MPC Project

Members Christy EL Skaff
Charbel EL Khoury
Marc Mouawad
Instructor Dr. Colin Jones

1 Introduction

This report explores the development of a Model Predictive Control (MPC) cruise controller tailored for a car navigating a highway scenario, with all simulations executed in MATLAB. The car model, including its non-linear dynamics, is provided and serves as the foundation for the project.In Section 2, the car dynamics are linearized and then decomposed into longitudinal and lateral subsystems, enabling focused control strategies. Section 3 presents the design of tracking MPC controllers for each subsystem, ensuring smooth and responsive driving behavior. Section 4 introduces an offset-free MPC controller to address practical MPC issues such as disturbances and tracking. Section 5 describes the implementation of a robust tube MPC controller for adaptive cruise control, which ensures safe distances from other vehicles despite uncertainties. Section 6 transitions to the design of a non-linear MPC (NMPC) controller using RK4 discretization, showcasing its application for tasks such as highway cruising and overtaking maneuvers.

2 Linearization

In this section, the car's dynamic model and its linearization are presented. Moreover, the model will be divided into two subsystems: longitudinal and lateral.

2.1 Deliverable 2.1: Analytical Expressions

The state vector is given by:

$$x = \begin{bmatrix} x \\ y \\ \theta \\ V \end{bmatrix} = \begin{bmatrix} m \\ m \\ rad \\ m/s \end{bmatrix} \tag{1}$$

Where x and y represent the position of the car's center of mass, θ is the heading angle of the car with respect to the x-axis, and V is the velocity of the vehicle.

The control input is $u = [\delta, u_T]^T = [rad, -]^T$ where δ is the steering angle of the front wheels, and u_T is the normalized throttle to the motor limited to $-1 \leq u_T \leq 1$.

The kinematics and dynamics of the vehicle are also given. The slip angle β is the angle between a wheel's actual direction of travel and the direction in which it is pointed. It is defines at the center of mass and computed based on the steering geometry:

$$\beta = \arctan(\frac{l_r \tan(\delta)}{l_r + l_f}) \tag{2}$$

where l_r and l_f are the distances from the center of mass to the rear and front axles, respectively. The longitudinal dynamics are influenced by the motor force, aerodynamic drag, and roll resistance:

$$\begin{aligned} F_{motor} &= u_T \frac{P_{max}}{V} \\ F_{drag} &= \frac{1}{2} C_d A_f V^2 \\ F_{roll} &= C_r mg \end{aligned} \tag{3}$$

where m is the vehicle mass, P_{max} is the maximum motor power, ρ is the air density, C_d is the drag coefficient, A_f is the frontal area of the vehicle, C_r is the roll coefficient, and g is the gravity constant.

Therefore, the complete state equations can be derived:

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} V\cos(\theta+\beta) \\ V\sin(\theta+\beta) \\ \frac{V}{l_r}\sin(\beta) \\ \frac{F_{mototr}-F_{drag}-F_{roll}}{m} \end{bmatrix} \tag{4}$$

Cruise control aims to maintain a constant target velocity V_ref but the position x always increases over time since the velocity is positive. Therefore, there is no true steady state for the longitudinal subsystem since x does not stabilize to a constant value. We will then focus on the remaining states (y,θ,V) for which steady-state conditions can be achieved and the steady-state subsystem can be given by:

$$f_s(x_s,u_s) = \begin{bmatrix} \dot{y} \\ \dot{\theta} \\ \dot{V} \end{bmatrix} = 0 \tag{5}$$

By applying a Taylor Series' expansion, we are able to linearize the system as to obtain:

$$\begin{aligned} &f(x,u) \approx f(x_s,u_s) + A(x-x_s) + B(u-u_s) \\ &x_s = [0,0,0,V_s] \\ &u_s = [0,u_{T,s}] \end{aligned} \tag{6}$$

Moreover we find the linearized A and B matrices as:

$$A = \frac{\partial f(x,u)}{\partial x}\Big|_{x_s,u_s} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{m}[\frac{u_{T,s}P_{max}}{V_s^2} + \rho C_d A_f V_s] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma \end{bmatrix}$$

$$B = \frac{\partial f(x,u)}{\partial u}\Big|_{x_s,u_s} = \begin{bmatrix} 0 & 0 \\ V_s\frac{l_r}{l_r+l_f} & 0 \\ V_s\frac{1}{l_r+l_f} & 0 \\ 0 & \frac{P_{max}}{mV_s} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ a & 0 \\ b & 0 \\ 0 & c \end{bmatrix}$$

Resulting in the linearized model:

$$f(x,u) = \begin{bmatrix} V_s \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ V - V_s \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ a & 0 \\ b & 0 \\ 0 & c \end{bmatrix} \begin{bmatrix} \delta \\ u_T - u_{T_s} \end{bmatrix}$$

2.2 Deliverable 2.2: Subsystem separation

Note that at steady state:

$$x_s = Ax_s + Bu_s$$

Moreover, when looking at the system matrices, it can be noted that the car's dynamics can be decomposed into independent longitudinal and lateral subsystems due to the minimal coupling present between them. The longitudinal motion is controlled primarily by the throttle or brake, which affects the car's velocity and forward position. For instance, pressing the accelerator will increase the speed without directly affecting the car's direction. Similarly, lateral motion is controlled by steering the wheels, which adjusts the car's direction and lateral position. This does not significantly alter the speed of the car. Note that the velocity that appears in both subsystems acts a scalar magnitude that only affects the general motion of the car.

Hence, longitudinal and lateral sybsystems will follow the dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} Vs \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & \gamma \end{bmatrix} \begin{bmatrix} x \\ V - V_s \end{bmatrix} + \begin{bmatrix} 0 \\ c \end{bmatrix} u_T \tag{7}$$

$$\begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & V_s \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \theta \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} \delta \tag{8}$$

3 MPC Controllers Design for Each Sub-System

3.1 Problem Formulation:

In this section, we will design a recursively feasible and stabilizing MPC controller for each sub-system. We will be using the tracking formulation with state $\bar{x} - \bar{x_s}$and input $u - u_s$ giving us the standard discretized linear dynamics:

$$\bar{x}^+ = f_d(\bar{x_s}, u_s) + \bar{A}_d(\bar{x} - \bar{x_s}) + \bar{B}_d(u - u_{T,s}) \tag{9}$$

Discretized using $\dot{x} = \frac{x^+ - x}{T_s}$, with $T_s = 0.1$. The matrices are updated accordingly.

The goal of MPC is to optimize our system over an infinite horizon, while satisfying constraints and ensuring stability and optimal performance. However, the use of an infinite horizon requires the storage of an infinite number of variables and so the computational cost to compute such a problem is very high. Therefore, we approximate the problem to a finite horizon where the cost function becomes:

$$\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & \sum_{i=0}^{N-1} l(x_i - x_s, u_i - u_s) + V_f(x_N - x_s) \\
\text{subject to:} \quad & x_{i+1} = A_d(x_i - x_s) + B_d(u_i - u_s) + f_d(x_s, u_s), \quad i = 0, \ldots, N-1 \\
& x - x_s \in \mathbf{X} \\
& x_N - x_s \in X_f \qquad (*) \\
& u - u_s \in \mathbf{U}
\end{aligned}$$

The terminal cost V_f along with the terminal constraint (*) are added to the problem to simulate an infinite horizon and ensure recursive feasibility as the terminal set X_f is designed to be positively invariant. For stability, quadratic cost will be chosen so that the system admits a Lyapunov function. The problem becomes:

$$\begin{aligned}
\min_{x \in \mathbb{R}^n} \quad & \sum_{i=0}^{N-1} (x_i - x_s)^T Q (x_i - x_s) + (u_i - u_s)^T R (u_i - u_s) + (x_N - x_s) P (x_N - x_s) \\
\text{subject to:} \quad & x_{i+1} = A_d(x_i - x_s) + B_d(u_i - u_s) + f_d(x_s, u_s), \quad i = 0, \ldots, N-1 \\
& x - x_s \in \mathbf{X} \\
& x_N - x_s \in X_f \qquad (*) \\
& u - u_s \in \mathbf{U}
\end{aligned}$$

3.2 Steady State Target Computation:

In this project, the goal is to track a given reference, in order to do that, a steady state target is computed. Using the steady state identity and the new reference, we find for the longitudinal subsystem:

$$V_{s,ref} = V_{ref}$$

$$u_{Ts,ref} = \frac{1 - \gamma}{B_{d,21}} (V_{ref - V_s}) + u_{T,s}$$

As for the lateral subsystem we find:

$$y_{s,ref} = y_{ref}$$

$$\delta_{s,ref} = \delta_s + \frac{B^T}{B^T B} (A(x_{ref} - x_s))$$

3.3 Terminal Set and constraints:

The problem from N to infinity is an unconstrained Linear Quadratic Regulator (LQR) formulated as a tracking problem with P being the solution to the discrete-time algebraic Riccati equation. The control law is then computed as: $u_{LQR} = K_{LQR}(x - x_s) + u_s$ with K representing the gain that allows the states to track accordingly.

As mentioned, a different MPC controller is implemented for each sub-system. The controlled invariant set was computed using the same algorithm for both, where polytopic projection was integrated (Algorithm (1)).

Algorithm 1: Conceptual Algorithm to Compute Invariant Terminal Set

Input: $\mathcal{X}$
Output: $\mathcal{X}_{\{}$
$\Omega_0 \leftarrow \mathcal{X}$;
while *True* **do**
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$;
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{X}_f = \Omega_i$;
  end
end

The constraints imposed on each subsystem are respectively provided for the longitudinal subsystem and lateral subsystem:

$$
\begin{aligned}
&-0.5m \leq y \leq 3.5m \\
&|\theta| \leq 5^o = 0.0873rad \\
&|\delta| \leq 30^o = 0.5236 \\
&-1 \leq u_T \leq 1
\end{aligned}
$$

It is also important to note that the tuning parameters Q,R,and H significantly influence the performance of the MPC controller. The weighting matrix Q penalizes deviations of the state of the system from their reference values. Higher weights in Q on velocity V, for instance, ensure that the car tracks the desired velocity more precisely and ensures the velocity reaches the reference quickly, reducing settling time, while lower weights on Q allow the position to drift slightly without high penalty. On the other hand, the matrix R penalizes high values on the control inputs. A higher R penalizes the input with higher costs. It smoothens the control action at the cost of slow response to changes in the system. Lower weight on R makes the control action more aggressive. Finally, the horizon length H determines how far into the future the controller predicts the system's behavior. Long-horizon improve performance, allowing for smooth responses, and ensures stability, at the cost of high computational complexity. Lower horizons, on the other hand, reduce the computational cost at the cost of limited future predictions. If the horizon is too small, it might also introduce instability to the system's response. Therefore, it is best to choose a horizon length that is neither too large nor too low. Table (1) illustrates the parameters for each subsystem.

Parameters	**Q**	**R**	**H (s)**
Lateral Subsystem	50	5	20
Longitudinal Subsystem	50	5	20

Table 1: Tuning Parameters for Both Subsystems

Moreover, it is important to note that the terminal set should be shifted to a new origin. Therefore, it will initially be defined as:

$$
\begin{bmatrix} F \\ MK_{LQR} \end{bmatrix} x \leq \begin{bmatrix} f \\ m + M(Kx_s - u_s) \end{bmatrix}
$$

Where $X := [Fx \leq f]$ and $U := [Mu \leq m]$.

Figure (1) shows the terminal invariant set computed for the lateral subsystem. As mentioned, this set was computed iteratively with an LQR gain K derived from the parameters Q and R.

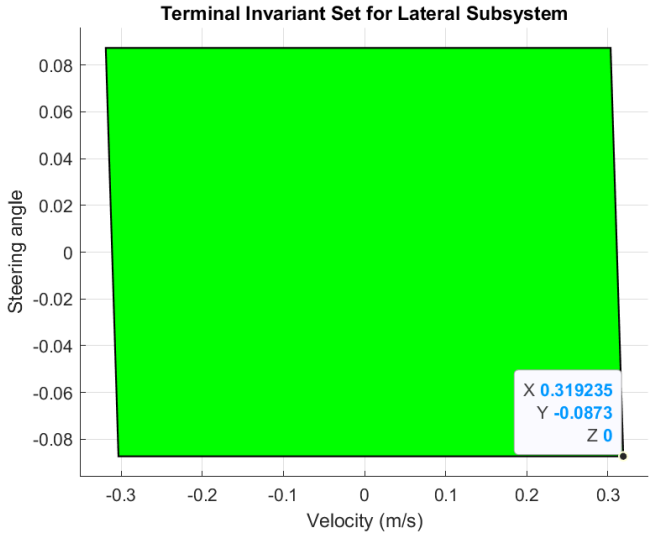

Figure 1: Terminal Set of the lateral subsystem

3.4 Experiment and Results:

For the experiment, a simulation is run with two references given to the controller:

- Reference 1: A velocity of 80 km/h, with y = 0m.
- Reference 2 (after 5 seconds): A velocity of 120 km/h, with y = 3m.

Figures(2)-(3)-(4)-(5) show the results of the simulation.

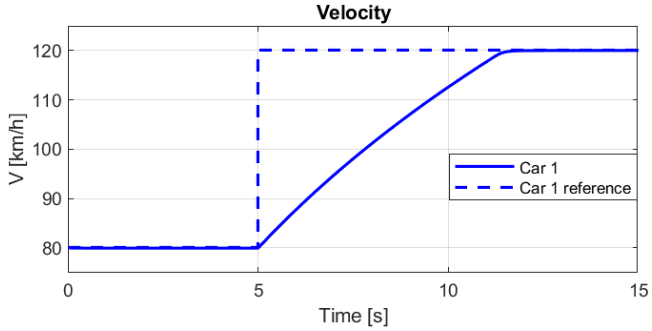

Figure 2: Velocity

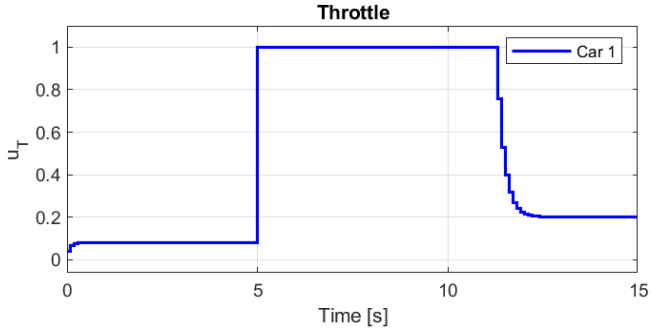

Figure 3: Throttle Input

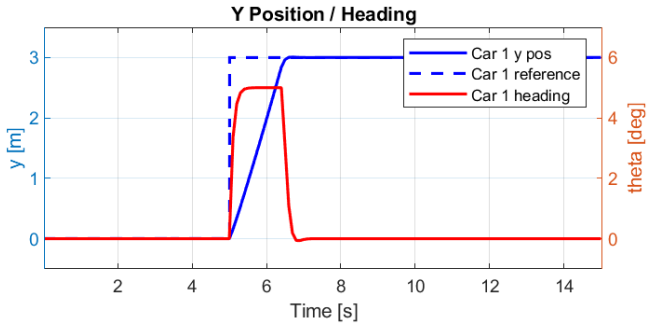

Figure 4: Lateral position and orientation

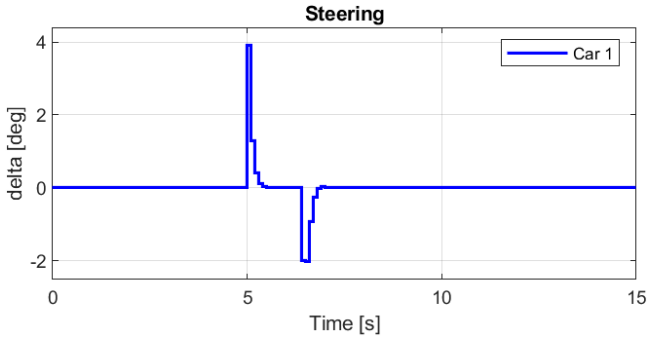

Figure 5: Steering angle

It is clear from the figure that both controllers meet the settling time specifications, where $t_s \approx 6.5s$ in the longitudinal case, and $t_s \approx 1.5s$ in the lateral case.

4 Offset-Free Tracking

In this section, the problem of offset-free tracking is dealt with. The main problem with the previously designed controllers, is that when tracking a reference different from the linearization point, a steady state error will remain due to the linearization offset. Hence, in this section, the state space of the longitudinal system is augment to account for disturbance rejection by estimating it using a Luenberger Observer for the longitudinal subsystem. Also note that for the experiments, the estimated states of the observer will be used for tracking.

4.1 State Space and Steady-State target updates:

The state space is augmented to account for the disturbance, assumed constant. The dynamics of the system will be as such:

$$\begin{aligned} x^+ &= f_d(x_s, u_s) + A_d(x - x_s) + B_d(u - u_s) + \hat{B}_d d \\ d^+ &= d \end{aligned} \tag{10}$$

Note that this equation is added as an equality constraint to the previous optimization problem, replacing the system dynamics constraints of part 3.

Moreover, for smoother change in inputs we add the following term to the objective to penalize sharp changes in inputs: $S_u(u_{i+1} - u_i)$, where $S_u = 5$ is the smoothness factor.

Taking this into account, the steady state target of the longitudinal controller is changed accordingly:

$$V_{s,ref} = V_{ref}$$

$$u_{s,ref} = u_s + \frac{(1-\gamma)(V_{ref} - V_s) - \hat{B}_d \hat{d}}{B_{d,21}}$$

Note that $\hat{B_d}$ should be chosen so that the system is observable, in this case a value of $\hat{B_d} = 1$ as it satisfies the criterion.

4.2 Observer Design:

For the observer design, we look at the following system, with $z = [x - x_s, d - d_s]^T$ and $d_s = \hat{d}$:

$$\hat{z}^+ = \begin{bmatrix} \gamma & \hat{B}_d \\ 0 & 1 \end{bmatrix} (\hat{z} - \hat{x}_s) + B_d(u - u_s) + L(C\hat{z} - y) + \begin{bmatrix} f_d(x_s, u_s) \\ 0 \end{bmatrix} \tag{11}$$

where $L = [-0.4976 - 0.06]$ represents the observer gains, chosen so that the observer poles are [0.7,0.8]. The observer represent the convergence speed of the estimates. The smaller they are the faster the convergence, however the tradeoff is that it would be much more sensitive to sensor noise. The chosen poles result in a fair trade between convergence speed and disturbance rejection.

Note that the values of Q,R, and horizon are kept the same as in part 3, so that the observer is the only variable in the following experiment.

4.3 Experiment and Results:

To test the design, the following simulation is performed:

- Reference 1: [0, 80/3.6].
- Reference 2 (after 2 seconds): [3 50/3.6].

Note that the linearization point is $V = \frac{120}{3.6} m/s$.
Figures(6)-(7)-(8)-(9)-(10) show the results of the simulation.

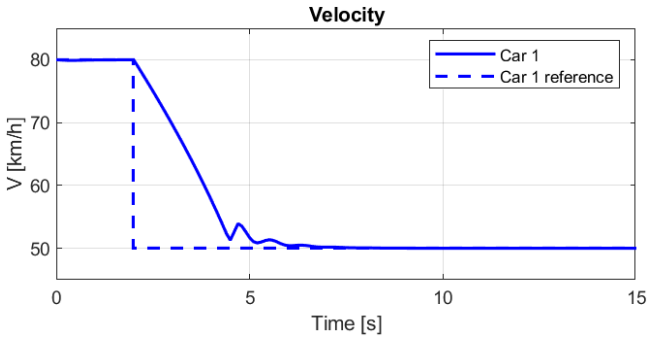

Figure 6: Velocity

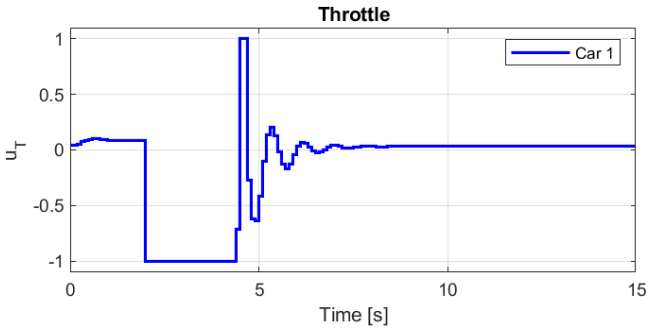

Figure 7: Throttle Input

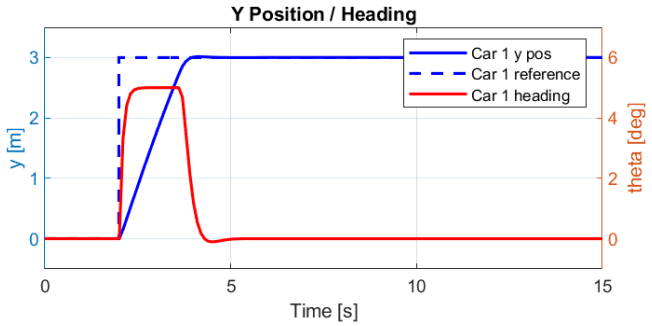

Figure 8: Lateral position and orientation

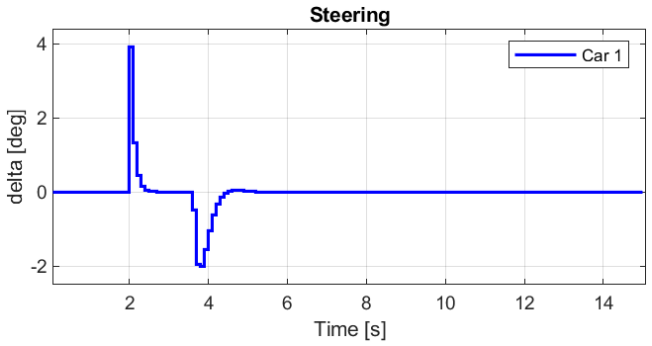

Figure 9: Steering angle

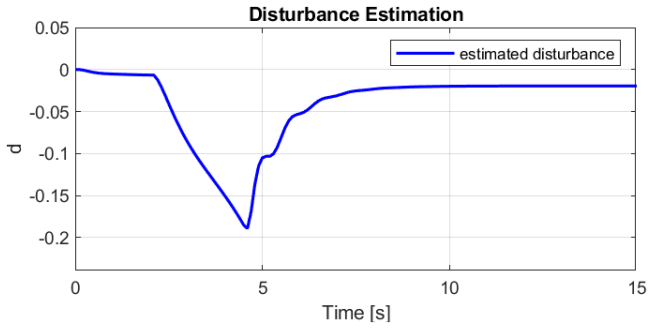

Figure 10: Estimated Disturbance

It is clear from the figure that both controllers meet the settling time specifications for both systems, moreover, we can see the disturbance estimate converging as the controller gets closer to settling time. Once the estimate converges, we also observe zero steady state error. However, as the controller is close to settling time, when the disturbance has not yet converged, we observe a few oscillations.

Moreover, by comparing the input δ at around 4.2 seconds, with the one from part 3, one can observe that the convergence is smoother. Since for the lateral subsystem the design was left unchanged, with the exception of the objective, this smoothness was introduced from the smoothing term that was added.

5 Robust Tube MPC for Adaptive Cruise Control

In this section, a robust tube MPC is designed, to keep a minimum distance to a car in front.

5.1 Relative Dynamics derivation:

The controlled car is referred to as the "ego" car, while the one in front is the "lead" car. Both cars are assumed to follow the same dynamics given by:

$$\tilde{x}^+ = f_d(x_s, u_s) + A_d(\tilde{x} - x_s) + B_d(\tilde{u} - u_s) \tag{12}$$

$$x^+ = f_d(x_s, u_s) + A_d(x - x_s) + B_d(u - u_s) \tag{13}$$

In order to solve this problem, the relative dynamics are first computed. Let $\Delta = [\Delta_x; \Delta_V] = \tilde{x}_{long} - x_{long} - x_{safe}$ be the relative state representing the longitudinal difference between the ego and the lead car states with $x_{safe} = [x_{safe,pos}; 0]$ is a steady safe point behind the lead car chosen by design.

To compute the relative dynamics, the equation of motion of the lead and ego cars are subtracted from each other:

$$\begin{aligned}
\tilde{x}^+ - x^+ &= f_d(x_s, u_s) + A_d(\tilde{x} - x_s) + B_d(\tilde{u} - u_s) - f_d(x_s, u_s) - A_d(x - x_s) - B_d(u - u_s) \\
\tilde{x}^+ - x^+ &= A_d(\tilde{x} - x) - B_d u + B_d \tilde{u} + A_d x_{safe} - A_d x_{safe} \\
\tilde{x}^+ - x^+ - x_{safe} &= A_d(\tilde{x} - x - x_{safe}) - B_d u + B_d \tilde{u} \\
\Delta^+ &= A_d \Delta - B_d u + B_d \tilde{u}
\end{aligned} \tag{14}$$

For the rest of this section, these equations of motion will be considered for the design of the controller.

5.2 Minimal Robust Invariant Set computation:

Since the behavior of the lead car is unknown, the lead car will be assumed to change its throttle by up to 0.5, and the input $\tilde{u_T}$ will be considered a bounded disturbance input. It will lie in the set $\mathcal{W} = [u_{T,s} - 0.5, u_{T,s} + 0.5]$, where $u_{T,s}$ is the linearization steady state throttle input.

As a first step, the nominal system containing tube centers is defined for the set computation as:

$$\Delta_n^+ = A_d \Delta_n - B_d u_{T,n} \tag{15}$$

Then the control policy is defined as:

$$\mu_{tube}(\Delta) = K(\Delta - \Delta_n) + u_{T,n} \tag{16}$$

Where K is the controller K stabilizing the following error Dynamics:

$$\Delta^+ - \Delta_n^+ = (A_d - B_d K)(\Delta - \Delta_n) + B_d \tilde{u_T} \tag{17}$$

In this case, K is computed as the infinite horizon LQR controller using the MATLAB function dlqr, with $Q = 50I$ and R = 5. The choise of Q and R is chosen for a quick convergence of the states while still adding a small weight on the magnitude of the input to avoid extreme values. The stabilizing condition verified by computing the spectral radius of the matrix $A - BK$: $\rho(A - BK) < 1$, this condition makes sure that all poles of the discrete system lie within the unit circle.

Once K is chosen, the minimal robust invariant set is initialize to $B_d * \mathcal{W}$ and Algorithm(2) is followed, to find the minimal robust invariant inside of which the error lies.

Algorithm 2: Minimal Robust Invariant Set Computation

Input: $A_d - B_d K$
Output: $\mathcal{E}_\infty$
$\Omega_0 \leftarrow B_d \mathcal{W}$;
while *True* **do**
  $\Omega_{i+1} \leftarrow \Omega_i \oplus (A_d - B_d K)^i$;
  if $||(A_d - B_d K)^i|| < 0.01$ **then**
    return $\mathcal{E}_\infty = \Omega_i$;
  end
  i=i+1
end

After following the explained procedure, for a controller K = [2.3305;2.8544], and the minimal robust invariant set can be found in Figure(11)

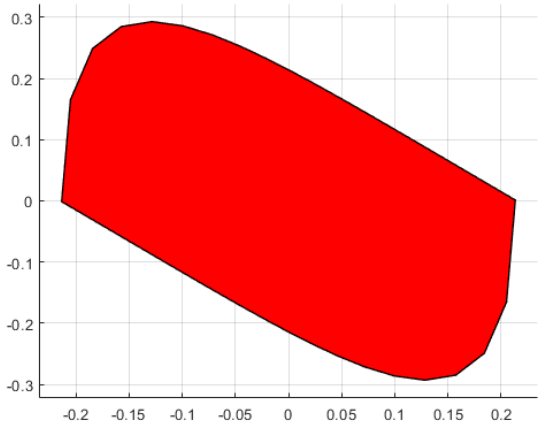

Figure 11: Minimal Robust invariant Set

5.3 Constraint Tightening:

The constraint on the states in this part is the relative distance between the two vehicles required to be greater than 6m. The equivalent constraint on the relative state would be:

$$\Delta_x \geq 6 - x_{safe,pos} \tag{18}$$

In this part of the project $x_{safe,pos} = 10$, meaning that the feasible set for the relative state will be:
$\mathcal{X} := [\Delta_x \geq -4]$. The safe steady-state point was chosen so that it is not too conservative, but also allows the terminal set to contain the origin so that the problem is feasible.

Concerning input constraints, the set will be: $\mathcal{U} := [-1 \leq u_T \leq 1]$.

We define the tightened constraints as:

$$\begin{aligned} \mathcal{X}_t &= \mathcal{X} \ominus \mathcal{E}_\infty \\ \mathcal{U}_t &= \mathcal{U} \ominus K\mathcal{E}_\infty \end{aligned} \tag{19}$$

5.4 Terminal Set computation:

For the terminal set computation, the terminal controller is chosen to be the same as the one designed for the minimally robust invariant set. Moreover, Algorithm(3) is the one that was followed to compute the terminal set $\mathcal{X}_f$:

Algorithm 3: Conceptual Algorithm to Compute Invariant Set

Input: $\mathcal{X} \ominus \mathcal{E}_\infty$
Output: $\mathcal{X}_f$
$\Omega_0 \leftarrow \mathcal{X} \ominus \mathcal{E}_\infty$;
while *True* **do**
  $\Omega_{i+1} \leftarrow \text{pre}(\Omega_i) \cap \Omega_i$;
  if $\Omega_{i+1} = \Omega_i$ **then**
    return $\mathcal{X}_f = \Omega_i$;
  end
end

Figure(12) shows the terminal set that was found, which as can be seen contains the origin [0,0].

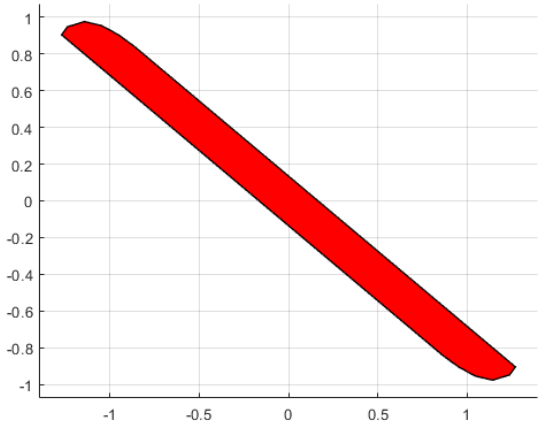

Figure 12: Terminal Set for the Relative dynamic system

5.5 Nominal System problem formulation:

After computing the previous sets offline, they are loaded once the control law computation is needed. For the latter, a nominal MPC problem is formulated for the nominal system as follows:

Feasible set:

$$\mathcal{Z}(x_0) := \left\{ \mathbf{z}, \mathbf{v} \;\middle|\; \begin{array}{ll} \Delta_{n,i+1} = A\Delta_{n,i} - Bu_{T,i} & i \in [0, N-1] \\ \Delta_{n,i} \in \mathcal{X} \ominus \mathcal{E}, & i \in [0, N-1] \\ u_{T,i} \in \mathcal{U} \ominus K\mathcal{E}, & i \in [0, N-1] \\ \Delta_{n,N} \in \mathcal{X}_f, & \\ x_0 \in z_0 \oplus \mathcal{E} & \end{array} \right\}$$

Cost function:

$$V(\mathbf{\Delta}_\mathrm{n}, \mathbf{u}_{\mathrm{T,n}}) := \sum_{i=0}^{N-1} \ell(\Delta_{n,i}, v_i) + V_f(\Delta_{n,N})$$

Optimization problem:

$$(\mathbf{v}^*(x_0), \mathbf{z}^*(x_0)) = \arg\min_{\Delta_\mathrm{n}, \mathbf{u}_{\mathrm{T,n}}} \{V(\mathbf{\Delta}_\mathrm{n}, \mathbf{u}_{\mathrm{T,n}}) \mid \mathbf{\Delta}_\mathrm{n}, u_{T,n} \in \mathcal{Z}(x_0)\}$$

Control law:

$$\mu_{\mathrm{tube}}(x) := K\left(\Delta - \Delta^*_{n,0}(x)\right) + u^*_{T,0}(x)$$

5.6 Simulation and Results:

To test the designed controller, experiments were conducted:

- In the first one, the lead car starts 10 meters in front of the ego car, and has a constant speed.
- The second one test the full range of uncertainty of the controller with the change of throttle input.
- For both experiments, the values of Q and R mentioned previously were used, along with a horizon of 20.

Figures (13)-(14)-(15) show the results of the first experiment.

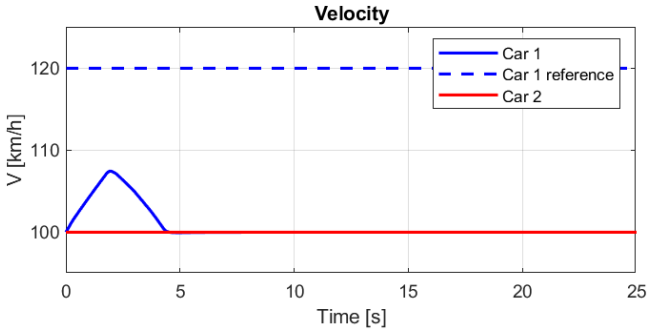

Figure 13: Speed of the ego and lead cars

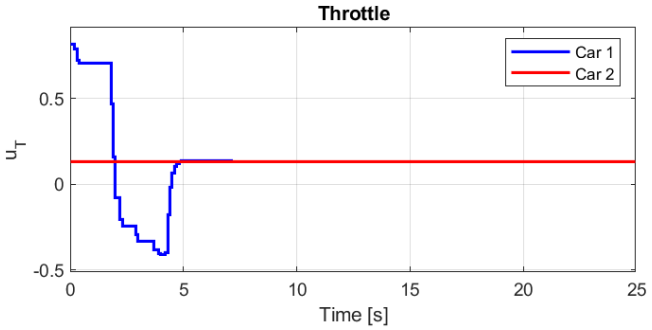

Figure 14: Throttle input for the ego car

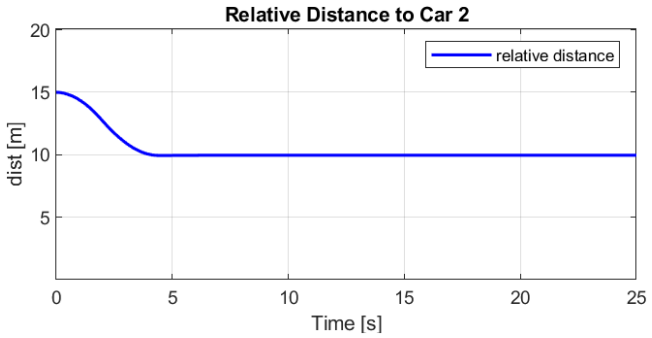

Figure 15: Relative distance between the cars

We see that in the first instant we have some accelerations so that the ego car can meet the minimum distance specifications, then at around 4 seconds we reach a steady state where the relative distance is of 10 meters.

As for the second experiment, Figures (16)-(17)-(18) show the results.

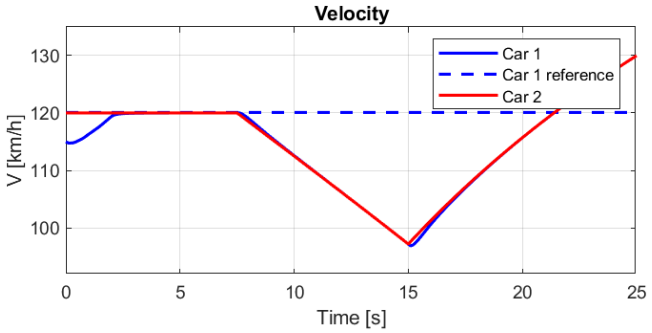

Figure 16: Speed of the ego and lead cars

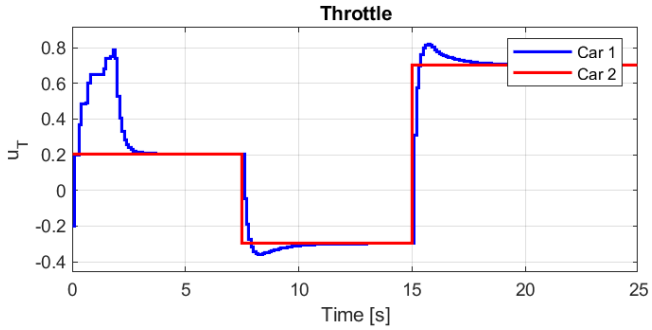

Figure 17: Throttle input for the ego car

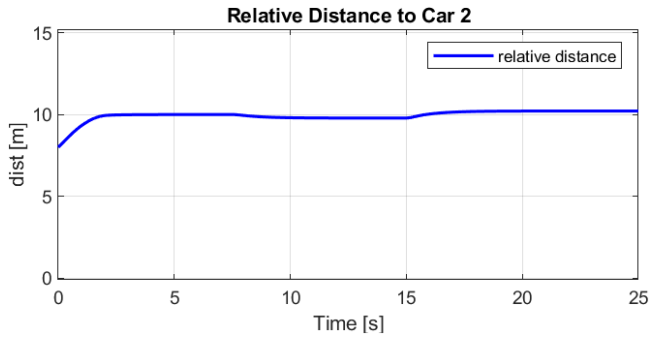

Figure 18: Relative distance between the cars

Once again, we see that the minimum distance requirement is met even though we observe more than one change in the input of the lead car. More importantly, the ego car does not undergo abrupt changes in the velocity in response to the change in disturbance, highlighting a good ride quality.

6 Nonlinear MPC

In this section, a controller is designed taking into account the nonlinearity of the system dynamics. The controller is able to track a reference speed and position, and overtaking other cars. The implementation is done using CasADi solver in MATLAB.

6.1 Deliverable 6.1: Nonlinear MPC Tracking Controller Design

The first step in order to design the NMPC controller is to discretize the system. This is done using the Runge_Kutta 4 (RK4) method. The RK4 discretization method is a numerical technique for approximating the solutions of ODEs by using weighted averages of slopes at intermediate points within each time step. It provides higher accuracy compared to simpler methods like Euler's, making it particularly suitable for systems requiring precise solutions. The discretization algorithm is:

$$x_{k+1} = x_k + h \times (\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}) \tag{20}$$

knowing that:

$$k_1 = f(x_k, u_k)$$

$$k_2 = f(x_k + \frac{h}{2} \times k_1, u_k)$$

$$k_3 = f(x_k + \frac{h}{2} \times k_2, u_k)$$

$$k_4 = f(x_k + h \times k_3, u_k)$$

where h is the sampling time.

In this section, the controller needs to track reference velocity and y position. This is achieved using the same approach as the linear controllers. The tuning parameters are the horizon length H, as well as the state and input cost matrices, Q and R. The horizon length is set to 5, as a trade-off between performance and computational

time. This length presented the same performance achieved by the system for H set to 10, 15 and 20, showing no tracking error and converging to the needed solution. Therefore, it was kept at 5 to minimize the needed time to run the controller.

For the cost matrices, Q and R were first set as identity matrices. To achieve tracking of the reference speed and position, in a settling time less than 5 seconds, the cost of these two states is increased to put higher weight and reach the reference in a faster time. The rest of the entries of the Q matrix are set to 0 as there is no cost directly ties to the x position or car heading. The Q matrix is then set as:

$$Q = \text{diag}([0, 30, 0, 30])$$

On the other hand, the entries of the R matrix are tuned to achieve the smoothest changes of speed steering. Through consecutive tests, the best values of the input costs were found to be:

$$R = \text{diag}([5, 5])$$

The cost to be minimized is then set as the quadratic cost weighted by the defined matrices above. The constraints for the states and inputs are the same as the one implemented in Part 3. An additional constraint is used to implement the RK4 discretization relating consecutive states.

The implemented controller reaches the speed reference in 3 seconds and the y-position reference in 1.5 seconds. The different plots show that the constraints are satisfied.

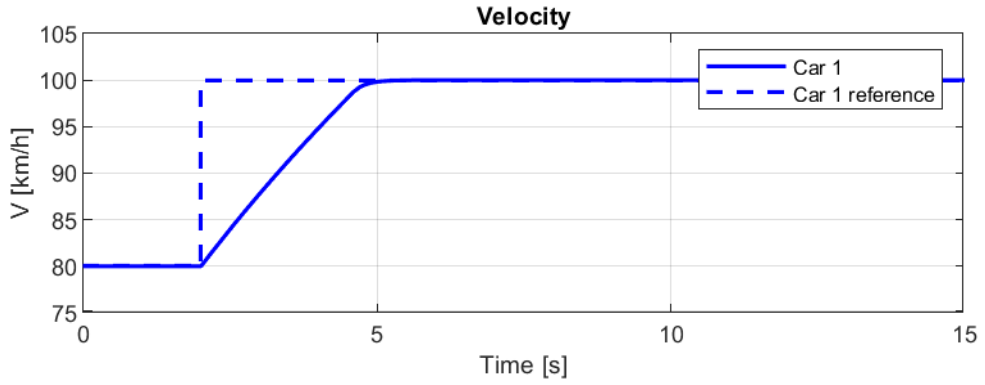

Figure 19: Speed tracking for the nonlinear controller

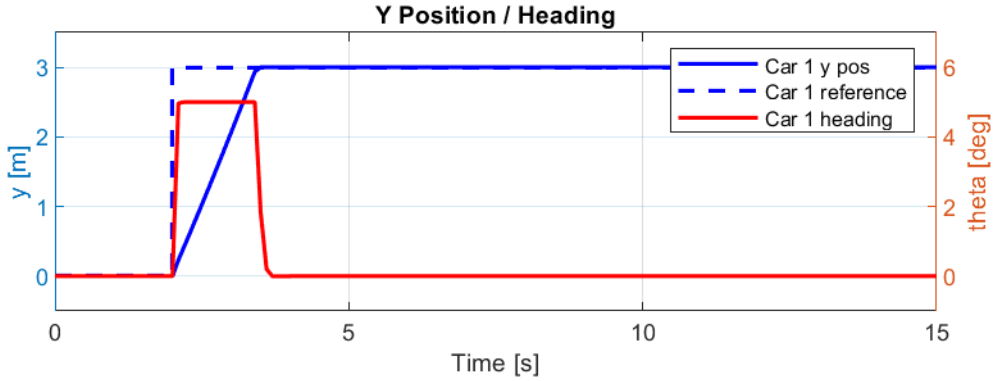

Figure 20: Y position tracking for the nonlinear controller

6.2 Deliverable 6.2: NMPC Controller Design for Overtaking

To implement the overtaking maneuver, an additional constraint is needed to keep the cars from colliding. This is done by implementing an elliptic boundary of the form:

$$(p - p_L)^T H (p - p_L) \geq 1$$

where p is the position of the ego car, p_L that of the other car, and H is the matrix defining the matrix size. H is taken to be a diagonal matrix imposing longitudinal and latitudinal constraints. The entries are chosen as:

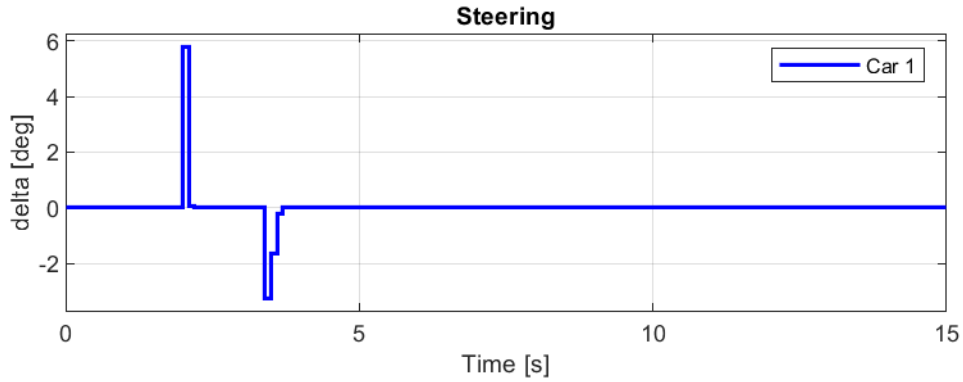

Figure 21: Steering input for the nonlinear controller

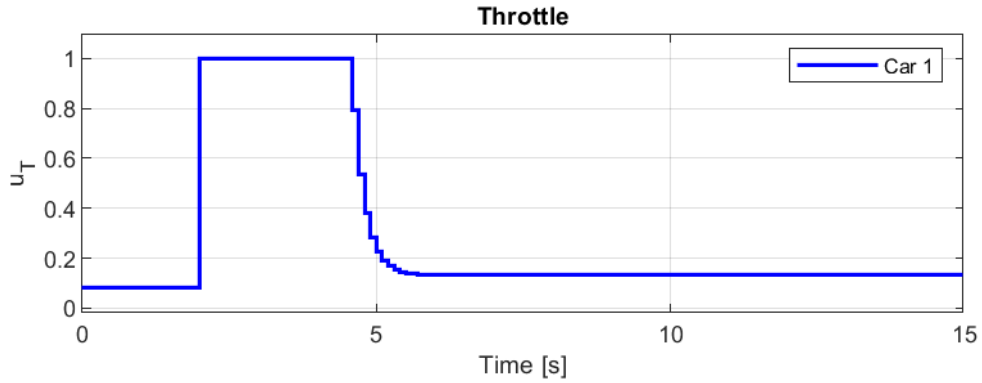

Figure 22: Throttle input for the nonlinear controller

$\begin{bmatrix} \frac{1}{25} & 0 \\ 0 & \frac{1}{9} \end{bmatrix}$, to ensure a safety distance on both dimensions. Additionally, the code from Section 6.1 ensures that the ego car respects the reference tracking. Additional constraints restricting the difference between consecutive inputs to ensure smooth driving and less jerk. They are of the form:

$$-0.1 \leq \delta_{t+1} - \delta_t \leq 0.1$$

$$-0.5 \leq u_{t+1} - u_t \leq 0.5$$

Also, a terminal cost is added to the cost function to ensure convergence of the solution in a smaller horizon. Moreover, due to the complexity of the problem, a larger horizon is needed. The horizon length is chosen to be 8 to ensure finding a solution while minimizing computational time. Through tuning for an optimal performance, the cost for the steering is input is also augmented to 30, to minimize change of directions during the overtake.

As we can see in Fig. 23 and 24, the ego car reaches the reference speed and then speeds up during the overtaking process, before returning to the specified speed. It also keeps the specified y-position until it is necessary to avoid collision, before returning to its lane after clearing the other car.

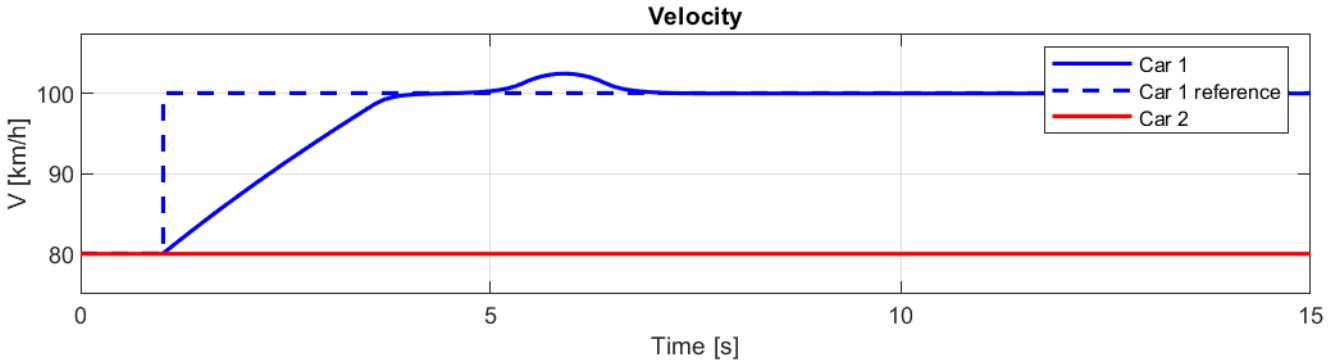

Figure 23: Speed tracking for the nonlinear controller with overtaking

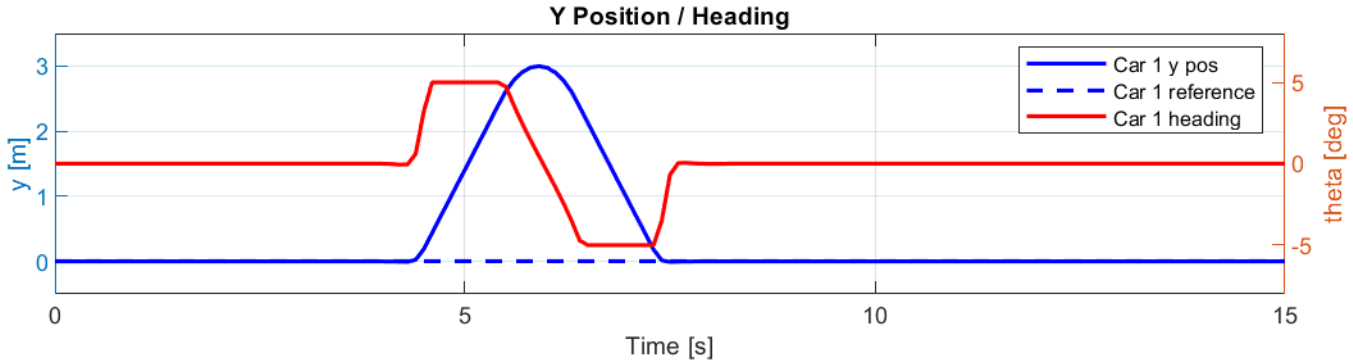

Figure 24: Y position tracking for the nonlinear controller with overtaking

This can be justified by the input plots. Positive steering is needed to change lanes as in Section 6.1, then a negative steering return the ego car to a straight position, then to the original lane. Another positive input is then applied to reset the car straight in its lane. As per Fig. 26, throttle is also increased to reach the reference speed,then is reduced to keep velocity constant. An acceleration is then needed to overtake the other car before a decrease in this input return the speed to the tracked value.

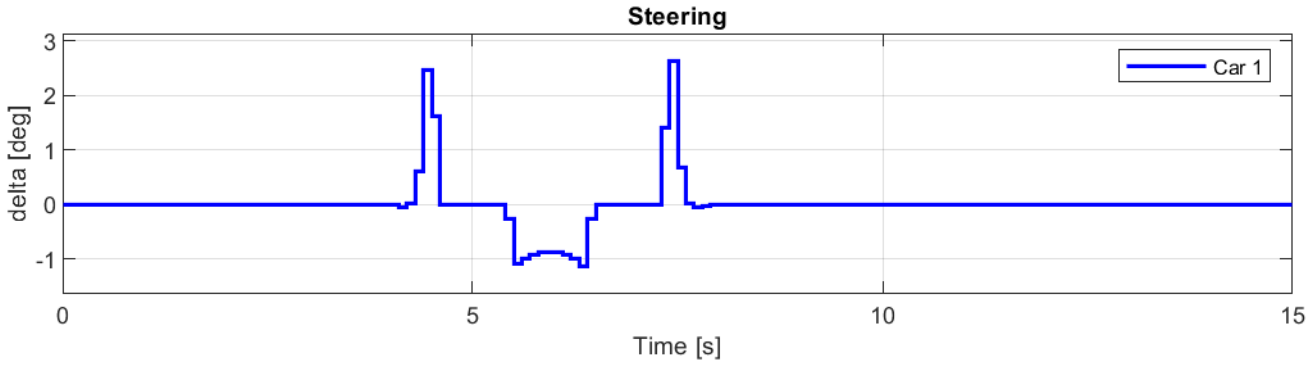

Figure 25: Steering input for the nonlinear controller with overtaking

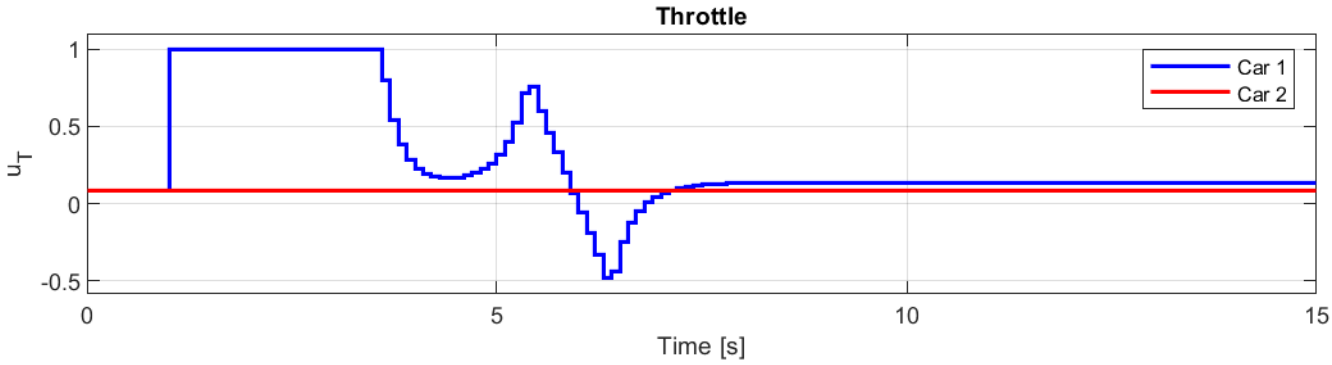

Figure 26: Throttle input for the nonlinear controller with overtaking

The elliptic constraints also ensure that the two cars do not collide. As it can be seen in Fig. 27, the minimum relative distance is of 3 meters.

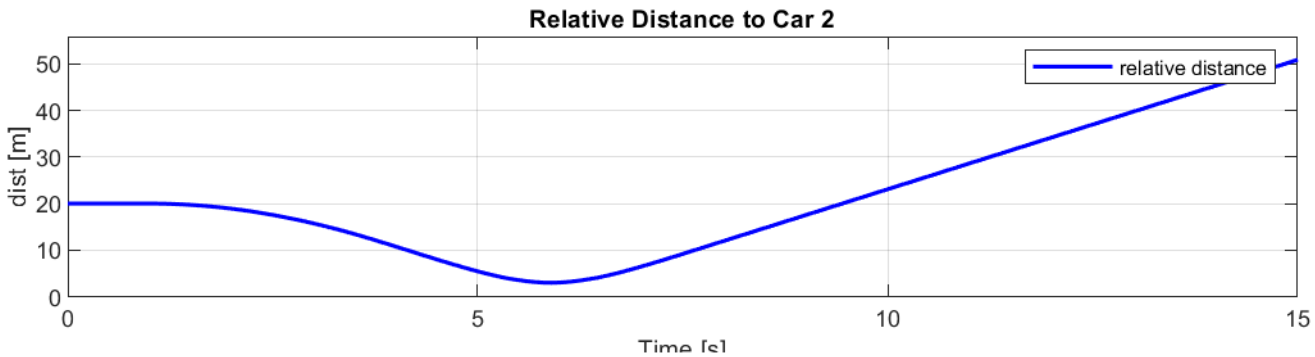

Figure 27: Relative distance between the 2 cars during overtaking