École Polytechnique Fédérale de Lausanne

Model Predictive Control

Mini-Project

Authors:

Estelle BAUMANN 347381

Johanne PINEL 324707

Julien THÉVENOZ 326772

Professor:

Colin Jones

Lausanne

2024 Autumn Semester

Contents

1 System Dynamics

2 Linearization

2.1 Deliverable

- Derive the analytical expressions of $f(x_s, u_s)$, A, and B as a function of x_s and us, where $\mathbf{x_s}$ = (0, 0, 0, Vs) and $\mathbf{u_s}$ = (0, $u_{T,s}$)

To do this, we will proceed step by step. First, let us define the state equation:

$$f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} Vcos(\theta + \beta) \\ Vsin(\theta + \beta) \\ \frac{V}{l_r} sin(\beta) \\ \frac{F_{motor} - F_{drag} - F_{roll}}{m} \end{bmatrix} = \begin{bmatrix} f1 \\ f2 \\ f3 \\ f4 \end{bmatrix} \tag{1}$$

With :

$$F_{\text{motor}} = \frac{u_T P_{\max}}{V}$$
$$F_{\text{drag}} = \frac{1}{2} \rho C_d A_f V^2$$
$$F_{\text{roll}} = C_r mg$$

Next, we can calculate the matrices A and B such that $A = \frac{\partial f(x,u)}{\partial x} \mid_{(x_s,u_s)}$ and $B = \frac{\partial f(x,u)}{\partial u} \mid_{(x_s,u_s)}$

$$A = \begin{bmatrix} \frac{\partial f1}{\partial x} & \frac{\partial f1}{\partial y} & \frac{\partial f1}{\partial \theta} & \frac{\partial f1}{\partial V} \\ \frac{\partial f2}{\partial x} & \frac{\partial f2}{\partial y} & \frac{\partial f2}{\partial \theta} & \frac{\partial f2}{\partial V} \\ \frac{\partial f3}{\partial x} & \frac{\partial f3}{\partial y} & \frac{\partial f3}{\partial \theta} & \frac{\partial f3}{\partial V} \\ \frac{\partial f4}{\partial x} & \frac{\partial f4}{\partial y} & \frac{\partial f4}{\partial \theta} & \frac{\partial f4}{\partial V} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -Vsin(\theta + \beta) & cos(\theta + \beta) \\ 0 & 0 & Vcos(\theta + \beta) & sin(\theta + \beta) \\ 0 & 0 & 0 & \frac{sin(\beta)}{l_r} \\ 0 & 0 & 0 & \frac{1}{m}(\frac{-u_T P_{Max}}{V^2} - \rho C_d A_f V) \end{bmatrix} \tag{2}$$

$$B = \begin{bmatrix} \frac{\partial f1}{\partial \delta} & \frac{\partial f1}{\partial u_T} \\ \frac{\partial f2}{\partial \delta} & \frac{\partial f2}{\partial u_T} \\ \frac{\partial f3}{\partial \delta} & \frac{\partial f3}{\partial u_T} \\ \frac{\partial f4}{\partial \delta} & \frac{\partial f4}{\partial u_T} \end{bmatrix} = \begin{bmatrix} V(-sin(\theta + \beta)) \frac{\partial \beta}{\partial \delta} & 0 \\ V(cos(\theta + \beta)) \frac{\partial \beta}{\partial \delta} & 0 \\ \frac{V}{l_r} cos(\theta + \beta) \frac{\partial \beta}{\partial \delta} & 0 \\ 0 & \frac{P_{Max}}{mV} \end{bmatrix} \tag{3}$$

Using $(arctan(x))' = \frac{1}{1+x^2}$ and $(tan(x))' = 1 + tan^2(x)$ we can find

$$\frac{\partial \beta}{\partial \delta} = \frac{\partial}{\partial \delta} arctan(\frac{l_r tan(\delta)}{l_r + l_f}) = \frac{1}{1 + (\frac{l_r tan(\delta)}{l_r + l_f})^2} \frac{\partial}{\partial \delta} \frac{l_r tan(\delta)}{l_r + l_f} = \frac{1}{1 + (\frac{l_r tan(\delta)}{l_r + l_f})^2} \frac{l_r (1 + tan^2(\delta))}{(l_r + l_f)}$$

Applying the values of $\mathbf{x_s}$ and $\mathbf{u_s}$, where $\mathbf{x_s}$ = (0, 0, 0, Vs) and $\mathbf{u_s}$ = (0, $u_{T,s}$), we get $\frac{\partial \beta}{\partial \delta} \mid_{(x_s,u_s)} = \frac{l_r}{(l_r + l_f)}$. So A and B are equal to :

$$A\,|_{(x_s,u_s)}= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & V_s & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{m}(\frac{-u_T P_{Max}}{V_s^2} - \rho C_d A_f V_s) \end{bmatrix} \tag{4}$$

$$B\,|_{(x_s,u_s)}= \begin{bmatrix} 0 & 0 \\ V_s \frac{\partial \beta}{\partial \delta}\,|_{(x_s,u_s)} & 0 \\ \frac{V_s}{l_r} \frac{\partial \beta}{\partial \delta}\,|_{(x_s,u_s)} & 0 \\ 0 & \frac{P_{Max}}{mV_s} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ V_s \frac{l_r}{l_r+l_f} & 0 \\ \frac{V_s}{l_f+l_r} & 0 \\ 0 & \frac{P_{Max}}{mV_s} \end{bmatrix} \tag{5}$$

2.2 Deliverable

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

Our state consists of four variables: x, y, θ, and V. However, when analyzing our system, we can divide it into two parts: one for the longitudinal axis and one for the lateral axis. Indeed, the observable variations along the longitudinal axis are those of velocity and acceleration. Along the lateral axis, we can observe the variation of the angle θ when we apply a steering angle δ to the car, as well as a change in the position y.

We can calculate the x-velocity of the car as $V_x = ||V|| * cos(theta) \approx ||V||$ since cosine can be approximated as 1 for small angles (which is valid since we are told $|\theta| \leq 5°$). This means car-road angle θ has no impact on the position or velocity in x. This separation of the system allows us to simplify the problem by breaking it down into two independent subsystems which can be considered independently, with only the throttle u_T affecting the longitudinal variables, and only the steering angle δ affecting the latitudinal variables (θ and y).

Here are the two sub-systems of our problem :

- Sys_lon: speed on the x axis (longitude) is handle by V and u_T

$$A_{lon} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{1}{m}(\frac{-u_T P_{Max}}{V^2} - \rho C_d A_f V) \end{bmatrix} \quad B_{lon} = \begin{bmatrix} 0 \\ \frac{P_{Max}}{mV} \end{bmatrix} C_{lon} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad D_{lon} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6}$$

- Sys lat: speed on the y axis (latitude) is handle by θ and δ

$$A_{lat} = \begin{bmatrix} 0 & V \\ 0 & 0 \end{bmatrix} \quad B_{lat} = \begin{bmatrix} V_s \frac{l_r}{l_r+l_f} \\ \frac{V_s}{l_f+l_r} \end{bmatrix} \quad C_{lon} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad D_{lon} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7}$$

3 Design MPC Controllers for Each Sub-System

3.1 Deliverable

- Explanation of design procedure that ensures recursive constraint satisfaction.

  In this approach, the infinite horizon optimal control problem is divided into two subproblems. The first subproblem involves solving a finite horizon MPC problem up to $k = N$, where the constraints are active. The objective is to minimize a quadratic cost over the period $0 \leq i \leq N-1$, while respecting the dynamic constraints $x_{i+1} = Ax_i + Bu_i$ and the input and state constraints $Cx_i + Du_i \leq b$, formulated as:

$$V^*(x,u) = \min_{x,u} \sum_{i=0}^{N-1} l(x_i, u_i) = \min_{x,u} \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) \tag{8}$$

  In the second subproblem, for $k \geq N$, optimal control is applied with an infinite horizon quadratic cost calculated by an unconstrained LQR. The cost to minimize is:

$$V_f^*(x_N) = \min_{x,u} \sum_{i=N}^{\infty} \left(x_i^T Q x_i + u_i^T R u_i\right) = x_N^T P x_N \tag{9}$$

  using the LQR control law starting from x_N where the input and state constraints are no longer necessary, as the system is assumed to be in a maximal invariant set. The LQR control law will maintain the system in this invariant set forever, which is why the cost is calculated over an infinite horizon and no constraints need to be applied.

  Therefore, the optimal cost function is:

$$J^*(x) = V^*(x,u) + V_f^*(x_N) = \min_{x,u} \sum_{i=0}^{N-1} \left(x_i^T Q x_i + u_i^T R u_i\right) + x_N^T P x_N. \tag{10}$$

  For the implementation of the MPC controllers, the system dynamics are defined by

$$x_{k+1} = A \cdot x_k + B \cdot u_k \tag{11}$$

  We define input and state constraints as follows :

$$MU \leq m \quad \& \quad FX \leq f \tag{12}$$

  For the longitudinal system we constrain only the input u_T :

$$-1 \leq u_T \leq 1 \quad \Rightarrow \quad M = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad U = \begin{bmatrix} u_T \end{bmatrix} \quad m = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{13}$$

  For the latitudinal system we constrain the input δ as well as the states y and θ:

$$\begin{matrix} |\theta| \leq 0.0873[rad] \\ -0.45m \leq y \leq 3.5m \end{matrix} \quad \Rightarrow \quad F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \quad X = \begin{bmatrix} y \\ \theta \end{bmatrix} \quad f = \begin{bmatrix} 3.5 \\ 0.0873 \\ 0.45 \\ 0.0873 \end{bmatrix} \tag{14}$$

$$|\delta| \leq 0.45236[rad] \quad \Rightarrow \quad M = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad U = \begin{bmatrix} \delta \end{bmatrix} \quad m = \begin{bmatrix} 0.45236 \\ 0.45236 \end{bmatrix} \tag{15}$$

In order to implement a recursive constraint, we employ an iterative loop that ensures the system dynamics and constraints are satisfied at every step over the entire prediction horizon. The procedure is executed as follows:

for $i = 1 : N - 1$

$\text{con} = [\text{con},\ x(:, i+1) = Ax(:, i) + Bu(:, i)];$

$\text{con} = [\text{con},\ Mu(:, i) \leq m];$

$\text{con} = [\text{con},\ Fx(:, i) \leq f];$

$\text{obj} = \text{obj} + (x(:, i) - x\_\text{ref})^T Q(x(:, i) - x\_\text{ref}) + (u(:, i) - u\_\text{ref})^T R(u(:, i) - u\_\text{ref});$

end

At each step, we impose the dynamics constraint $x_{k+1} = A \cdot x_k + B \cdot u_k$, along with the input constraint $M * u_k \leq m$ and the state constraint $F * x_k \leq f$. These constraints are applied iteratively across the prediction horizon. The control optimization aims to compute some control inputs that adhere to the system dynamics, minimize tracking errors, and satisfy the input constraints.

NB : For the longitudinal subsystem, there is no $F * x_k \leq f$ in the for loop since we there are no constraints on the state [x V].

- Explanation of choice of tuning parameters. (e.g., Q, R, H).

  The tuning parameters Q and R define the trade-off between tracking performance and control effort in the MPC controller. For the car's MPC, we initialize:

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad R = \begin{bmatrix} 1 \end{bmatrix} \tag{16}$$

  - Q penalizes **state deviations**, where each of the two state variables (e.g., position and velocity) is equally weighted with a penalty of 10. This ensures a strong focus on minimizing state errors.

  - R penalizes the **control input**. Increasing it discourages large or aggressive control actions, promoting smooth and stable vehicle operation.

  The values presented for Q and R are the typical default values we try in the first place. Luckily for us, they worked well enough from the start, giving a decent balance between accurate state tracking and reasonable control efforts.

The **prediction horizon** H determines the number of future time steps over which the MPC controller optimizes the control inputs. For our car's MPC controller, we choose a horizon of $H = 15$.

This choice balances between computational efficiency and control performance. A shorter horizon would reduce computation time but might not capture enough future behavior to make accurate predictions. A longer horizon could improve performance but increase computational complexity. By selecting $H = 15$, we ensure that the controller has sufficient foresight to plan effective control actions while maintaining real-time feasibility.

- Terminal invariant set for the lateral subsytem

Our MPC-controller is trying to reach a terminal set within N steps. This terminal set $\mathbb{X}_f$ is the maximum controlled invariant set, meaning it is the largest set for which there exists a control input that will maintain the system in the set. Essentially, if we are in the set and we have the correct control law giving us the appropriate control input for each time step, we will stay in the set forever. This maximum control invariant set is far away from the state and input constraints borders, so a simple control law like LQR can be used without risk.
A pre-set of $\mathbb{O}$ contains all the states which will be brought inside of $\mathbb{O}$ at the next timestep by the system's dynamics. By definition, all states in an invariant set must still be in the invariant set at the next timestep, so the pre-set of an invariant set is itself. To find the terminal set, we use an iterative approach : at every iteration we compare the set with its pre-set using polytope intersection. Once the pre-set and the set are equal, we have found an invariant set. Our starting set is the set $\mathbb{X}$ defined by our constraints and closed-loop (i.e controlled by the LQR) dynamics, so we know that the terminal control invariant set we find is inside of these constraints as well.

In figure 1 we can see the $\mathbb{X}_f$ for the latitude subsystem. We can observe that its limits for y and θ are significantly smaller than the original constraints imposed on the variables.

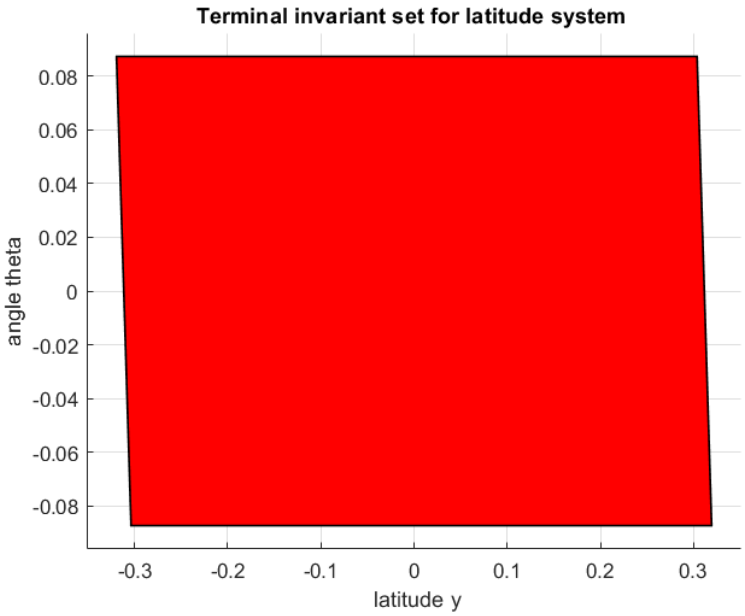
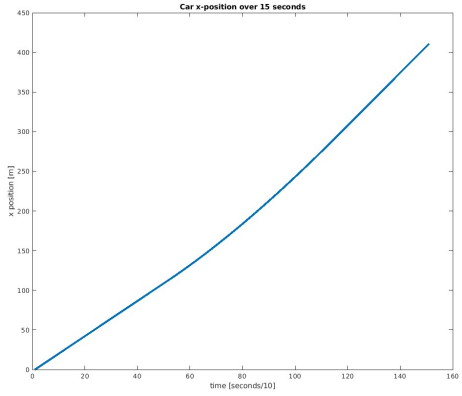
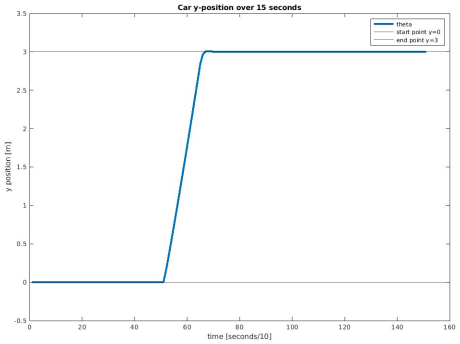


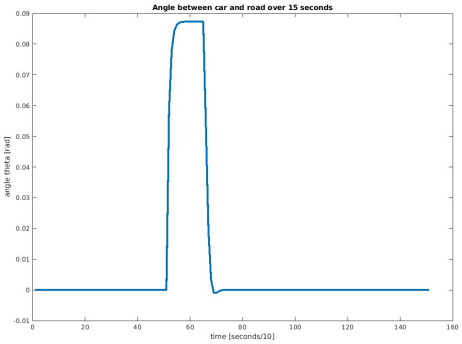
Figure 1: Terminal invariant set $\mathbb{X}_f$ of the latitude subsystem



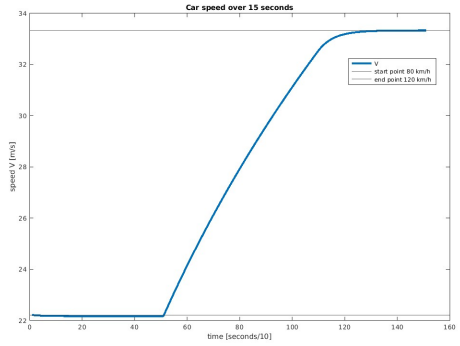
(a) x



(b) y



(c) theta



(d) V

Figure 2: Closed loop plots for each dimension of state vector X

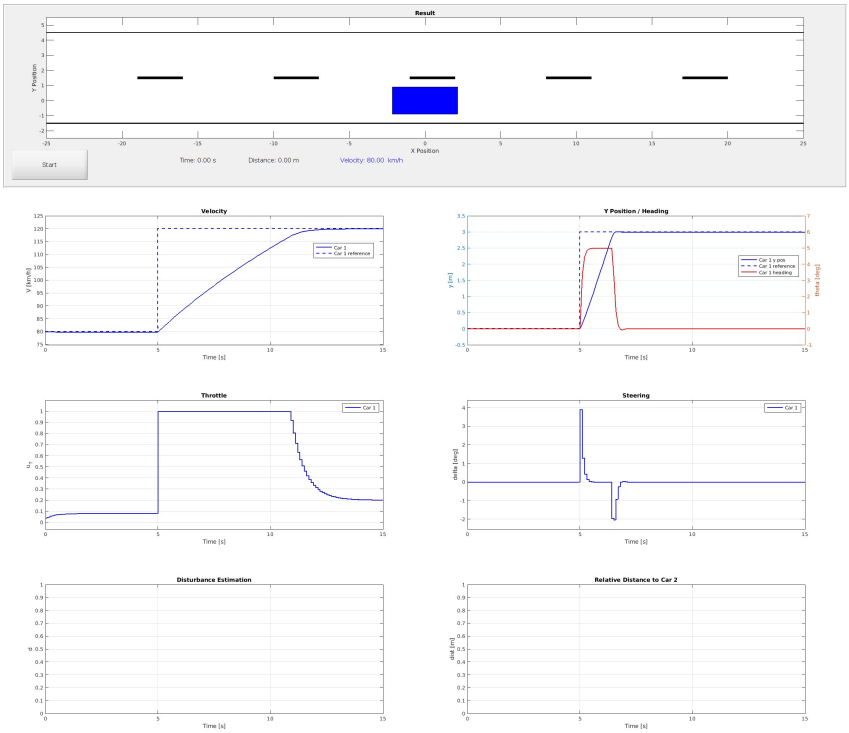


Figure 3: MPC controller simulation for the longitude (Left) and latitude (Right) systems

4 Offset-Free Tracking

4.1 Deliverable

- Explanation of your design procedure and choice of tuning parameters.

  To achieve offset-free speed tracking in the presence of disturbances, we extend the system's state to include an estimate of the disturbance d, forming the extended state vector: $z = \begin{bmatrix} V \\ d \end{bmatrix}$, where V is the velocity. The extended state dynamics are then given by:

$$z_{k+1} = A_{\text{hat}} z_k + B_{\text{hat}} u_k + L \cdot \left(C_{\text{hat}} z_k - y_k \right), \tag{17}$$

  where $L = \begin{bmatrix} L_x \\ L_d \end{bmatrix}$ is the observer gain and y_k the measured state.
  The measured state is given by the equation

$$y_k = C_{hat} * z_k + v = Cx * x_k + C_d * d_k + v \tag{18}$$

  The C_{hat} matrix is the measurement or sensor matrix and v is white noise. The first element C_x tells us how the sensor measures the state, and the second element C_d pertains to noise. In our case, we have a perfect and noiseless measurement of speed, so $C_x = 1$ and $C_d = 0$. We also have no white noise, so the equation for the measured state is greatly simplified and becomes simply

equal to real state (we have a perfect sensor).

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ d_k \end{bmatrix} + 0 = x_k \tag{19}$$

The matrices A_{hat}, B_{hat} are created from the A_d, B_d and matrices of our discretized system and defined as follow:

$$A_{\text{hat}} = \begin{bmatrix} A_d(2,2) & B_d(2) \\ 0 & 1 \end{bmatrix}, \quad B_{\text{hat}} = \begin{bmatrix} B_d(2) \\ 0 \end{bmatrix}, \quad C_{\text{hat}} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{20}$$

As presented in lecture 6, the error dynamics for this estimator are given by

$$\begin{bmatrix} x_{k+1} - \hat{x}_{k+1} \\ d_{k+1} - \hat{d}_{k+1} \end{bmatrix} = \left(\begin{bmatrix} A_d(2,2) & B_d(2) \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} \begin{bmatrix} C_x & C_d \end{bmatrix} \right) \begin{bmatrix} x_k - \hat{x}_k \\ d_k - \hat{d}_k \end{bmatrix}$$

We want these error dynamics to be stable (i.e have eigenvalues smaller than 1) such that the error converges to 0 and we obtain a precise estimate. To do this, we design poles with values smaller than unity in the observer gain L. Small pole-values (< 0.5) will make the dynamics converge faster but may lead to numerical instabilities, which is why we chose medium pole values of [0.5 0.6]. To calculate the L we use the place function : $L = -\text{place}(A_{\text{hat}}^\top, C_{\text{hat}}^\top, \text{poles})^\top$.

To compute the steady-state targets for velocity V_s and input u_s, we adapt the dynamics to include the disturbance estimate d_{est}. Let x_{ref} represent the reference state vector (which in fact contains only the speed element for the longitudinal subsystem). The steady-state velocity is set to: $x_{\text{ref}} = \begin{bmatrix} V_{ref} \end{bmatrix}$ (we can set the reference velocity to be equal to the steady-state velocity since we have perfect measurement as seen above). We are given the state dynamics

$$x^+ = f_d(x_s, u_s) + Ad \cdot (x - x_s) + B_d \cdot (u - u_s) + B_{hat} \cdot d_{est} \tag{21}$$

We assume that the system operates at (or near) steady state, and our goal is to track a reference x_{ref}. At steady state, we have $x^+ = x$, and tracking a reference implies that x can be replaced by x_{ref}. In the longitudinal system, $f_d(x_s, u_s)$ only corresponds to $\overline{x}_s = \begin{bmatrix} Vs \end{bmatrix}$. Plugging in these values yields :

$$x_{ref} = x_s + Ad \cdot (x_{ref} - x_s) + B_d(u_{ref} - u_s) + B_{hat} d_{est} \tag{22}$$

This equation allows us to extract the steady-state reference input :

$$u_{ref} = \frac{x_{\text{ref}} - B_{hat} \cdot d_{est} - x_s - A_d \cdot (x_{\text{ref}} - x_s)}{B_d} + u_s, \tag{23}$$

where x_s and u_s are the nominal steady-state values for the system.

By incorporating d_{est} into the steady-state computation, the controller effectively compensates for

disturbances, ensuring the system tracks the reference velocity $x_{\text{ref}} = \begin{bmatrix} V_{ref} \end{bmatrix}$ with zero steady-state error.

In 4 we show that our offset-free controller behaves appropriately for a speed and lane change.

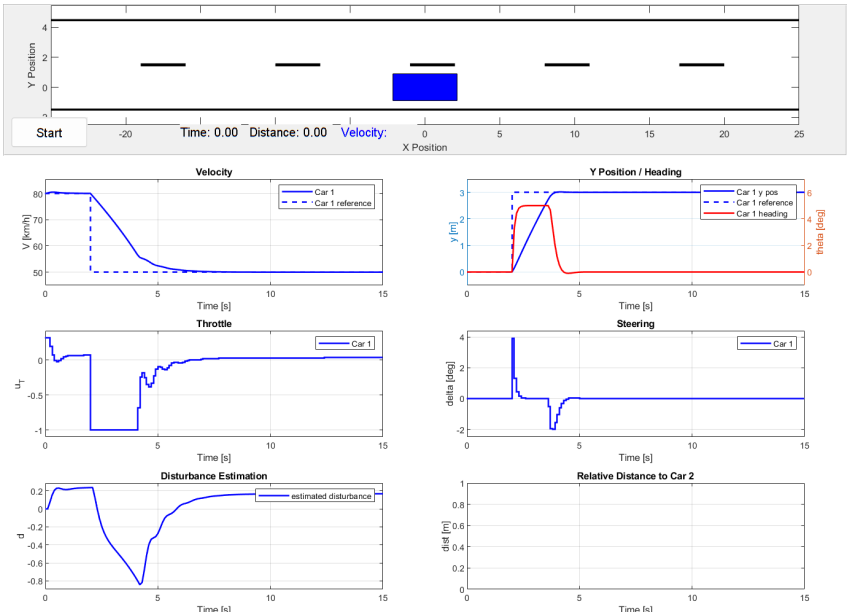
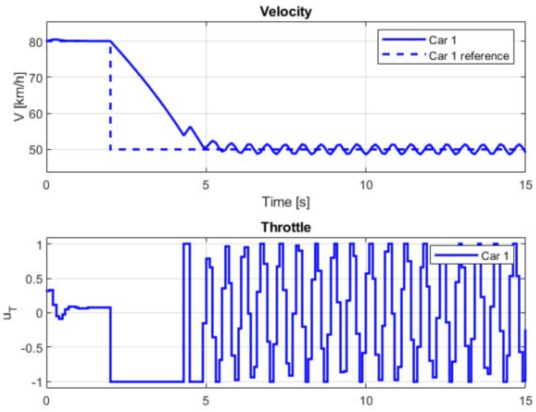


Figure 4: Controller with offset-free

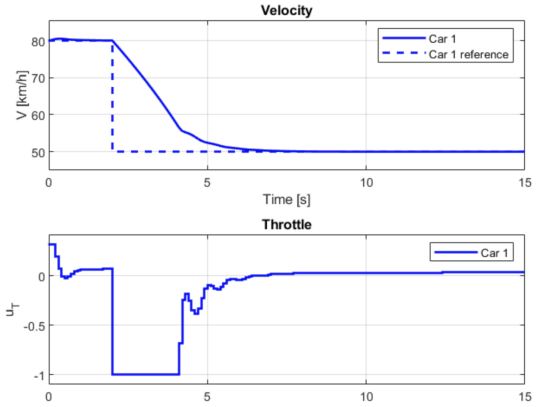
For the choice of tuning parameters, we started with the default parameters as usual:

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad R = \begin{bmatrix} 1 \end{bmatrix} \tag{24}$$

This worked out of the box for the latitudinal controller, but not for the longitudinal controller. For the later, the controller changed the throttle way too aggressively, which created very strong oscillations around the reference state instead of stabilizing (see 5a). To solve this, we increased the input penalty of the control input R up to 10 (for the longitudinal system only). Increasing the cost of control inputs forced the controller to act more smoothly, which achieved stable and correct tracking (see 5b).



(a) R=1 : too aggressive input control



(b) R=10 : smoother input control leads to stable tracking

Figure 5: Impact of R on longitudinal controller

5 Robust Tube MPC for Adaptive Cruise Control

5.1 Deliverable

- Explanation of the design procedure.

For this part, the longitudinal dynamics of the car we control (the blue car) are redefined to be relative to the car on front of it (the red car). The variable we will control is now the difference Δ between the states of the blue and red car. In the following, a tilde on a variable denotes that the variable pertains to the red car (so $\tilde{u_T}$ is the red car's throttle for example).

$$\boldsymbol{\Delta}^+ = A\boldsymbol{\Delta} - Bu_T + B\tilde{u_T} \tag{25}$$

where

$$\boldsymbol{\Delta} = \begin{bmatrix} \Delta_x \\ \Delta_V \end{bmatrix} = \tilde{\boldsymbol{x}}_{long} - \boldsymbol{x}_{long} - \boldsymbol{x}_{safe} \tag{26}$$

Red's throttle $\tilde{u_T}$ is treated as a disturbance (w), bounded by 0.5 above and under the steady-state throttle : $\tilde{u_T} \in W := [u_{T,s} - 0.5; u_{T,s} + 0.5]$.

To create a noise-robust controller, we will use the tube-MPC approach. Tube-MPC splits the control in two parts : the first ("nominal system controller") ignores noise and does normal MPC by using tightened constraints, while the second is charged with minimizing the error due to noise between the nominal system and the actual system.

The first step is to separate the system dynamics in two different equations.

$$z^+ = Az + Bv \quad \& \quad e^+ = (A - BK)e + w \tag{27}$$

Those two equations represent respectively : the guiding of the system to the origin (in a noise-free form), and the compensation for deviations from the first system (entirely driven by noise). Now how

do we implement tube-MPC ? The method is divided in two parts, one offline and the other online. We implement the offline part in a new file called tube_mpc_sets.m in which we will choose a stabilizing linear controller K and calculate the tightened constraints $\tilde{X}$ and $\tilde{U}$. The first step to compute those tightened constraints is to find the minimal robust invariant set ϵ_∞ following the rule $\epsilon_{i+1} = (A-BK)\epsilon_i \oplus W$. Once ϵ is found, we apply the followings Pontryagin Differences to find the tightened constraints:

$$\tilde{X} = X \ominus \epsilon \quad \& \quad \tilde{U} = U \ominus K\epsilon \tag{28}$$

We then use their associated constraints matrices and vectors (M_{thight} and m_{thight}) to compute $\mathbb{X}_f$, our invariant set, defined by

$$F_{tight} \leq f_{tight} \quad \& \quad M_{tight}K \leq m_{tight} \tag{29}$$

We then transfer our offline results for sets and controller into our online part, where we proceed to solve the minimization problem.

Our cost function depends on the variables **z** and **v** and takes the form

$$V(z,v) = \sum_{i=0}^{N-1} I(z_i, v_i) + V_f(z_N) \tag{30}$$

We use the quadratic stage cost $l(v,z) = z_i^T P z_i + v_i^T R v_i$, with P calculated offline thanks to the Lyapunov function solver of Matlab: $P = dlyap((A-BK)^T, Q + K^T R K)$

For our terminal cost, we recall the optimal cost of the LQR control law

$$V_f = z_N^T Q_f z_N \tag{31}$$

We implement the following loop:

for $i = 1 : N-1$

$$\begin{aligned}
&\text{con} = [\text{con},\ z(:,i+1) = Az(:,i) + Bv(:,i)];\\
&\text{con} = [\text{con},\ e(:,i+1) = (A-BK)e(:,i)];\\
&\text{con} = [\text{con},\ u(:,i) = K(\Delta(:,i) - z(:,i)) + v(:,i)];\\
&\text{con} = [\text{con},\ M_{tight}v(:,i) \leq m_{tight}];\\
&\text{con} = [\text{con},\ F_{tight}z(:,i) \leq f_{tight}];\\
&\text{con} = [\text{con},\ \text{abs}(u_i) \leq 1];\\
&\text{obj} = \text{obj} + z(:,i)^T P z(:,i) + v(:,i)^T R v(:,i);
\end{aligned}$$

end

The equation $u_i = K(x_i - z_i) + v_i$ allows the real trajectory to stay close to the nominal one.

To ensure that the state of our last step lies inside the terminal set $\mathbb{X}_f$, we also implement the

following constraint after the loop:

$$\text{con} = [\text{con},\ X_f.A * z(:, N) \leq X_f.b];$$

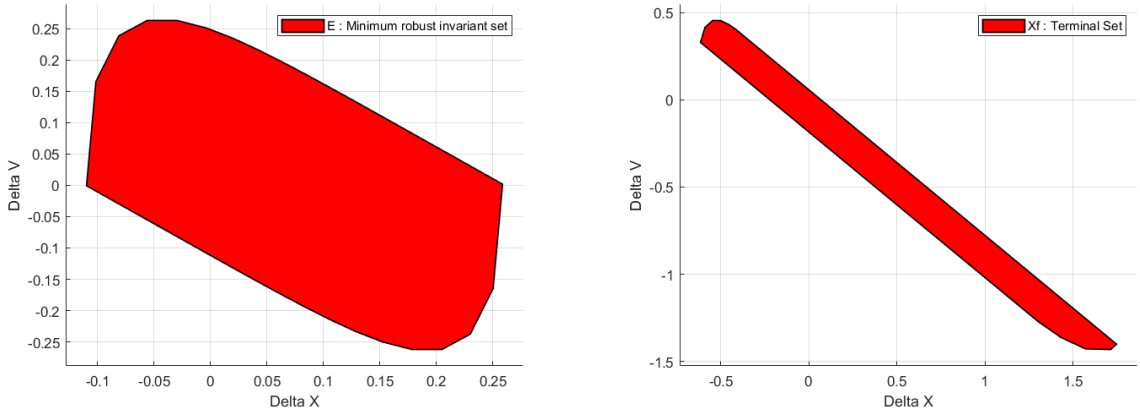
Where $X_f.A$ and $X_f.b$ extract the matrices defining the terminal set.

- Explanation of choice of tuning parameters

$$Q = \begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix} \quad \& \quad R = 1 \tag{32}$$

Using a larger Q than R will result in a smaller minimal invariant set E. This allows for more precise state tracking. As a reminder, if we imagine that our (noiseless) nominal system's state is at the origin, E is the maximum set of all possible states in which our real system (with noise) could actually be. So around the line-trajectory given by the nominal system, we draw a tube of the size E which gives us all the possible states in which our real system could be over the trajectory (hence the name tube-MPC). Therefore the smaller E is, the closer we are to a zero-noise known trajectory. Another way to see it is that a higher Q/R ratio leads to a bigger value of K, which translates into a more aggressive control and therefore a smaller minimal invariant set. However, R is used to penalize control effort, the higher it is, the smoother our control inputs will be. Having an R too small could lead to impractical aggressive actions, which would not be appreciated by our passengers.

- Plots of the minimal invariant set E and terminal set $\mathbb{X}_f$.



(a) Minimal invariant set E

(b) Terminal set $\mathbb{X}_f$

Figure 6

- Plots showing the robust controller working in different configurations.

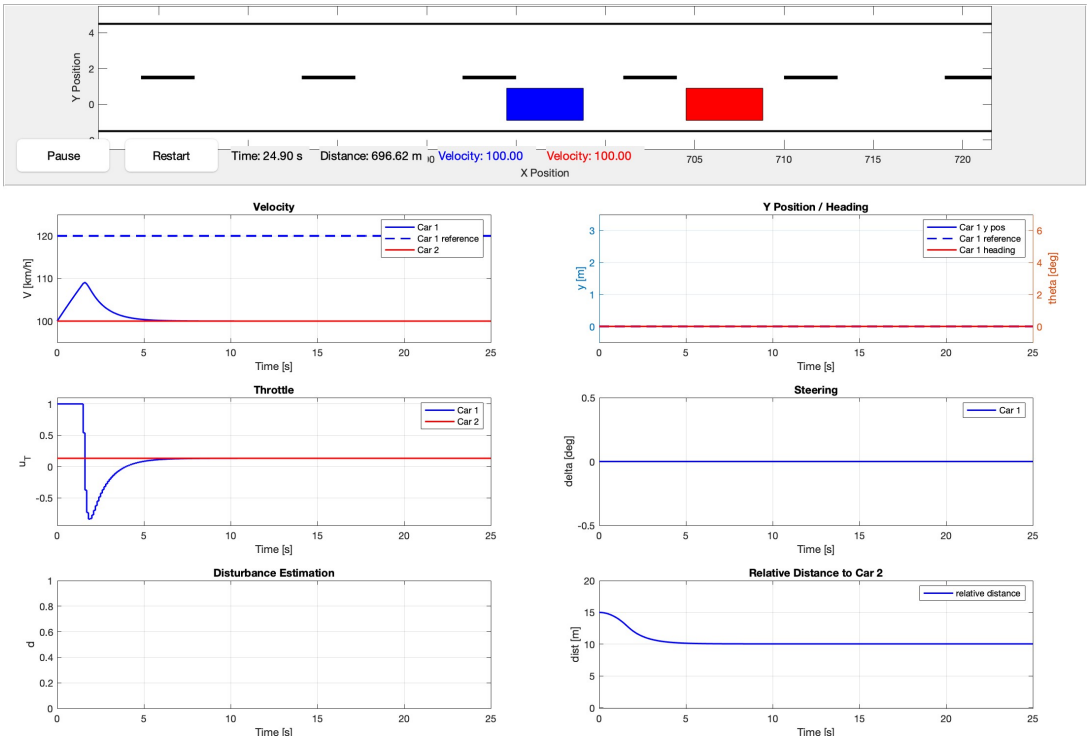


Figure 7: Robust MPC - Situation 1

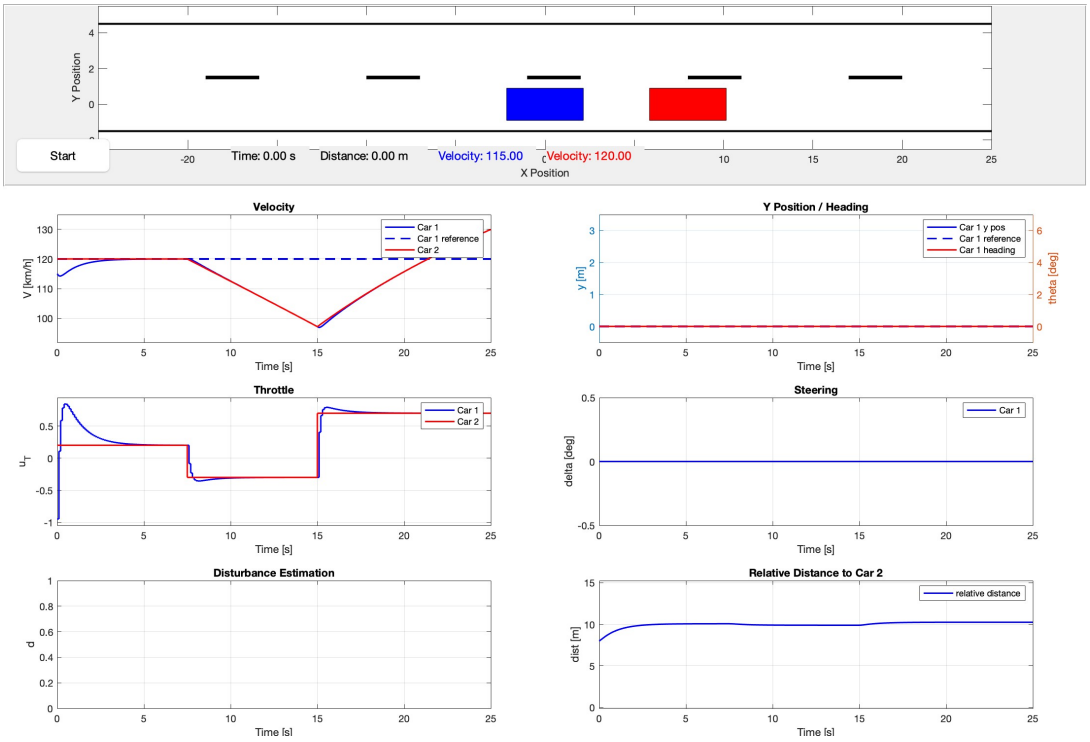


Figure 8: Robuste MPC - Situation 2

6 Nonlinear MPC

6.1 Deliverable

- Explanation of your design procedure and choice of tuning parameters.

  The Nonlinear Model Predictive Control (NMPC) in this context is designed to optimize the control of a vehicle while anticipating its future behavior and avoiding collisions with another vehicle during an overtaking maneuver. It uses real-time optimization to calculate the control inputs that ensure the vehicle follows a desired trajectory while respecting constraints. To implement it, we proceed as follows :

  1. Vehicle Dynamics :

     The vehicle's dynamics are modeled using a discrete-time equation:

     $$f_{\text{discrete}} = @(x,u)\,\text{RK4}_{\text{car}}(x,u,h,@car.f) \tag{33}$$

     This line defines a discrete-time function f_{discret} which uses the Runge-Kutta 4th order integration method (RK4) to update the state of a car model.

  2. Cost Function :

     The cost function balances trajectory tracking and control effort:

     $$J = \sum_{k=1}^{N} \left[1000 \cdot (x_{4,k} - v_{\text{ref}})^2 + 1000 \cdot (x_{2,k} - y_{\text{ref}})^2 + 0.01 \cdot u_{1,k}^2 + u_{2,k}^2\right], \tag{34}$$

     where:

     - $x_{4,k}$: Velocity.
     - $x_{2,k}$: Lateral position.
     - $u_{1,k}$: Steering angle.
     - $u_{2,k}$: Throttle.

     We selected the weighting factors for each parameter to ensure smooth driving. The steering input δ must not be too aggressive, and both the speed and the position of the car should remain as close as possible to the reference values.

  3. Constraints :

     The optimization problem includes the following constraints:

     - Dynamic Constraints

       The dynamics of the vehicle are enforced:

       $$x_{k+1} = f_{\text{discret}}(x_k, u_k).$$

– Control Input Limits

The control inputs are bounded:

$$-0.5236 \leq u_{1,k} \leq 0.5236, \quad -1 \leq u_{2,k} \leq 1. \tag{35}$$

– State Limits

The states of the vehicle are constrained:

$$-0.5 \leq x_{2,k} \leq 3.5, \quad -0.0873 \leq x_{3,k} \leq 0.0873, \tag{36}$$

where $x_{3,k}$ is the vehicle's orientation angle.

4. Optimization Problem :

The NMPC optimization problem can be formulated as:

$$\min_{U,X} J \tag{37}$$

(with J according to 34)

subject to the constraints defined above, where $U = \{u_1, u_2, \ldots, u_N\}$ is the control trajectory, and $X = \{x_1, x_2, \ldots, x_{N+1}\}$ is the state trajectory.

- Describe whether or not your controller results in steady-state tracking error. Explain why.

In our results (Figure 9) we do not observe any steady-state tracking error. This shows that the cost function (34) efficiently penalizes the deviations of variables from their references. Furthermore, our choices of weighting factors prioritizes error reduction in position and velocity. This result is also obtained thanks to a sufficiently long prediction horizon of H=15 seconds (meaning 150 steps), allowing the controller to have enough foresight. Additionally, the optimization problem is updated at each timestep which allows it to correct for any deviations dynamically. Those factors ensure for the controller to minimize tracking error and achieve steady-state performance.

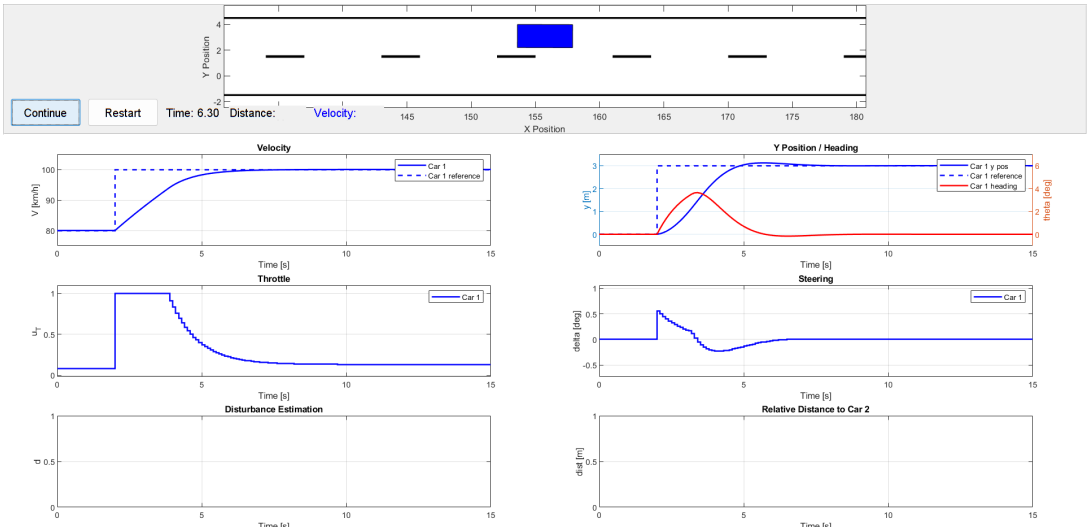


Figure 9: NMPC Controller

6.2 Deliverable

- Explanation of your design procedure and choice of tuning parameters.

To avoid collisions with other cars, we implemented a new constraint : if the blue vehicle is too close to the red one, it must change lanes to maintain its speed without colliding. For this purpose, we defined a safety zone in the shape of an ellipse, representing the area the blue car must avoid to prevent collisions. This is formulated with the following constraint:

$$(p_k - p_{L,k})^\top H (p_k - p_{L,k}) \geq 1, \tag{38}$$

where:

- $p_k = [x_{1,k}; x_{2,k}]$: Position x and y of the ego vehicle.

- $p_{L,k} = [\hat{x}_{1k}; \hat{x}_{2k}]$: Position x and y of the other vehicle.

-$H = \begin{bmatrix} \frac{1}{100} & 0 \\ 0 & \frac{1}{9} \end{bmatrix}$: Scaling matrix for the safety ellipse.

- The **position of the other car** is determined for each time step in the prediction horizon as follows:

  The x-position (p_{L1}) is calculated assuming constant velocity:

$$p_{L1}(k) = x_{\text{initial}} + v_x \cdot t_k, \tag{39}$$

  where $x_{initial}$ = the initial position of the other car at time t = 0, v_x = the other car velocity, and t_k is the discrete time step.

  The y-position (p_{L2}) remains constant over time:

$$p_{L2}(k) = y_{\text{initial}} \quad \text{for all } k, \tag{40}$$

where $y_{initial}$= initial latitude position of the other car.

These positions are combined into a matrix representing the trajectory of the other car:

$$\mathbf{p_L} = \begin{bmatrix} p_{L1} \\ p_{L2} \end{bmatrix},$$

which is used in the collision avoidance constraint to maintain a safe distance between the controlled vehicle and the other car.

- The **scaling matrix** for the safety zone between cars is calculate as follow :

  The inequality defining the safety zone is given by:

$$(\mathbf{p}_R - \mathbf{p}_{LR})^\top \mathbf{H} (\mathbf{p}_R - \mathbf{p}_{LR}) \geq 1$$

  where:

$$\mathbf{H} = \begin{pmatrix} H_{11} & 0 \\ 0 & H_{22} \end{pmatrix}.$$

  This expands as:

$$H_{11}(x_{1k} - \hat{x}_{1k})^2 + H_{22}(x_{2k} - \hat{x}_{2k})^2 \geq 1. \tag{41}$$

  We define:

$$a = x_{1k} - \hat{x}_{1k} \quad \text{and} \quad b = x_{2k} - \hat{x}_{2k},$$

  where a and b represent the semi-axes of the ellipse defined by the matrix $\mathbf{H}$ as illustrate in figure 10.

  To ensure a minimum safety distance between the two cars, we set:

$$a = 10\,\text{m} \quad \text{and} \quad b = 3\,\text{m}.$$

  Thus, the coefficients of the matrix $\mathbf{H}$ are given by:

$$H_{11} = \frac{1}{a^2} = \frac{1}{100} \quad \text{and} \quad H_{22} = \frac{1}{b^2} = \frac{1}{9}. \tag{42}$$

Our code for deliverable 6.2 takes a much longer time (1-2 min) to solve the NMPC problem of 6.1. We suppose that it has to do with poor implementation of the trajectory along the ellipse.
It's slow but it works !

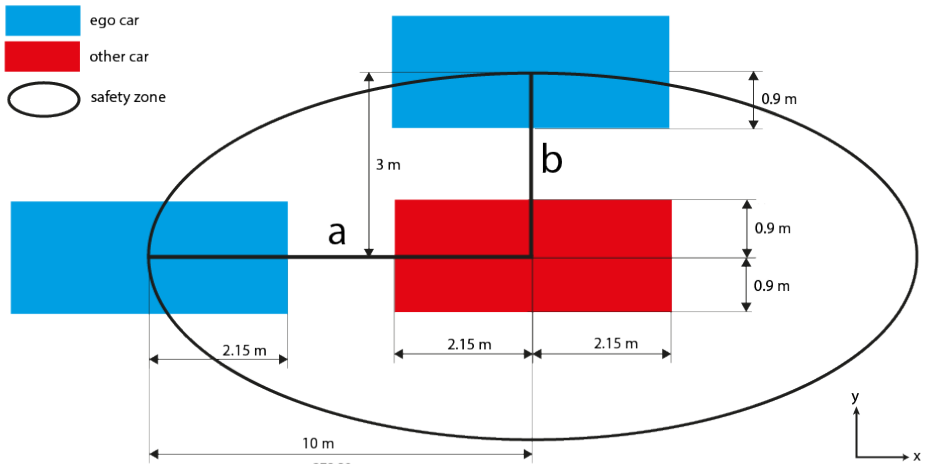


Figure 10: Safety zone between cars

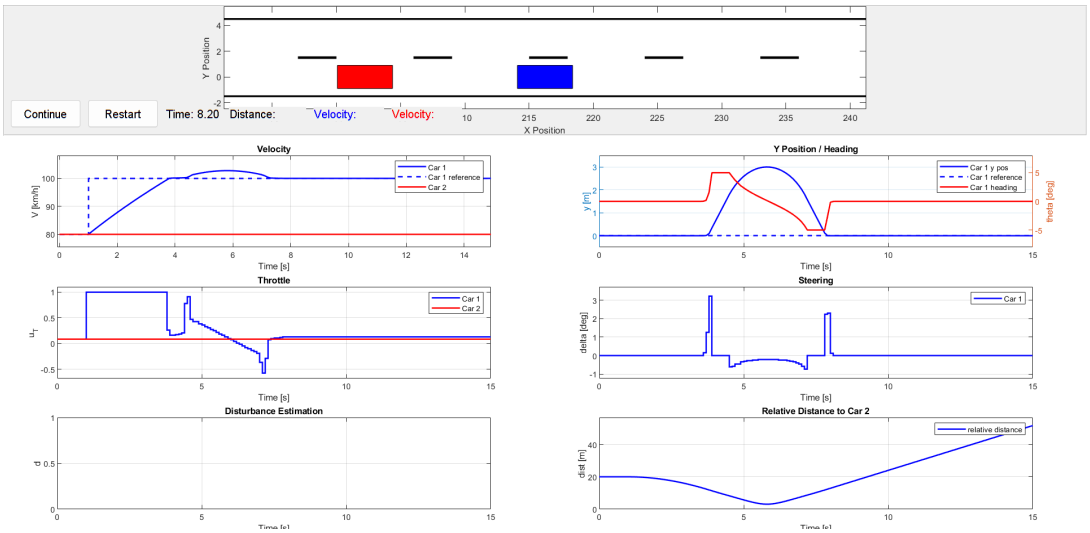


Figure 11: NMPC Controller for Overtaking