Mini-Project
-Model predictive control-
ME-425

Groupe AW :
Georg SCHWABEDAL 328434
Gautier DEMIERRE 340523
Benjamin BAHUREL 326888

December 2024

Figure 1: EPFL logo

1 Introduction

This project focuses on building an advanced Model Predictive Control (MPC) system for an autonomous car to handle real-world highway driving scenarios, ensuring a smooth and safe driving experience. After analyzing the system, we linearized its dynamics to simplify control, dividing it into longitudinal (speed control) and lateral (steering) subsystems. These controllers were then enhanced to ensure precise and offset-free tracking of speed and position, accounting for external factors like drag and disturbances. To address interactions with other vehicles on the highway, a robust tube-MPC controller was developed to maintain safe distances, even in the face of unpredictable behavior from a lead car. Finally, we scaled up to a nonlinear MPC controller capable of managing full-system dynamics, enabling the car to perform advanced maneuvers such as lane changes and overtaking.
The entire project was developed in MATLAB, which is included as part of the deliverables. Our reasoning and implementation are based on the project description, Model Predictive Control (ME-425) resources provided on the moodle page and TA's explanations.

2 Part 2 : Linearization

We are dealing with a nonlinear system, and we know that linearizing it simplifies the problem, enabling practical analysis and control design while maintaining acceptable accuracy within a limited operating range. Therefore, we establish the mathematical foundations to utilize a linearized version of our problem in the future, on points such as a steady state target velocity.

2.1 Deliverable 2.1 :

To begin we wish to analyse the system *Derive the analytical expressions of* $f(x_s, u_s)$*,* A*, and* B *as a function of* x_s *and* u_s*, where* $x_s = (0, 0, 0, V_s)$ *and* $u_s = (0, u_{T,s})$.

We know that :

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} V cos(\theta + \beta) \\ V sin(\theta + \beta) \\ \dfrac{V}{l_r} sin(\beta) \\ \dfrac{F_{motor} - F_{drag} - F_{roll}}{m} \end{bmatrix} = \begin{bmatrix} V cos(\theta + \beta) \\ V sin(\theta + \beta) \\ \dfrac{V}{l_r} sin(\beta) \\ \dfrac{\frac{u_T P_{max}}{V} - \frac{1}{2}\rho C_d A_f V^2 - C_r mg}{m} \end{bmatrix}$$

Moreover, we have : $f(\boldsymbol{x}, \boldsymbol{u}) \approx f(\boldsymbol{x_s}, \boldsymbol{u_s}) + A(\boldsymbol{x} - \boldsymbol{x_s}) + B(\boldsymbol{u} - \boldsymbol{u_s})$ where $A = \left.\dfrac{\partial f(x,u)}{\partial x}\right|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$, $B = \left.\dfrac{\partial f(x,u)}{\partial u}\right|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$, $\boldsymbol{x_s} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_s \end{bmatrix}$ and $\boldsymbol{u_s} = \begin{bmatrix} 0 \\ u_{T,s} \end{bmatrix}$. We start by derivating the analytical expression for $f(\boldsymbol{x_s}, \boldsymbol{u_s})$, asuming steady-state :

$$f(\boldsymbol{x_s}, \boldsymbol{u_s}) = \left.\begin{bmatrix} V cos(\theta + \beta) \\ V sin(\theta + \beta) \\ \dfrac{V}{l_r} sin(\beta) \\ \dfrac{F_{motor} - F_{drag} - F_{roll}}{m} \end{bmatrix}\right|_{(\boldsymbol{x_s}, \boldsymbol{u_s})} = \begin{bmatrix} V_s cos(\theta_s + \beta_s) \\ V_s sin(\theta_s + \beta_s) \\ \dfrac{V_s}{l_r} sin(\beta_s) \\ \dfrac{F_{motor,s} - F_{drag,s} - F_{roll,s}}{m} \end{bmatrix}$$

where :

$$\beta_s = atg(\frac{l_r tg(0)}{l_r + l_f}) = atg(0) = 0 \text{ and } \theta_s = 0$$

We have :

$$f(\boldsymbol{x_s}, \boldsymbol{u_s}) = \begin{bmatrix} V_s cos(\theta_s + \beta_s) \\ V_s sin(\theta_s + \beta_s) \\ \dfrac{V_s}{l_r} sin(\beta_s) \\ \dfrac{F_{motor,s} - F_{drag,s} - F_{roll,s}}{m} \end{bmatrix} = \begin{bmatrix} V_s cos(0 + 0) \\ V_s sin(0 + 0) \\ \dfrac{V_s}{l_r} sin(0) \\ \dfrac{F_{motor,s} - F_{drag,s} - F_{roll,s}}{m} \end{bmatrix} = \begin{bmatrix} V_s \\ 0 \\ 0 \\ \dfrac{\frac{u_{T,s} P_{max}}{V_s} - \frac{1}{2}\rho C_d A_f V_s^2 - C_r mg}{m} \end{bmatrix}$$

We now compute the matrix $A = \left.\dfrac{\partial f(x,u)}{\partial x}\right|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$, we have :

$$A = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial \theta} & \frac{\partial \dot{x}}{\partial V} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial \theta} & \frac{\partial \dot{y}}{\partial V} \\ \frac{\partial \dot{\theta}}{\partial x} & \frac{\partial \dot{\theta}}{\partial y} & \frac{\partial \dot{\theta}}{\partial \theta} & \frac{\partial \dot{\theta}}{\partial V} \\ \frac{\partial \dot{V}}{\partial x} & \frac{\partial \dot{V}}{\partial y} & \frac{\partial \dot{V}}{\partial \theta} & \frac{\partial \dot{V}}{\partial V} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})} = \begin{bmatrix} \frac{\partial V cos(\theta+\beta)}{\partial x} & \frac{\partial V cos(\theta+\beta)}{\partial y} & \frac{\partial V cos(\theta+\beta)}{\partial \theta} & \frac{\partial V cos(\theta+\beta)}{\partial V} \\ \frac{\partial V sin(\theta+\beta)}{\partial x} & \frac{\partial V sin(\theta+\beta)}{\partial y} & \frac{\partial V sin(\theta+\beta)}{\partial \theta} & \frac{\partial V sin(\theta+\beta)}{\partial V} \\ \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial x} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial y}) & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \theta} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial V} \\ \frac{\partial \dot{V}}{\partial x} & \frac{\partial \dot{V}}{\partial y} & \frac{\partial \dot{V}}{\partial \theta} & \frac{\partial \dot{V}}{\partial V} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$$

For each row we have :

Row 1 : $\frac{\partial V cos(\theta+\beta)}{\partial x} = 0$, $\frac{\partial V cos(\theta+\beta)}{\partial y} = 0$, $\frac{\partial V cos(\theta+\beta)}{\partial \theta} = -V sin(\theta+\beta)$ and $\frac{\partial V cos(\theta+\beta)}{\partial V} = cos(\theta+\beta)$

Row 2 : $\frac{\partial V sin(\theta+\beta)}{\partial x} = 0$, $\frac{\partial V sin(\theta+\beta)}{\partial y} = 0$, $\frac{\partial V sin(\theta+\beta)}{\partial \theta} = V cos(\theta+\beta)$ and $\frac{\partial V sin(\theta+\beta)}{\partial V} = sin(\theta+\beta)$

Row 3 : $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial x} = 0$, $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial y}) = 0$, $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \theta} = 0$ and $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial V} = \frac{sin(\beta)}{l_r}$

Row 4 : $\frac{\partial \dot{V}}{\partial x} = 0$, $\frac{\partial \dot{V}}{\partial y} = 0$, $\frac{\partial \dot{V}}{\partial \theta} = 0$ and $\frac{\partial \dot{V}}{\partial V} = \frac{\frac{-u_T P_{max}}{V^2} - \rho C_d A_f V}{m}$

Then we have :

$$A = \begin{bmatrix} \frac{\partial V cos(\theta+\beta)}{\partial x} & \frac{\partial V cos(\theta+\beta)}{\partial y} & \frac{\partial V cos(\theta+\beta)}{\partial \theta} & \frac{\partial V cos(\theta+\beta)}{\partial V} \\ \frac{\partial V sin(\theta+\beta)}{\partial x} & \frac{\partial V sin(\theta+\beta)}{\partial y} & \frac{\partial V sin(\theta+\beta)}{\partial \theta} & \frac{\partial V sin(\theta+\beta)}{\partial V} \\ \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial x} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial y}) & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \theta} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial V} \\ \frac{\partial \dot{V}}{\partial x} & \frac{\partial \dot{V}}{\partial y} & \frac{\partial \dot{V}}{\partial \theta} & \frac{\partial \dot{V}}{\partial V} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})} =$$

$$\begin{bmatrix} 0 & 0 & -V sin(\theta+\beta) & cos(\theta+\beta) \\ 0 & 0 & V cos(\theta+\beta) & sin(\theta+\beta) \\ 0 & 0 & 0 & \frac{sin(\beta)}{l_r} \\ 0 & 0 & 0 & \frac{\frac{-u_T P_{max}}{V^2} - \rho C_d A_f V}{m} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})} = \begin{bmatrix} 0 & 0 & -V_s sin(\theta_s+\beta_s) & cos(\theta_s+\beta_s) \\ 0 & 0 & V_s cos(\theta_s+\beta_s) & sin(\theta_s+\beta_s) \\ 0 & 0 & 0 & \frac{sin(\beta_s)}{l_r} \\ 0 & 0 & 0 & \frac{\frac{-u_T P_{max}}{V^2} - \rho C_d A_f V}{m} \end{bmatrix}$$

$$\mathrm{A} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\frac{-u_T P_{max}}{V^2} - \rho C_d A_f V}{m} \end{bmatrix}$$

We can finally compute the matrix $B = \frac{\partial f(x,u)}{\partial u}\Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$, we have :

$$B = \begin{bmatrix} \frac{\partial \dot{x}}{\partial \delta} & \frac{\partial \dot{x}}{\partial u_T} \\ \frac{\partial \dot{y}}{\partial \delta} & \frac{\partial \dot{y}}{\partial u_T} \\ \frac{\partial \dot{\theta}}{\partial \delta} & \frac{\partial \dot{\theta}}{\partial u_T} \\ \frac{\partial \dot{V}}{\partial \delta} & \frac{\partial \dot{V}}{\partial u_T} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})} = \begin{bmatrix} \frac{\partial V cos(\theta+\beta)}{\partial \delta} & \frac{\partial V cos(\theta+\beta)}{\partial u_T} \\ \frac{\partial V sin(\theta+\beta)}{\partial \delta} & \frac{\partial V sin(\theta+\beta)}{\partial u_T} \\ \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \delta} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial u_T} \\ \frac{\partial \dot{V}}{\partial \delta} & \frac{\partial \dot{V}}{\partial u_T} \end{bmatrix} \Big|_{(\boldsymbol{x_s}, \boldsymbol{u_s})}$$

For each row we have :

Row 1 : $\frac{\partial V cos(\theta+\beta)}{\partial \delta} = -V sin(\theta+\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2}$ and $\frac{\partial V cos(\theta+\beta)}{\partial u_T} = 0$

Row 2 : $\frac{\partial V sin(\theta+\beta)}{\partial \delta} = V cos(\theta+\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2}$ and $\frac{\partial V sin(\theta+\beta)}{\partial u_T} = 0$

Row 3 : $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \delta} = \frac{V}{l_r} cos(\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2}$ and $\frac{\partial \frac{V}{l_r} sin(\beta)}{\partial u_T} = 0$

Row 4 : $\frac{\partial \dot{V}}{\partial \delta} = 0$ and $\frac{\partial \dot{V}}{\partial u_T} = \frac{P_{max}}{mV_s}$

Then, we have :

$$B = \begin{bmatrix} \frac{\partial V cos(\theta+\beta)}{\partial \delta} & \frac{\partial V cos(\theta+\beta)}{\partial u_T} \\ \frac{\partial V sin(\theta+\beta)}{\partial \delta} & \frac{\partial V sin(\theta+\beta)}{\partial u_T} \\ \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial \delta} & \frac{\partial \frac{V}{l_r} sin(\beta)}{\partial u_T} \\ \frac{\partial \dot{V}}{\partial \delta} & \frac{\partial \dot{V}}{\partial u_T} \end{bmatrix}\Big|_{(\boldsymbol{x_s},\boldsymbol{u_s})} = \begin{bmatrix} -V sin(\theta+\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2} & 0 \\ V cos(\theta+\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2} & 0 \\ \frac{V}{l_r} cos(\beta)\frac{\frac{l_r}{l_r+l_f}\frac{1}{cos^2(\delta)}}{1+\beta^2} & 0 \\ 0 & \frac{P_{max}}{mV} \end{bmatrix}\Big|_{(\boldsymbol{x_s},\boldsymbol{u_s})} = \begin{bmatrix} 0 & 0 \\ \frac{V_s l_r}{l_r+l_f} & 0 \\ \frac{V_s}{l_r+l_f} & 0 \\ 0 & \frac{P_{max}}{mV_s} \end{bmatrix}$$

2.2 Deliverable 2.2 :

Explain from an intuitive physical / mechanical perspective, why this separation into independent subsystems is possible.

Since we linearized the system along the x-direction, we can consider that the interactions between the subsystems are negligible, but this is only valid if we have small variations around the state equilibrium. In our example, the coupling between the steering angle and the velocity is negligible considering that a small enough steering angle will not change the velocity too much. In the same way, a small change of velocity will not have too much effect on lateral position. Therefore, we can consider that the longitudinal system composed of the x position and the velocity V can be decoupled from the lateral system composed of the y position and the steering angle θ.

3 Part 3 : Design MPC Controllers for Each Sub-System

3.1 Deliverable 3.1 :

Since the full system could be separated in two sub-systems, we will discuss of the design procedure and the parameters tuning of each sub-system separately.

3.1.1 Longitudinal subsystem :

For this sub-system, the purpose was to design an MPC controller that would make the car's velocity go from 80 km/h to 120 km/h without steady-state error.

In order to do that, we used a classic MPC with delta form, with the classic cost matrices Q and R, and constraints on the input (which was the throttle here), no constraints on the state since there was no speed limits given by the document. We did not add a terminal constraint set since the only variable that was constrained was velocity, we concluded that a final cost would be enough for the longitudinal part, we took $Q_{final} = Q/2$ which is a good conservative weight.
For the tuning part, we wanted to be sure that we would go to the desired speed reference fast enough so we penalized a lot the deviation from the reference state ; which gave us Q = $\begin{bmatrix} 0 & 0 \\ 0 & 25 \end{bmatrix}$ as a good value for this, you can notice that we don't penalize the x position since it's not relevant in this deliverable.
Moreover, we did not penalized a lot the control effort since we wanted the controller to react fast once again, which gave us R = 0.2.
Those values are a bit odd, but since we were penalizing more the state than the input, we obtained a velocity behavior that looked like the one we could get from an offset free tracking controller. Indeed we could track a speed of 80 km/h almost perfectly even with the linearization point being at 120 km/h.
The tuning on the H value was not really relevant on this part, in fact, since the acceleration is the same, the prediction horizon does not have any big impact on the controller.

For the computation of u_{ref}, at steady state, the state doesn't change, so x(k+1) = x(k).
This is is giving us : $0 = A * (Vs_{ref} - x_s) + B * (us_{ref} - u_s) \Rightarrow us_{ref} = us - A * (Vs_{ref} - xs)/B$
We also had $xs_{ref} = ref$ which is pretty straightforward since we want to track speed reference.

3.1.2 Lateral subsystem :

In this part, we wanted to make the car shift from the right lane to the left lane in less than 3 seconds. In order to do that, we, once again, implemented an MPC controller in delta form, with Q and R costs.

This time, we added a terminal constraint set on top of the terminal cost in order to ensure recursive feasibility and stability. We implemented in the same way as seen in the course and the exercises by computing an LQR solution then computing iteratively the pre-sets and intersecting them until they were equal. The constraints and the final cost obtained this way are then used in the MPC to control the lateral motion of the car.

This implementation resulted in the following terminal invariant set :

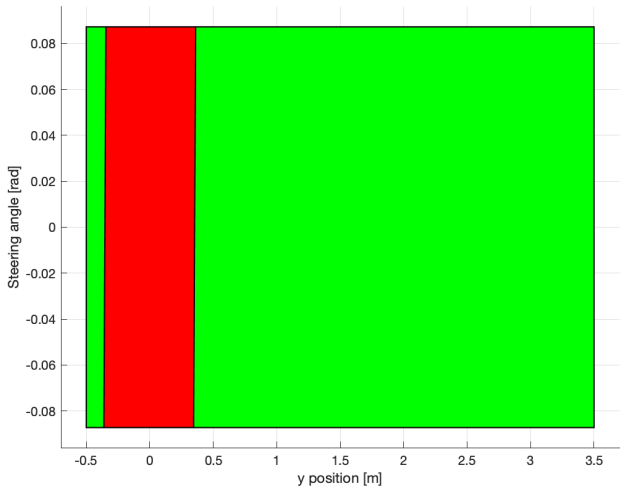


Figure 2: Plot of the terminal invariant set of the lateral sub-system

The green space being the state constraints and the red part being the terminal invariant set. We can see that the region is centered around 0, which is logic since we were working with an MPC in delta form. In the example above, $Q_{cont} = Q/2$ and $R_{cont} = R/2$, this gives us a rectangular set. Changing these weights change the size and the centering of the terminal set. By choosing $Q_{cont} = Q/80$ and $R_{cont} = 20 * R$, we had the following set :

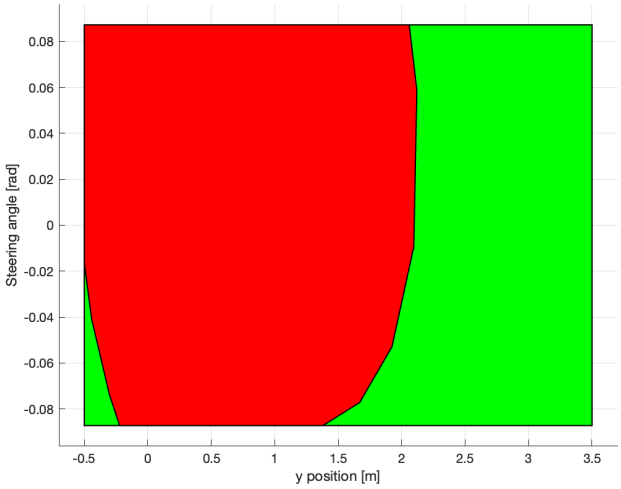


Figure 3: Plot of the terminal invariant set of the lateral sub-system

Now, we have a much larger terminal set which ensures a better stability and feasibility. The set still contains the origin, and we wanted to keep every point on the axis y = 0m which is why we tuned the parameters this way, so that when the car was at the right y pos, whatever its angle would be, the result would converge to the origin.

We also added new constraints on the state to be sure that the optimizer would not find solutions not feasible in the real world, and to make a smooth ride for the passengers.

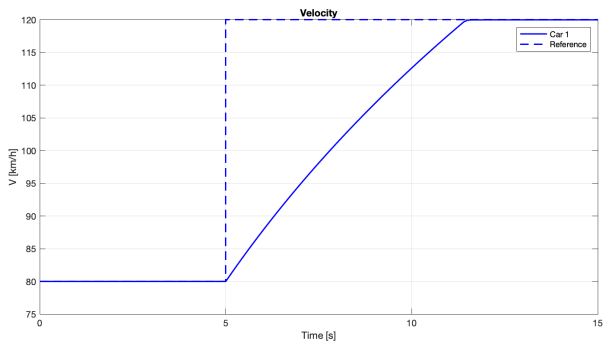
For the computation of x_{ref} and u_{ref}, $x_{ref} = ref$ the position we're trying to track and $us_{ref} = 0$ since we want the car to be horizontal when the tracking reference has been reached.

3.1.3 Tuning of Q,R,H

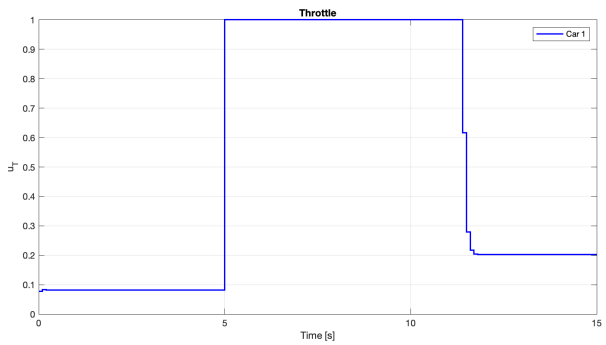
- H : we chose H = 2s since it was giving good results, choosing a bigger H would only result in longer computation time and no significant upgrades on the behavior. Choosing a shorter H was creating a poorer lateral behavior.
- Q, R : we chose Q = $\begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$ and R = 4 which were giving good results, the performances asked by the document were met.

3.1.4 Plots :

The plots can be obtained by running the file 'Deliverable3.m' and can be found in the folder 'plots' of every deliverable folder.

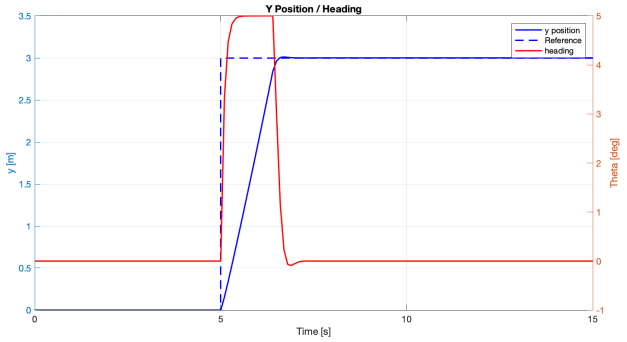


(a) Plot of the velocity along time

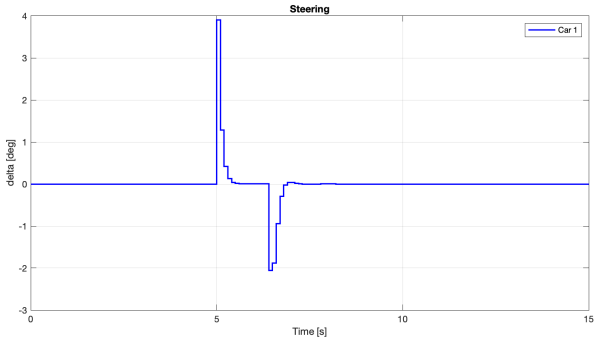


(b) Plot of the throttle input along time

We can see that the velocity requirements are met, the settling time is less than 10s.



(a) Plot of the position / heading along time



(b) Plot of the steering input along time

Once again, the requirements are met, we are switching lane in less than 3s, with a smooth movement.

4 Part 4 : Offset-Free Tracking

4.1 Deliverable 4.1 :

For this part, we focus only on changing the longitudinal controller and creating an estimator for the constant disturbance that is applied to the system. We saw in the last part that due to the linearization of the steady state, it was resulting in an offset when we were tracking velocities lower than the velocity of the linearized point.

Then, the purpose of this deliverable is to achieve an offset-free tracking controller. For the implementation of both the controller and the estimator, we used the course of 'Practical MPC' and the exercise series.

4.1.1 Design procedure :

- Estimator :

  For the estimator, we followed the design procedure of the course : we started by defining the augmented matrices (with the discretized ones given in the function), which gave us :

  $\widehat{A} = \begin{bmatrix} A_d(2,2) & B_d(2) \\ 0 & 1 \end{bmatrix}$ with $A_d(2,2)$ the evolution of the velocity, $B_d(2)$ the impact of the disturbance on the velocity and [0 1] representing a constant disturbance.

  $\widehat{B} = \begin{bmatrix} B_d(2) \\ 0 \end{bmatrix}$ with $B_d(2)$ the impact of the input on the velocity and 0 meaning we can not directly measure the disturbance.

  $\widehat{C} = \begin{bmatrix} C_d(2,2) & 0 \end{bmatrix}$ with $C_d(2,2)$ the velocity measurement and 0 meaning we can not directly measure the disturbance.

  Having these matrices, we can compute the estimator L by using the poles placement method. We used the values 0.6 and 0.7 for the poles since it was said in the exercises that those poles were giving pretty fast answers.

  Then, thanks to the estimator, we can predict what the next state of our system will be accounting the disturbance. We did this in the delta form since our controller is in this form too, which is given by the following formula :

  $\Delta\widehat{z_{next}} = \widehat{A} * \Delta z + \widehat{B} * \Delta u + L * (\Delta y - \widehat{C} * \Delta z)$

  Then, we transform it back into normal coordinates : $\widehat{z_{next}} = \Delta\widehat{z_{next}} + \widehat{x_s}$

- Controller :

  The controller was designed exactly like the one in the deliverable 3.1, we used the same method for computing us_ref. The only changes we made are on the dynamics of the system because, now, we need to take into account the disturbance.

  In order to do this, we created a vector d = [0 B(2,1)*d_est] (dimensions to match with the states that has two variables).

  Then, the dynamics change from $X(:,k+1) == mpc.A * X(:,k) + mpc.B * U(:,k)$ to $X(:,k+1) == mpc.A * X(:,k) + mpc.B * U(:,k) + d$

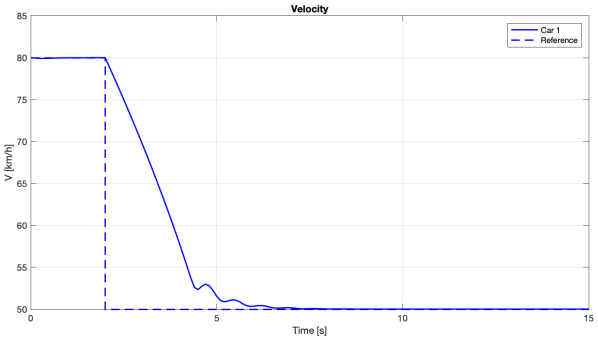
- Parameters tuning : We choose a prediction horizon of 3 since it was giving us good performance, above this value, we could not notice any relevant changes. When the prediction horizon was too short, we had a completely oscillating behavior which is not a wanted behavior.

  The tuning of Q, R and Q_{final} is the result of iterations to get the appropriate behavior while avoiding too much oscillations of the system which could result in a bad experience for the passengers of the car.

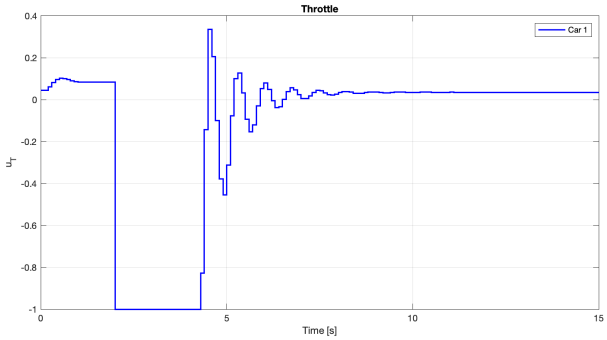
For the computation of u_s, at steady state, the state doesn't change, so x(k+1) = x(k).
This is is giving us : $0 = A * (Vs_{ref} - x_s) + B * (us_{ref} - u_s)$ with $us_{ref} = u_s - A * (Vs_{ref} - x_s)/B + B * d_{est} = u_s - A * (Vs_{ref} - x_s)/B - d_{est}$
We also had $xs_{ref} = ref$ which is pretty straightforward since we want to track speed reference.

4.1.2 Plots :

The plots can be obtained by running the file 'Deliverable4.m' and can be found in the folder 'plots' of every deliverable folder.

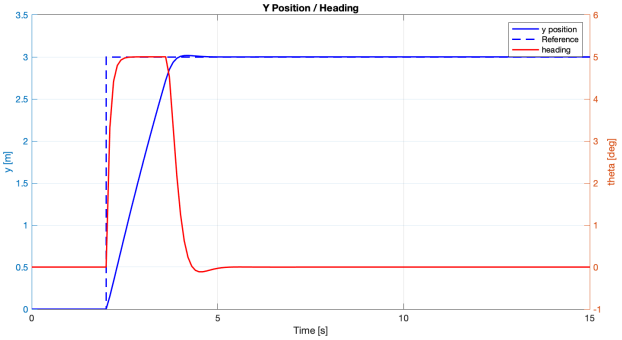


(a) Plot of the velocity along time

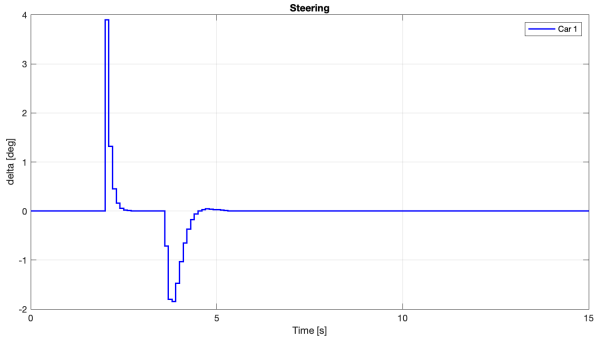


(b) Plot of the throttle input along time

We can see that the velocity requirements are met, we have an offset free-tracking.



(a) Plot of the position / heading along time



(b) Plot of the steering input along time

We keep the lateral behavior that we had obtained in the deliverable 3.1.

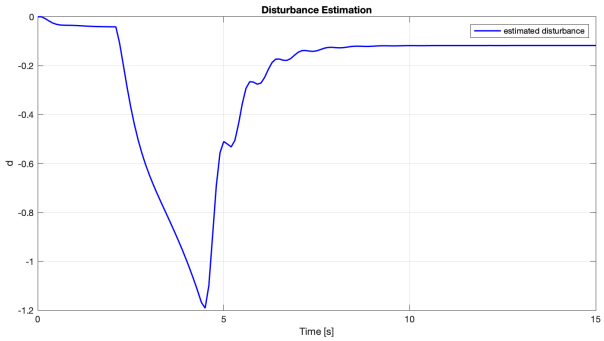


Figure 8: Plot of the estimated disturbance along time

We see that the controller compensate for the estimated disturbance with the previous plots.

5 Part 5 : Robust Tube MPC for Adaptive Cruise Control

5.1 Deliverable 5.1 :

Our objective is to control the relative position and speed of the ego car while tracking the lead car as a reference, accounting for uncertainties in the lead car's throttle behavior. Since this tube-MPC is designed to ensure a safe distance between the two cars, our focus will be exclusively on tuning the longitudinal dynamics. We track our performance and optimize over the relative longitudinal state between the cars:

$$\Delta = \tilde{x_{\text{long}}} - x_{\text{long}} - x_{\text{safe}}$$

Our implementation follows the recommended process, drawing primarily on the concepts covered in weeks 4, 6, 8, and 9, specifically focusing on constrained systems,practical and robust MPC.

5.1.1 Minimum Robust Invariant Set

We used an LQR controller to compute K, using the matrices A and B derived from the discretization of the longitudinal system. Special attention was given to inverting the sign of B, as outlined in equation (8) of the reference document. The cost matrices Q and R were selected by iterative tests to achieve optimal performance, respectively `diag ([20, 54.5])` and `1.5`.

We followed the algorithm presented on slide 24 of Robust MPC II. The polyhedron Ω_0, represents the initial uncertainty of ± 0.5 around the steady-state $u_{T,s}$. Iterating, the next set W_{i+1} is computed by applying the system dynamics to the previous set and adding the disturbance (Minkowski sum). We stop once consecutive sets W_{i+1} and W_i are sufficiently similar referring to the proposed tolerance or until a programming safety criterion is reached.

We tuned the cost functions Q and R and used the `.minHRep()` function to robustly minimize the polyhedron ϵ. This helped reduce memory usage but was computationally heavier, making each iteration more expensive but globally more efficient. Robustness can be seen through it's smooth and compact shapes, symmetry and containment of the origin.

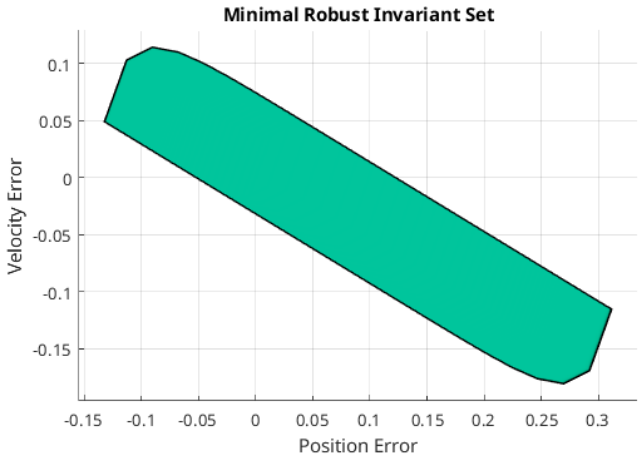


Figure 9: Minimal Robust Invariant set of disturbance.

5.1.2 Tightened Constraints

We utilize the Pontryagin Difference to compute the new tightened constraints. These will ensure feasibility at all time under a certain control law even for worst-case scenarios. We have chosen `x_safe_pos = 8 meters` as it is a 33 % margin of the `distance_min = 6 meters`. Future testing has proven this value to be acceptable.
We established the state constraints within the polytope X that limits the state :

$$\Delta \leq distance\_min - x\_safe\_pos \Rightarrow \Delta \leq -2$$

and the input constraints within the polytope U between the maximum possible braking and acceleration:

$$-1 \leq u \leq 1$$

New state and input constraints, tightened to take into account the disturbance are:

$$\tilde{X} = X - \epsilon$$

$$\tilde{U} = U - K * \epsilon$$

5.1.3 Terminal Components

Terminal controller, constraints and weight respectively stability, feasibility and ensure optimal long-term performance without the need to compute an excessively large (or infinite) horizon.
The terminal controller K_f was computed through LQR. The matrices A and B as in 6.1.1. The terminal weights Q_f and R_f are respectively the previous $Q/2$ and $R * 2$. These choices reflect the expectation that by the end of the prediction horizon, the system will have approached a steady-state condition, justifying a higher penalty on control inputs to prioritize stability and efficiency in the final stage.
The terminal constraints is the set such as if the system starts in this set, it remains in the set under the given dynamics and constraints indefinitely. We use the method to compute a maximal invariant set given in the Introduction to Constrained Systems course slide 29. First establishing the polytope χ_f which represents the terminal invariant set using the tightened constraints: $\tilde{X}$ and $\tilde{U}$. Proceding we compute the intersection region between χ_f and it's preset $\chi_{f,pre}$ until there is no variation between iterations no more or until a programming safety criterion is reached:

$$\chi_f \leftarrow \chi_f \cap \chi_{f,pre}$$

The resulting has been shown below. We have confidence in our model as it encompasses a reasonable region in a smooth and symmetrical way.

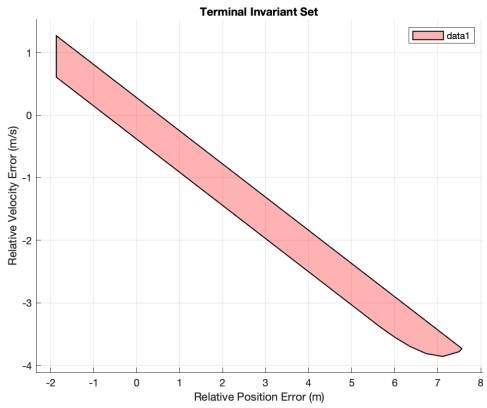


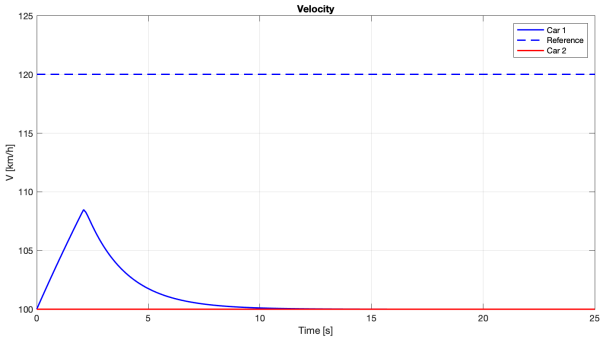
Figure 10: Terminal constraints.

5.1.4 Controller Design

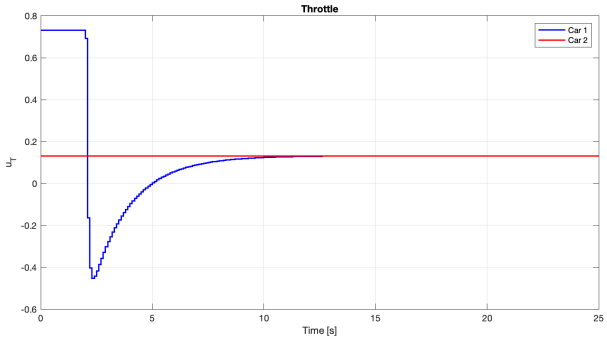
We optimize an objective function *obj* and a *cost* list to generate the MPC through an optimizer.
Obj is the function we wish to minimize representing the cost to follow a trajectory. We use a quadratic cost re-utilizing Q and R as state and input penalty and a terminal cost leveraging the previously computed Q_f.
Cost is responsible for the listing of all the possible constraints applying on our system. We successively set up the deviation from the safe reference point (8 meters), the system dynamics in relative coordinates, the tightened state and input constraints and the terminal set constraints.

5.1.5 Plots :

Case 1: The following plots were obtained by applying the initial conditions and references: `myCar.x0 = [0 0 0 100/3.6]'` with `myCar.ref = [0 120/3.6]'` and the other car starting from `otherCar.x0 = [15 0 0 100/3.6]'` with `otherCar.u = car.u_const(100/3.6)`. The plots can be obtained by running the file 'Deliverable5 1.m' and can be found in the folder 'plots' of every deliverable folder.



(a) Velocity of each car



(b) Throttle input along time

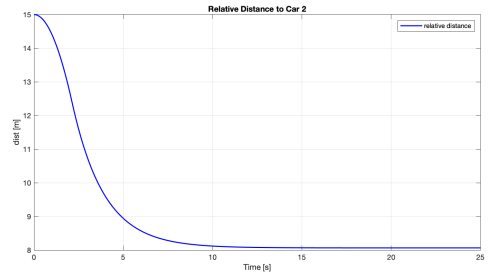
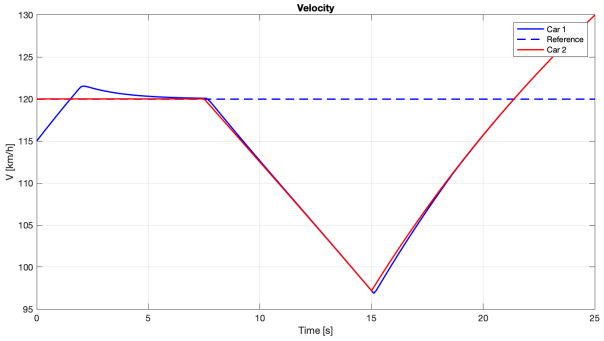
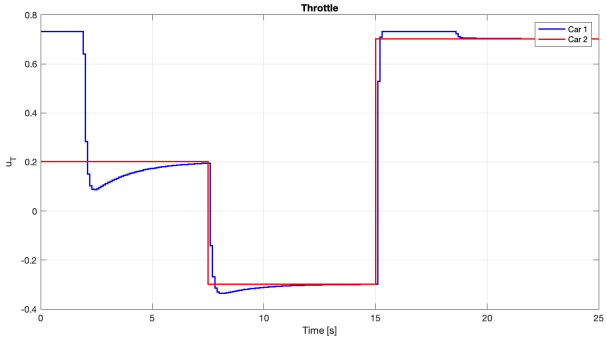


Figure 12: Relative distance between cars

Case 2: The following plots were obtained by applying the initial conditions and references:
`myCar.x0 = [0 0 0 115/3.6]'` with `myCar.ref = [0 120/3.6]'` and the other car starting from `otherCar.x0 = [8 0 0 120/3.6]'` with `otherCar.u = car.u_fwd_ref()` and `otherCar.ref = car.ref_robust()`.



(a) Velocity of each car



(b) Throttle input along time

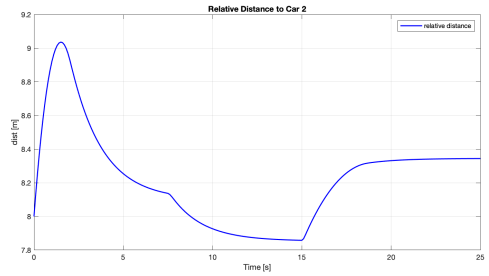


Figure 14: Relative distance between cars

As we can see from the plots, the ego car behaves as expected to the disturbances instantiated by the lead car as it brakes when the lead car slows down and it accelerates when the lead car is getting ahead.

6 Part 6 : Nonlinear MPC

6.1 Deliverable 6.1 :

6.1.1 Design Procedure :

In order to implement the NMPC, we imagined our design procedure from what was proposed in the Exercise 7.

We first started by attributing a variable to every state and input components in order to manipulate them in an easier way.

Then, we wanted to force the first state of the car in the optimizer to be the x_0 state. After that, we implemented the cost function : we used an error-type implementation where we want to minimize the distance between the state of the car (y position and velocity) and the reference that the NMPC needs to track (3 meters in y position and 100 km/h in velocity).

We also added a term on the steering input, which can be minimized if we increase the weight on this term to obtain faster or slower settling times. Indeed, minimizing the steering will result in a longer settling time, but probably a more pleasant ride experience for the passengers.

This results in the following cost function :

$$cost = 10 * (error\_speed * error\_speed') + 10 * (error\_y * error\_y') + 10 * (steering * steering')$$

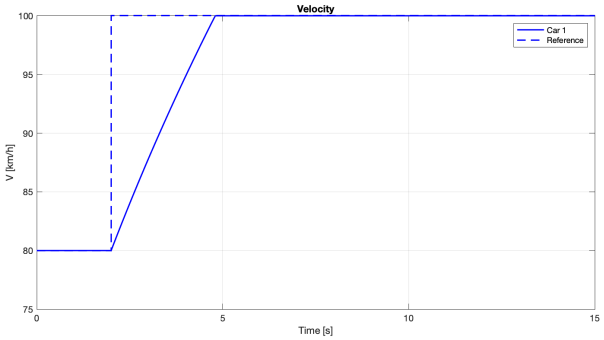
Then, we added the dynamical constraints on the state and inputs of the system (just like in exercise 7), implemented thanks to RK4. To finish, we add the inputs and states constraints that were mentioned in the previous parts, we ensure that $u_0 == U(1,:)$ and the minimization.

6.1.2 Parameters Tuning :

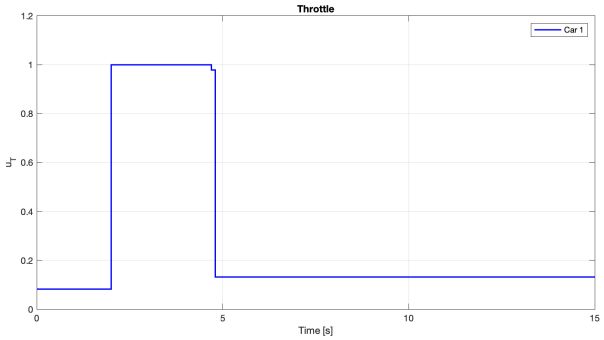
- Prediction Horizon : even with extremely short prediction horizon, the system's behavior was the same as the behavior with longer prediction horizon. Only the computation time was changing. This can be explained by the fact that nothing comes to disturb the tracking, so whatever the horizon is, the sequence of inputs will, most likely, remain the same.
- Weights w_i in the cost function : we chose an equal weight of 10 for every term of the cost function since we did not want to target specifically any of the terms.

6.1.3 Plots :

The plots can be obtained by running the file 'Deliverable6_1.m' and can be found in the folder 'plots' of every deliverable folder.

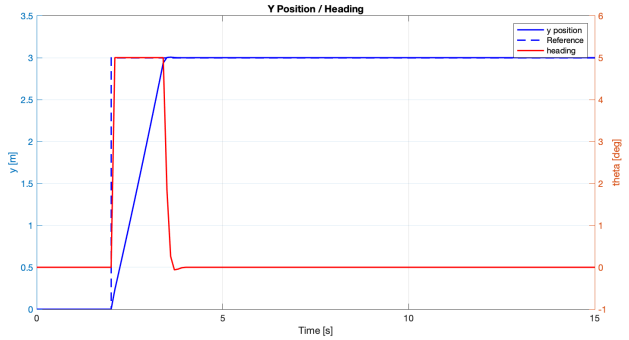


(a) Plot of the velocity along time

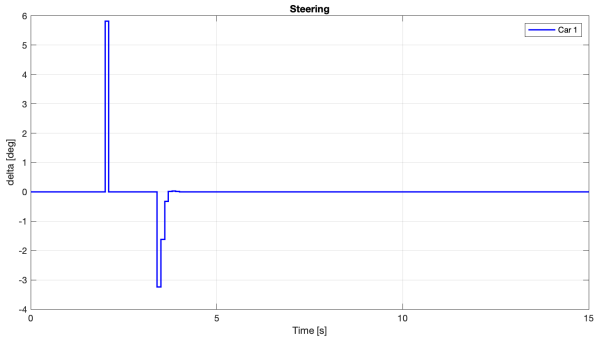


(b) Plot of the throttle input along time

We can see that the velocity requirements are met : we track the reference at 100 km/h with no steady-state error and with all the constraints not broken. Because we are using RK4 and, thus, we use the exact model of the system then there is no error possible. In the previous deliverables, we were using a linearization point different than our actual speed which was causing the offset.



(a) Plot of the position / heading along time



(b) Plot of the steering input along time

The plot shows that the 'switching lane' function is also done correctly with a smooth trajectory.

6.2 Deliverable 6.2 :

6.2.1 Design Procedure :

The NMPC design is almost the same as the one presented in the deliverable 6.1. Indeed, we used the same code, but we added a few lines to implement the "overtake" function. The only noticeable change from the previous code is about the cost function. We changed the weight on the y position error in order to give it a priority over the rest. This was done since the weights of 6.1 were giving an oscillating behavior, which seems logic since there was no change on y position for this deliverable.

The only thing we needed to add was the "overtake" function to make everything works.
In order to do this, we defined the vectors p and p_L of, respectively, the longitudinal and lateral coordinates of the ego car and the other car.

For the p vector, we used the $X(1,:)$ and $X(2,:)$ vectors of the ego car which represent respectively the x and y position along time. For the other car, we set $y_L = 0$ for all times of the simulation. For the longitudinal coordinates, we used the following equation : $x_{other}(k+1) = x_{other}(k) + k * T_s * v_{other}$ in order to predict the future position of the other car.

Once this was done, we implemented an ellipsoidal constraint in the same loop as $f_{discrete}$ to constrain every state. We used the following ellipsoidal constraint : $(p - p_L)^T * H * (p - p_L) \geq 1$.

6.2.2 Design of the matrix H :

The implementation of the ellipsoidal constraint was implying the tuning of a new matrix H. As recommended, we used a diagonal matrix to compute the constraint.

When we develop $(p-p_L)^T * H * (p-p_L) \geq 1$ and we take the diagonal values of H as α and β, we get the following equation :

$$\alpha(x - x_0)^2 + \beta(y - y_0)^2 \geq 1$$

with x,y the coordinates of the ego car and x_0,y_0 the coordinates of the other car.

We took two extreme cases where we have : one with $x - x_0 = 0$ and the other with $y - y_0 = 0$. It gives the following system :

$$\begin{cases} \alpha(x - x_0)^2 \geq 1 \\ \beta(y - y_0)2 \geq 1 \end{cases}$$

From there, we chose that a good value for the lateral security distance $(y - y_0)$ would be 3 meters, which is the same as switching lanes (what we would do in real life). For the longitudianl security distance, by trying different values, we concluded that 8 meters was giving good results.

Solving the system, we have =

$$\begin{cases} \alpha \cdot 8^2 \geq 1 \Rightarrow \alpha = \dfrac{1}{64} \\ \\ \beta \cdot 3^2 \geq 1 \Rightarrow \beta = \dfrac{1}{9} \end{cases}$$

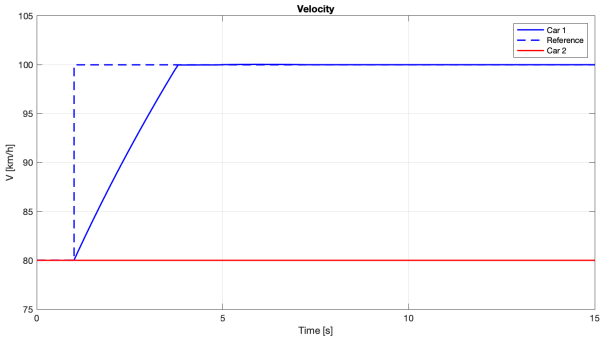
This gives us a final matrix : $H = \begin{bmatrix} \frac{1}{64} & 0 \\ 0 & \frac{1}{9} \end{bmatrix}$

6.2.3 Prediction horizon :

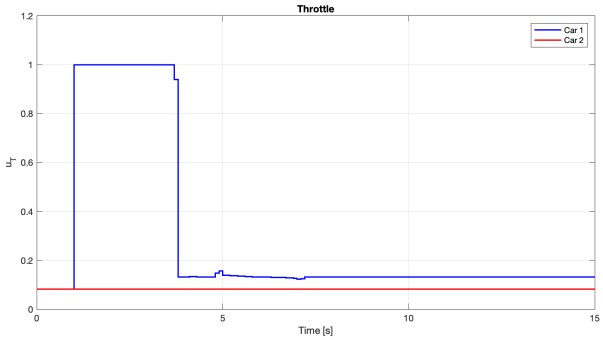
We chose a value of $H = 2s$ which was giving us good results. Too high values of the prediction horizon were leading to long computations, and sometimes the car would not overtake the other car and would stay behind it. This can be explained by the fact that the car wasn't fast enough in the prediction horizon to consider an overtake; hence the cost was lower if the car stayed behind the other car. If the horizon was too short, we could observe oscillations and poor reaction time, which is not a behavior that is wanted.
The rest of the implementation is the same as the code provided in the deliverable 6.1.

6.2.4 Plots :

The plots can be obtained by running the file 'Deliverable6_2.m' and can be found in the folder 'plots' of every deliverable folder.

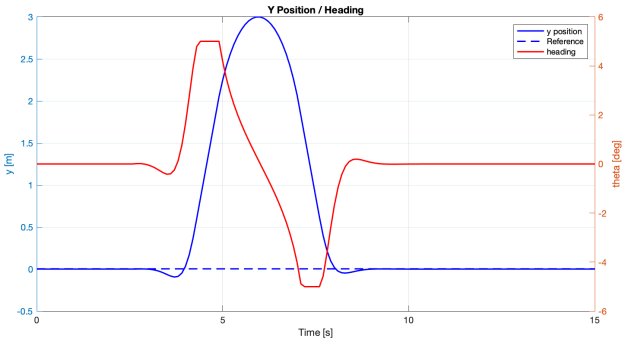


(a) Plot of the velocity along time

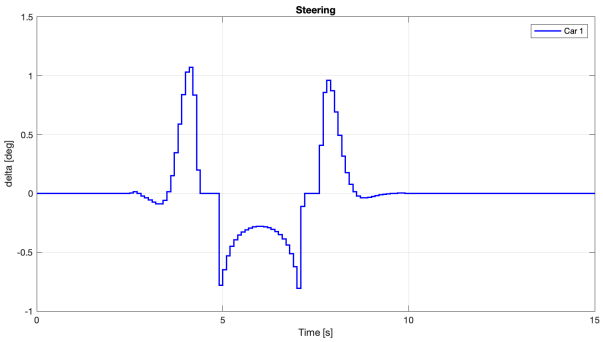


(b) Plot of the throttle input along time

We can see that the velocity requirements are met; moreover, the throttle is used as a real person would do: a first acceleration to attain 100 km/h, then we keep a constant throttle input to stay at this speed.



(a) Plot of the position / heading along time



(b) Plot of the steering input along time

The 'overtake' is implemented correctly, the car is dodged with a smooth trajectory, with decent steering and heading angles.

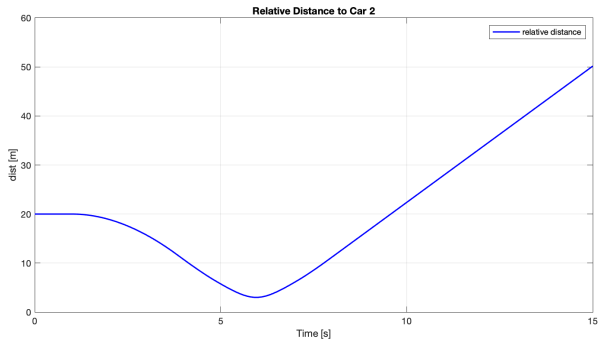


Figure 19: Plot of the relative distance between the two cars along time

Finally, with this last plot, we can show that the overtake takes place during the seconds 3 to 8, which is visible on the previous plots of heading and steering. Moreover, the relative distance doesn't overcome 3 meters (the safety distance in y position).

7 Conclusion :

In the end, our controllers effectively address some of the most common real-world challenges encountered while driving on a highway. We developed solutions for linearized dynamics, offset-free tracking, robust tube MPC, and nonlinear MPC, tackling tasks such as cruise control in dynamic environments, lane changes, and overtaking.
Our major challenges included tuning parameters for robust performance under disturbances and ensuring computational efficiency in tube- and nonlinear-MPC. We overcame these difficulties by deepening our understanding of the course material, analyzing reference solutions from the exercises and seeking support from the course team through ED or in class.
We thank Professor Colin Jones for his guidance and the foundational resources that supported our work and to the ME-425 teaching assistants and support team for their invaluable help throughout this project.