma(Q * (\tilde{V} - z_V - V_{ref})^2 + R * (v - u_{ref})^2)$
General tube MPC implementation follows from lecture note.

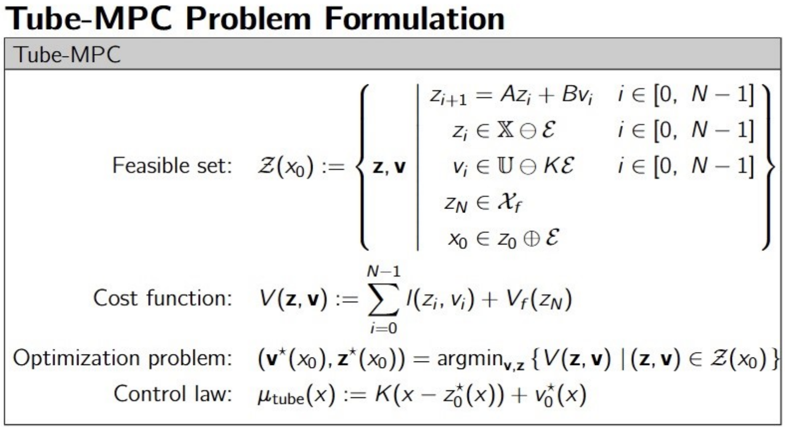


Figure 10: Tube MPC implementation

.

5.2 Choice of tuning parameters

1. x_{safe}**:** The larger it is, terminal constraint set X_f will easily contain origin(Figure 8). It has to be strictly bigger than 6. However, if we set a too large value, the controller will prioritize staying far behind lead car and there will be less control authority for tracking reference velocity. To exploit control authority, I chose the value of 8.

2. Q **and** R**:** It is used in computation of $\mathcal{E}$ and MPC cost. We chose eye(2) and 1 for simplicity, which is working fine. It leaves enough authority for nominal control v. $Q_f = Q$ also for simplicity. In general large Q enforces the system to closely track the reference, at the expense of control efforts. With large R, it will use as little control effort as possible.

3. **Horizon length:** Large horizon gives better control with higher computational cost. We are using $H = 4$ as it gives good control in reasonable computation time.

5.3 Plots and analysis

5.3.1 Constant lead velocity

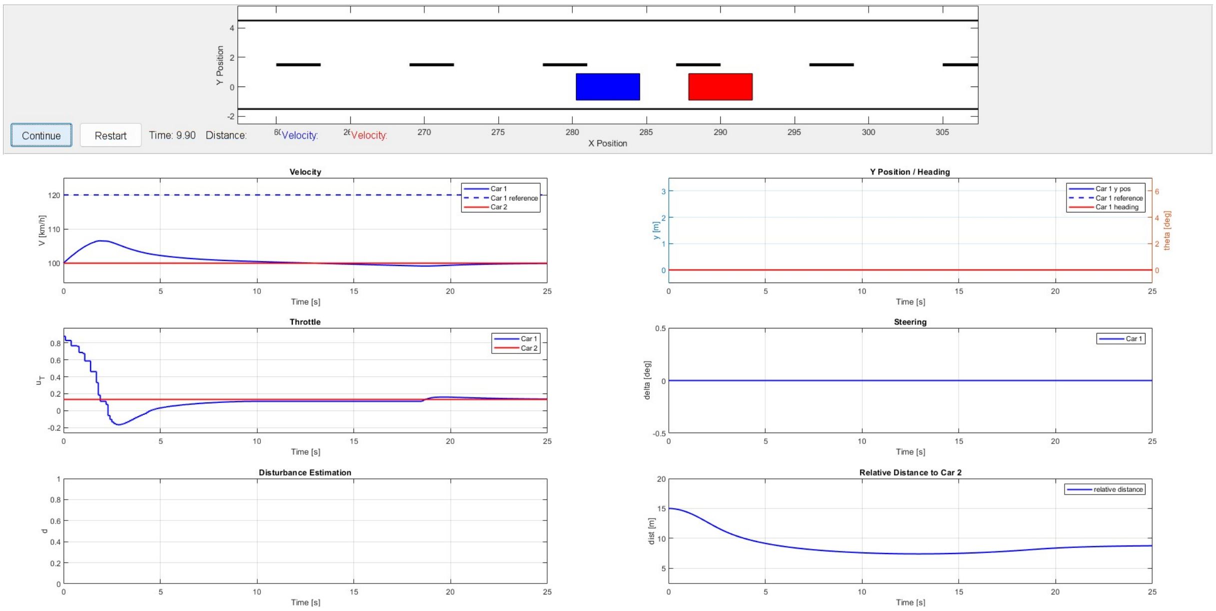


Figure 11: Simulation with constant lead velocity

. Initially lead car is 15m ahead of ego car. Lead car is moving at 100km/h and ego car has a reference velocity of 120km/h. If ego car follows its reference velocity strictly, it will soon collide with lead car. The velocity plot shows that initially ego car increases speed towards 120km/h, but soon slows down to maintain safety distance. Plot of relative distance shows that it stays strictly above 6 and converges around 8, which is our choice for x_{safe} parameter.

5.3.2 Variable lead velocity

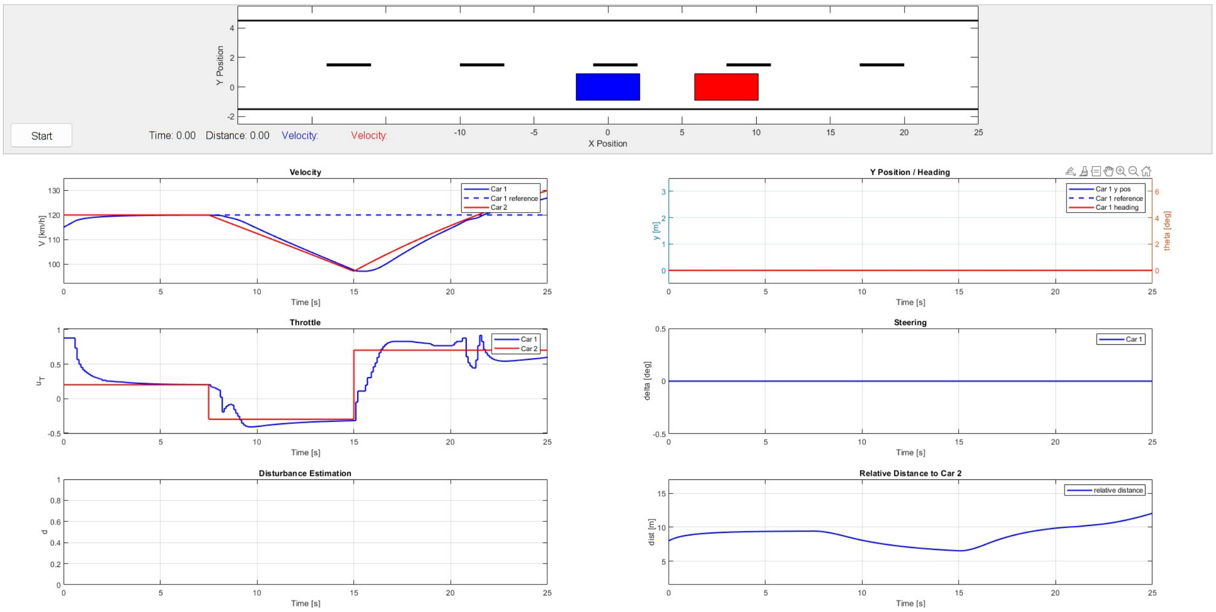


Figure 12: Simulation with variable lead velocity

. Ego car begins from 115km/h and accelerates to track reference speed of 120km/h. Lead car is also moving at 120km/h, so there's no concern for safety distance and ego car is able to track reference velocity. When lead car randomly decelerates, ego car also decelerates to maintain safety distance(which again stays strictly above 6). When lead car accelerates, ego car also accelerates to track reference velocity again.
It has a good ride quality. When lead car is suddenly changing throttle at around 7.5s and 15s, ego car displays smooth motion. This is primarily achieved by having reasonable K. Recall that $u = v + K(\Delta - z)$. If K is too small, then $\mathcal{E}$ will be too big and there will be no control authority left for nominal control v. If K is too large, it will overreact to a sudden change in lead car motion, namely $\Delta - z$ in our formulation.

6 Deliverable 6.1: Nonlinear Model Predictive Control (NMPC)

For this section we didn't have to decompose the car dynamics into independent subsystems. Instead, the Nonlinear MPC takes the full state $\mathbf{x}$ as input and input commands $\mathbf{u}$ are provided.

6.1 Design Procedure

We now proceed to design a controller based on NMPC to track the given reference. Initially, we will focus on setting up the problem to ensure its applicability to a nonlinear dynamic model. Subsequently, we will address the optimization of the various free parameters.

1. **Discretization:** One of the main challenges with nonlinear MPC is the discretization of the nonlinear continuous model. For this purpose, we opted to use the Runge–Kutta 4 (RK4) method. This method is highly precise as it utilizes a second-order Taylor series expansion around the desired state for a given input. The gradient $\dot{x} = f(x, u)$ at a point x for an input u can be obtained from the physical plant using the MATLAB command `car.f(x,u)`. Based on the sampling period T_s, the next sampled state is computed as:

$$x_{k+1} = x_k + T_s \cdot \left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right), \tag{30}$$

where:

$$\begin{aligned}
k_1 &= f(x_k, u_k), \\
k_2 &= f\left(x_k + \frac{h}{2} \cdot k_1, u_k\right), \\
k_3 &= f\left(x_k + \frac{h}{2} \cdot k_2, u_k\right), \\
k_4 &= f\left(x_k + h \cdot k_3, u_k\right).
\end{aligned}$$

2. **Cost Function:** A quadratic cost function was designed to penalize tracking errors and control effort:

$$J = \sum_{k=1}^{N-1} \left((\mathbf{x}_k - \mathbf{r}_k)^\top Q(\mathbf{x}_k - \mathbf{r}_k) + \mathbf{u}_k^\top R\mathbf{u}_k\right) + (\mathbf{x}_N - \mathbf{r}_s)^\top Q(\mathbf{x}_N - r_k) \tag{31}$$

3. **Constraints:** The controller enforces the following constraints:
   - **State constraints:**

$$-0.5 \leq y \leq 3.5, \quad -5^\circ \leq \theta \leq 5^\circ. \tag{32}$$

   - **Input constraints:**

$$-30^\circ \leq \delta \leq 30^\circ, \quad -1 \leq u_T \leq 1. \tag{33}$$

6.2 NMPC Tuning Parameters:

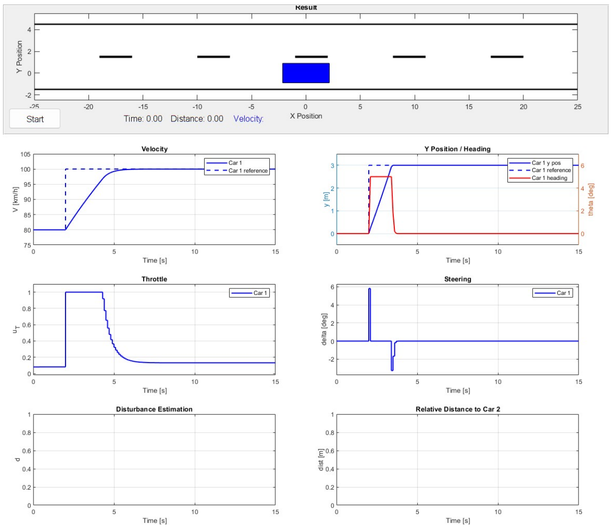
The parameters requiring tuning include the individual coefficients of the diagonal matrices Q and R, which correspond to the weights assigned to the associated states and inputs in the cost function to be minimized, as well as the horizon H.

1. **Horizon Length H:** We observed that H plays a crucial role in balancing the computational cost and the controller's performance. After testing, we chose $N = 15$, as higher values offered minimal performance improvements while significantly increasing the computation time.
2. **Tuning Q and R:** Our initial guess started with the identity matrix, which is $Q = R = \mathrm{diag}(1, 1)$ and it actually works just fine. The settling time is approximately 3 seconds when accelerating from 80 km/h to 100 km/h and the lane changing is done within about 1.5 seconds as shown in Figure 13a.
3. **Tuning Q_f:** Finally, for the terminal cost we used $Q_f = Q$ and we dropped the terminal constraints as we were told to according to the project description.

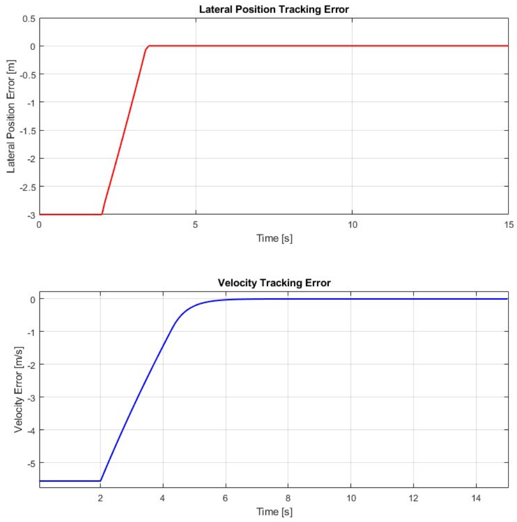
6.3 Steady-State Tracking Error Analysis

There is no steady state tracking error because we are using the exact nonlinear model. Steady state tracking error comes from linearization error.

Figure 13b shows the lateral position and velocity tracking performance. The controller smoothly tracked the reference trajectory with no tracking error while respecting constraints.



(a) Visualization of the simulation results of the NMPC controller.



(b) Lateral position and velocity tracking error.

Figure 13: Simulation results of the NMPC controller.

7 Deliverable 6.2: NMPC for Overtaking Maneuver

In this section, we present the implementation and evaluation of the Nonlinear Model Predictive Control (NMPC) framework to perform an overtaking maneuver. The objective was to enable the ego car to safely overtake another car traveling in the same direction while avoiding collisions.

7.1 Design Procedure

Most of the constraints are the same as in the **Deliverable 6.1** section. We only add an additional ellipsoidal collision avoidance constraint and comfort constraint to achieve an overtaking maneuver that is comfortable for passengers.

1. **Collision Avoidance:** An ellipsoidal safety constraint was implemented to maintain a safe distance between the ego car and the other car:

$$(\mathbf{p} - \mathbf{p}_L)^\top H(\mathbf{p} - \mathbf{p}_L) \geq 1, \tag{34}$$

   where $H = \text{diag}\left(\frac{1}{a^2}, \frac{1}{b^2}\right)$ (a and b as the major and minor semi-axis of the ellipsoid), $\mathbf{p} = [lon, lat]^\top$, is the longitudinal and lateral position of the ego car, and $\mathbf{p}_L$ of the other car. We chose $a = 8$ and $b = 3$ as a and b shown in Figure 14 are approximately 8 and 3. In the simulation, it works well with these parameters.

2. **Comfort Constraint:** A rate constraint with maximum of 0.4 is applied to the input so that the acceleration and deceleration would not be too sharp and thus causing discomfort of passengers. The rate constraint is defined as:

$$|\mathbf{U}_{k+1} - \mathbf{U}_k| \leq \gamma_{max} \tag{35}$$

7.2 NMPC Tuning Parameters

For the overtaking maneuver, we did not modify the horizon length H and the terminal weight Q_f. Only the Q and R is modified.

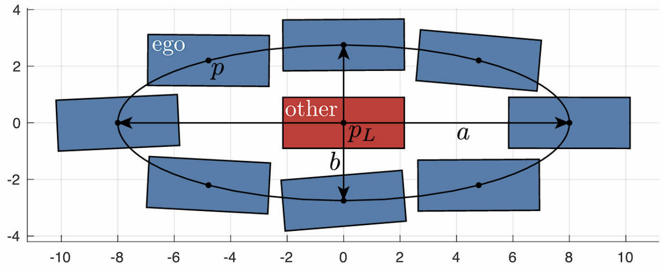
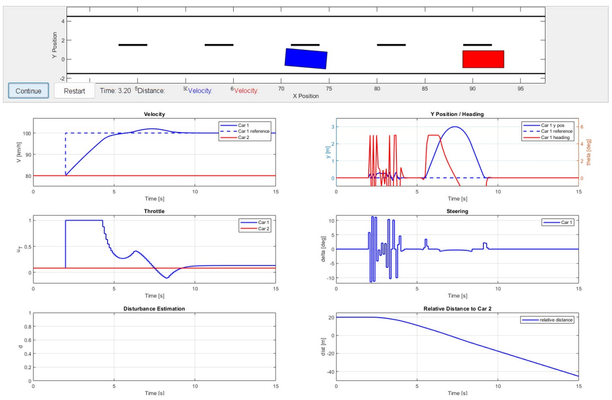
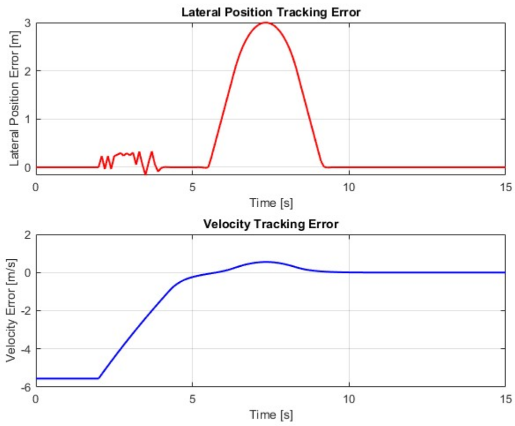


Figure 14: Ellipsoid constraint. Taken from EPFL MPC Course Project Description.

1. **Tuning of Q and R:** Our initial guess for Q and R are $Q = \text{diag}(1, 1)$ and $R = \text{diag}(1, 1)$. But this results in an oscillating behavior around $T = 3$ as shown in Figure 15a. To eliminate this behavior, we decide to modify the weights to $Q = \text{diag}(0.1, 1)$ and $R = \text{diag}(10, 1)$ to penalize more on the velocity tracking error and the steering input. The results will be shown in the **Results and Observation** section.



(a) Visualization of the simulation results. An oscillation behavior is observed around $T = 3$.

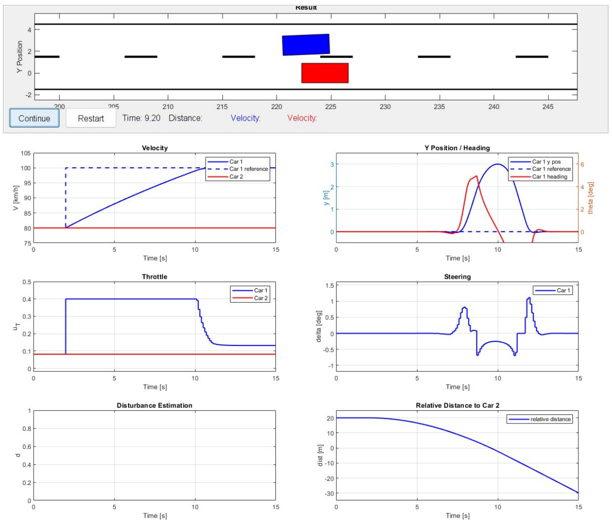


(b) Lateral position and Velocity tracking error.

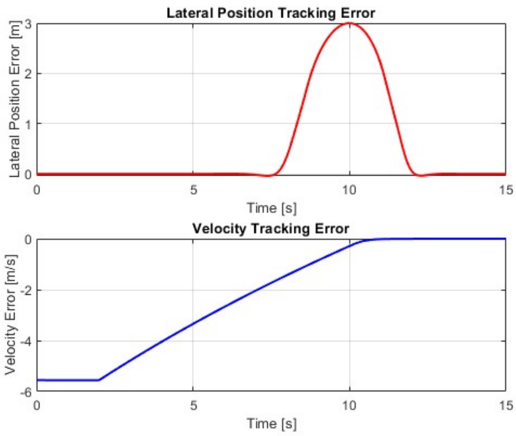
Figure 15: Simulation results of the NMPC controller with Q and R as identity matrices.

7.3 Results and Observations

1. The NMPC controller successfully enabled the ego car to overtake the other car without collisions as shown in Figure 16a.
2. The ellipsoidal collision constraint ensured a safe distance between the cars during the maneuver.
3. Tracking errors for y and V were minimized, indicating accurate reference trajectory tracking.
4. Smooth control inputs (δ, u_T) ensured stability and driver comfort.



(a) Visualization of the simulation results of overtaking maneuver.



(b) Tracking error of lateral position and velocity during overtaking maneuver.

Figure 16: Simulation results of the NMPC controller with tuned Q and R.