POLITECNICO
DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Master's Degree Thesis

Master of Science in Mechatronic Engeneering

NMPC Design For Autonomous Driving Applications

Coordinator:
Prof. Carlo Novara

Student:
Fabio Faliero

Accademic Year 2018/2019

Table of Contents

List of Figures

List of Tables

List of Codes

Introduction

In the last years, the automotive industry, has invested a huge amount of resources, in expanding automation and control systems for vehicles. This choice has been advanced for developing autonomous drive algorithms, passing through the implementation of increasingly accurate car management and safety systems.

The possible solution to obtain an autonomous driven car, has been classified in six levels, from 0 (manual drive) to the level 5 (autonomous drive without the need for the driver). Current systems are not only far from level 5, but even from the fourth one.

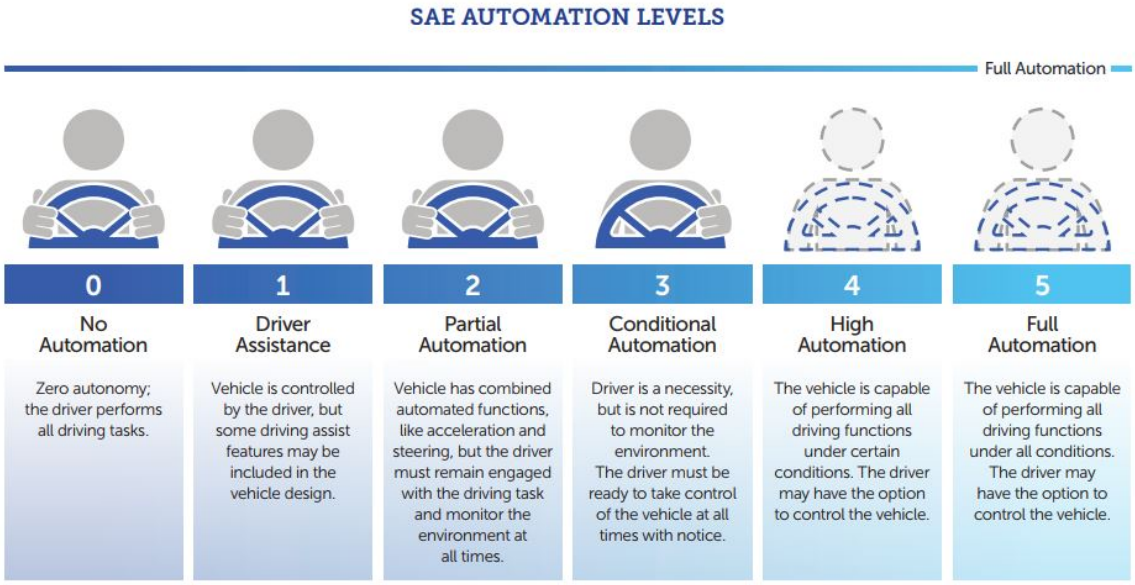


Fig. 1: SAE Automation Levels [1]

Current Advanced Driver Assistance Systems (ADAS) integrate different technologies for the detection of road and environmental conditions (using radar, ultrasound sensors, scanners, cameras, GPS etc.), to assist the driver in presence of potential dangers, hence, to reduce accidents event. The advancing technologies level and the algorithms define the capabilities of ADAS and their «depth» of intervention. Current Autonomous Driving system can be based on ADAS or algorithm that substitute, in some measure, the driver, in normal and predictable conditions.[2]
The essential elements that define an Autonomous driven vehicle are the perception, localization and control, and for each of them, exist a lot of different realization solutions.

The aim of this thesis realized under the supervision of Prof. Carlo Novara, is to delve into the use of an NMPC-based algorithm (Non-Linear Model Predictive Control) such as control law for the autonomous car, assuming measured vehicle states and obstacles perception.
Hence, it is necessary to evaluate the capability of the algorithm to generate robustly the engine or braking acceleration, and the steering angle inputs to drive the vehicle to the reference output.

The references could be both fixed or time-dependent, with or without the presence of fixed or moving obstacles depending on the application case. The path planning algorithm and the conversion of the sensors incoming information into constraint for the simulation system, could be assumed as known data, since it is necessary to study both of them separately for a more accurate study.

The study about algorithm has been realized simulating via computer some of the cases above, using the MATLAB® software, and the simulation environment Simulink®, which already includes some interesting features to implement an accurate analysis.

The whole thesis has been structured in four main sections. The first section talks about the vehicle models' study and explains the most significant ones used during simulations, distinguishing between the model involved in the control law, to predict the system behaviour, and a more accurate model used to shape a real case fashioned plant.
The latter reports its response to the signal generated by the NMPC. To reproduce a realistic dynamic behaviour of the actuator, it has been adopted a different block, placed between controller and plant, which also reproduces the nonlinear link between the intended model inputs and the ones managed by the vehicle.

The second part contains a general introduction to the NMPC algorithm, the optimization problem to be solved at each sample with the related mathematical formulation, and explains the meaning of the tunable parameters from the theoretical point of view.

The third section describes the application case, with the design and tuning of the controller, explaining the choices and the constraints used to build the prediction.

In the end, the last section involves the design of three tests justifing their need. The three simulations verify the effective working of the entire system, limits and capabilities related to the models used and at last one real case application of the NMPC controller for an ADAS.

1 Models Design

The model design plays a lead role for the study purpose, such that the NMPC working principle is based on the solution of a mathematical problem that refers to the prediction obtained from that model.
At the end of this section the purpose is to obtain the vehicle as a MIMO system described with state equations. Hence, is fundamental to choose the proper equations to describe the evolution of the system, reaching a compromise between accuracy and simplicity. To satisfy those conditions the model must make the calculation easier, otherwise a complex one leads to unfeasibility, with a compelling states description. Since the entire work is based on computer simulations, it is also necessary to design another model that differentiates from the prediction one. This model represents the real system and contributes to verify the robustness of the control law.

Once identified a restricted group meaningful models, the best configuration is the combination of those models both for plant and controller. To select the best one, in each case it has been asked to the controller to move the car, that starts in standstill condition, and stops to another fixed reference point on the plane.
The models, coming both from literature[3] and the Vehicle Dynamic Blockset, located in the simulation environment Simulink, are studied in a fashion of increasing complexity and accuracy.

All the models refer to a planar representation from the top point of view, with three degrees of freedom (3DoF) which are the two movements of the vehicle on its longitudinal and lateral axis and its rotation around the vertical axis. This kind of choice is oriented on the nature of the objective, that, for example. have to reach a reference position avoiding the obstacles on the trajectory. The last assumption about the environmental condition is null road inclination, hence, the action of the vertical forces is negligible.

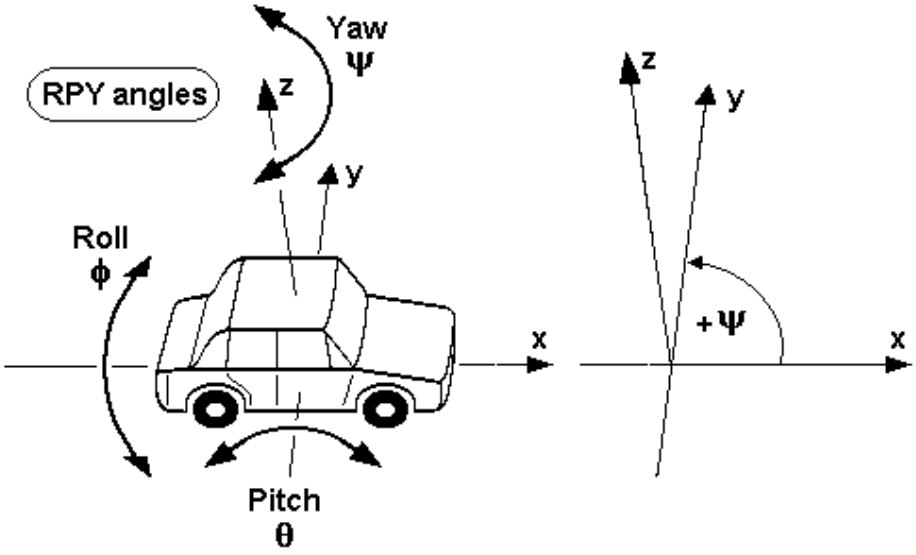


Fig. 2: Vehicle and Inertial RFs

The reference frame of the vehicle known as the body frame $F_{\mathrm{B}} = \{o, i, j, k\}$, with axes x,y,z, lies on its centre of gravity with the vertical axis oriented with the same direction of the inertial axis $F_{\mathrm{IN}} = \{O, I, J, K\}$, with axes X,Y,Z. The rotation of the body frame with respect to the inertial one around Z, represent the Yaw angle Ψ, with a positive direction as shown in the Figure **??**

1.1 Models classification

The models can be classified according to different characteristics, such as the equations used to represent it. There can be identified a kinematic model, which describes only the velocities of the body frame, neglecting the effect of the tyre's mechanic, which explains the longitudinal and lateral friction forces. To emphasize the contact of tyres with the ground and some other external forces that acts on the vehicle it is necessary to use a dynamic model.

An additional distinction comes from the study of the individual behaviour on each tyre using the so-called dual-track model, instead of the single-track model or "bicycle", which merges the wheels on the same axis (front or rear) in one that lies on the longitudinal axis. This simple representation reduces the number of calculation required, but neglects the different distribution of the vertical loads for each wheel, and a non-constant friction on the single tyres. According to this, there is no difference between dual or single-track in a kinematic model, therefore, by definition, there's no interest on vehicle dynamic. So it is sufficient to use the bicycle representation for the kinematic model, because there's no difference in the results using

One last distinction about the input system variables,is between quasi-stationary and non-stationary conditions. The first assumes the longitudinal acceleration of the vehicle equal to zero, so this kind of model uses the longitudinal velocity as one of the inputs that the controller generates. On the contrary thinking about the real working principle of cars, it is necessary to feed the machine with an engine torque and a braking torque.

			Stationary			Non-stationary			MATLAB			
			Kinematic	Dinamic		Kinematic	Dinamic		Dyn. Stationary		Dyn. Non-Stationary	
				Single Track	Dual track		Single track	Dual Track	Single track	Dual Track	Single track	Dual Track
S	Kin											
	Dyn	ST										
		DT										
NS	Kin											
	Dyn	ST										
		DT										

Table 1: Controller-Plant Model Configurations

Table 1 figures the whole combinations between plant and controller models distinguished above, and highlights the most significative ones deeply argued in the next pages. The combinations in

green represent the reliable models which provide an accurate result, while although the yellow ones are reliable the simulation shows some limits in reaching the target. Instead the red ones represent a failure in calculation due to the complexity of the system. At last the gray configurations can't be represented, because the complexity of the prediction model is higher than the plant one, so those combination are meaningless.

1.2 Relevant models

The most relevant models that shows up the design advance, are the ones defined in literature, while the Simulink models, although it offers a detailed vehicle dynamic, there is any relevant contribution to the research. The convention of these models also uses a body frame with an opposite vertical axis, hence, the states and the outputs must be rotated. Excluding those models, it is also possible to reduce the design errors due to inconsistency of the whole system. Note that any of the following models define the outputs of the system, since they depend on the application's objective.

1.2.1 Kinematic Stationary

The first model has a low level of complexity, according to some assumption that must follow, it is the stationary kinematic model, which equations describes the velocities of the CoG along the $X\ Y$ inertial axis and the angular velocity around Z. The equations come from the solution of a bicycle model, with only front steering wheel, and depend on the steering angle, the longitudinal velocity and the vehicle slip angle β. As shown in the Figure 3, l_f and l_r are respectively the distance from the CoG to the front and the rear axis.

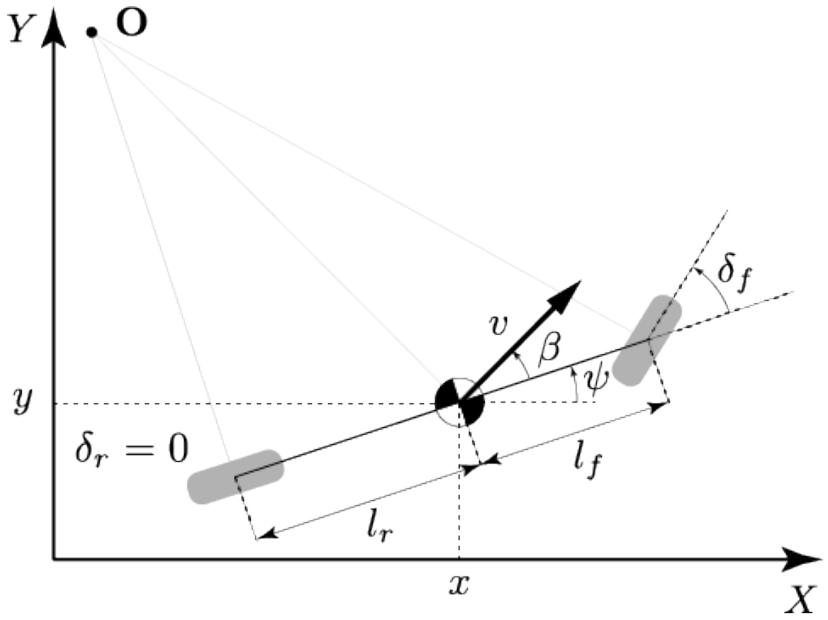


Fig. 3: Kinematic Single Track Vehicle Model

It is possible to the define the yaw rate, function of the steering angle, the longitudinal vehicle velocity and the vehicle slip angle. The vehicle slip angle is β, the one between the velocity v and the body frame longitudinal axis, and depends only on the geometry of the system Figure 3. The kinematic model introduces another important assumption about the wheel longitudinal velocity, indeed according to the hypothesis it lies precisely on the longitudinal wheel axis, hence the wheel slip angle is negligible, almost zero. The response to lateral contact forces on the tires it's also negligible. This last assumption is formulated under the hypothesis of low speed, under 5 m/s.

$$\begin{aligned}\dot{X} &= v\cos(\psi+\beta)\\ \dot{Y} &= v\sin(\psi+\beta)\\ \dot{\psi} &= \frac{v}{l_r}\sin(\beta)\end{aligned}$$

where

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

The inputs of this case of study are the vehicle velocity and the steering angle for the front wheel, while the states derived by the model are the position X and Y of the CoG with respect to the inertial frame, and the Yaw angle.

$$x = \begin{bmatrix}X\\Y\\\psi\end{bmatrix}, u = \begin{bmatrix}v\\\delta_f\end{bmatrix}$$

1.2.2 Single Track Stationary

With a higher speed the previous assumptions are no longer consistent, so, it is necessary describe widely the lateral vehicle dynamic. For this model it is still longer available the stationariness, that means a null acceleration on the vehicle longitudinal axis.

$$\ddot{x} = 0$$

This approximation is stronger when the autonomous system objective concerns the capability to maintain a cruise constant speed and small variations. The lateral dynamic is not affected by this choice; so, the vehicle behaviour still follows the second Newton law on the y axis:

$$
\begin{aligned}
a_y &= \frac{F_{yf} + F_{yr}}{m}, \\
F_{yf} &= 2\cos\delta_f F_{cf}, \\
F_{yr} &= 2F_{cr}
\end{aligned}
$$

Where F_{yf} and F_{yr} are the resulting lateral forces respectively on the front and rear axis of the bicycle system, while F_{cf} and F_{cr} are the lateral frictional tire forces.

$$
\begin{aligned}
F_{cf} &= -C_{af}\alpha_f, \\
F_{cr} &= -C_{ar}\alpha_r
\end{aligned}
$$

At the same time the acceleration a_y has one contribution $\ddot{y}$ related to the motion on y and one derived from the centripetal action $V_x\dot{\psi}$, hence the resulting equation are:

$$
\begin{aligned}
a_y &= \ddot{y} + \dot{\psi}\dot{x}, \\
a_y &= \frac{2}{m}(\cos\delta_f F_{cf} + F_{cr}), \\
\ddot{y} &= -\dot{\psi}\dot{x} + \frac{2}{m}(\cos\delta_f F_{cf} + F_{cr})
\end{aligned}
$$

Computing the equilibrium of the angular momentum around the vertical axis z, the resulting equation is:

$$
\ddot{\psi} = \frac{2}{I_z}(l_f F_{cf} - l_r F_{cr})
$$

In the tire frictional forces, C_a is the contribution of the cornering stiffness, and θ is the tire velocity angle. Those angles for both front and rear tyre are the following:

$$
\begin{aligned}
\theta_f &= \tan^{-1}\left(\frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right), \\
\theta_r &= \tan^{-1}\left(\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}\right),
\end{aligned}
$$

While the tires slip angles are:

$$
\alpha = \delta - \theta
$$

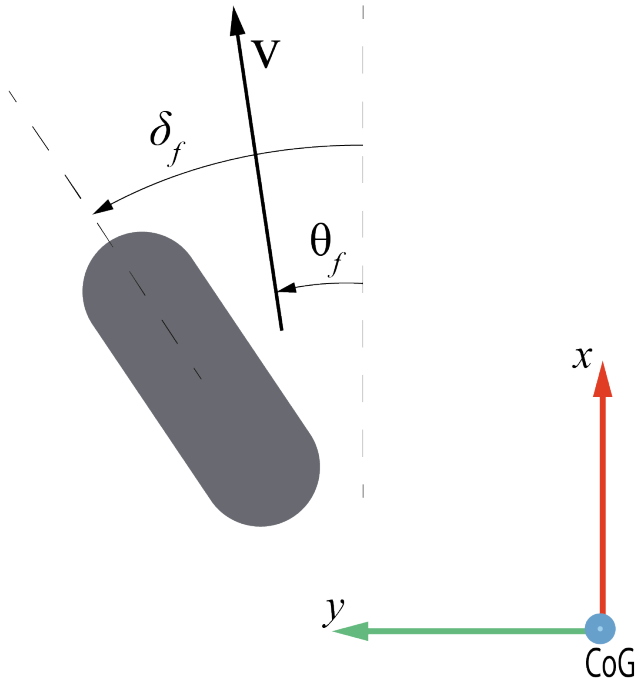


Fig. 4: Tire Slip Angle

Since the rear steering angle is equal to zero

$$\alpha_f = \delta_f - \theta_f$$
$$\alpha_r = -\theta_r$$

The CoG velocity with respect to the inertial frame, has a contribute of both the x and y velocities in the body frame.

Hence, the dynamic equations of the bicycle stationary system are the following:

$$\begin{aligned}
\ddot{x} &= 0, \\
\ddot{y} &= -\dot{\psi}\dot{x} + \frac{2}{m}(\cos\delta_f F_{cf} + F_{cr}) \\
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi, \\
\dot{Y} &= \dot{x}\sin\psi + \dot{y}\cos\psi, \\
\dot{r} &= \frac{2}{I_z}(l_f F_{cf} - l_r F_{cr}), \\
\dot{\psi} &= r
\end{aligned}$$

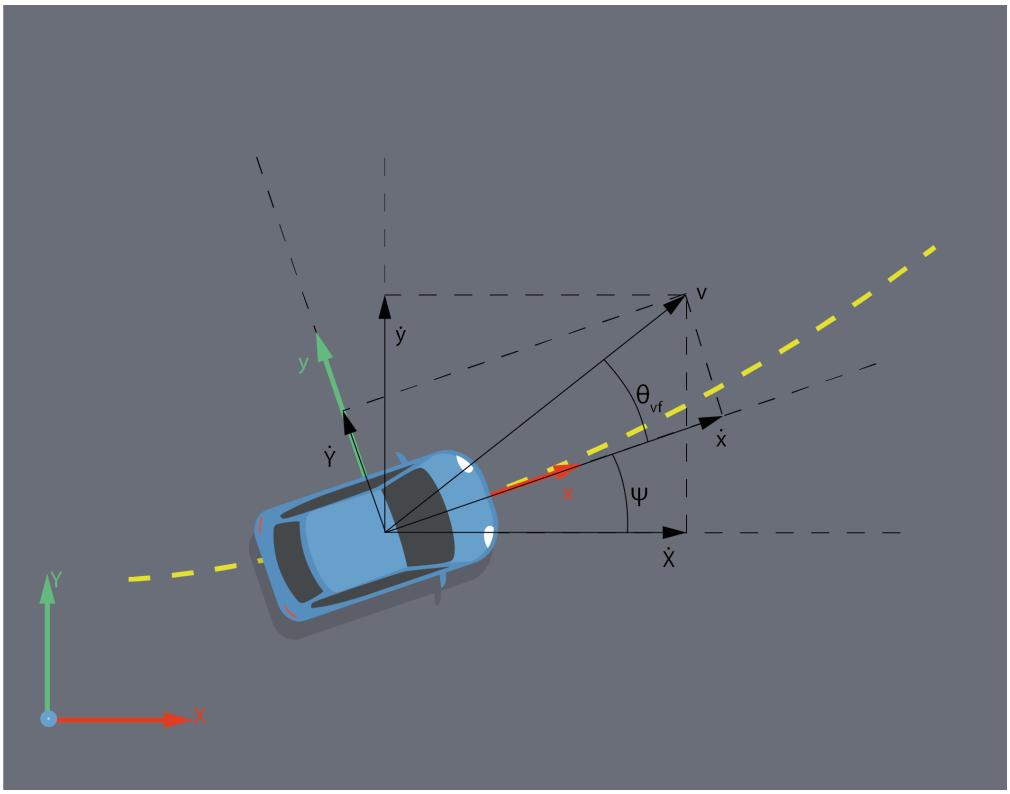


Fig. 5: Trigonometric decomposition of speed

The state vector contains more elements than the kinematic stationary one, while the inputs received are the same v and δ_f.

$$x = \begin{bmatrix} \dot{y} \\ X \\ Y \\ r \\ \psi \end{bmatrix}, u = \begin{bmatrix} v \\ \delta_f \end{bmatrix}$$

1.2.3 Kinematic Non-Stationary

To implement the acceleration input inside the model, it is necessary add one more additional state equation with respect to the kinematic stationary model, which define the longitudinal acceleration as the derivative of the vehicle velocity. However, this solution doesn't give any precise information about the longitudinal vehicle dynamic. So the mathematical model equations are:

$$\begin{aligned}
\dot{X} &= v\cos(\psi+\beta)\\
\dot{Y} &= v\sin(\psi+\beta)\\
\dot{\psi} &= \frac{v}{l_r}\sin(\beta)\\
\dot{v} &= a\\
\beta &= \tan^{-1}\left(\frac{l_r}{l_r+l_f}\tan(\delta_f)\right)
\end{aligned}$$

Since the input for non-stationary model is different, the new state and input vectors are:

$$x = \begin{bmatrix} X\\ Y\\ \psi\\ v \end{bmatrix}, u = \begin{bmatrix} a_x\\ \delta_f \end{bmatrix}$$

1.2.4 Single Track Non-Stationary

The last model relevant for the study case among the purposes, Is the non-stationary bicycle vehicle, which includes a dynamic description on both longitudinal and lateral axis, and the angular yaw dynamic. The longitudinal behaviour is a function of two contributes, $V_y\dot{\Psi}$ that refers to the centripetal acceleration, and the summation of the longitudinal forces acting on the vehicle. As for the previous cases, the basic assumption neglect any non-linear disturbance on the vehicle such as the drag force or other external disturbances, well defined in literature [3], that means a unique contribution of acceleration a_x that summarize the engine force and the braking one, which are the actually managed variables. Hence the resulting model equations are:

$$\begin{aligned}
\ddot{x} &= \dot{\psi}\dot{y} + a_x,\\
\ddot{y} &= -\dot{\psi}\dot{x} + \frac{2}{m}(\cos\delta_f F_{cf} + F_{cr})\\
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi,\\
\dot{Y} &= \dot{x}\sin\psi + \dot{y}\cos\psi,\\
\dot{r} &= \frac{2}{I_z}(l_f F_{cf} - l_r F_{cr}),\\
\dot{\psi} &= r
\end{aligned}$$

The states and the inputs vectors of this last MIMO system are:

$$x = \begin{bmatrix} \dot{x} \\ \dot{y} \\ X \\ Y \\ r \\ \psi \end{bmatrix}, u = \begin{bmatrix} v \\ \delta_f \end{bmatrix}$$

1.3 Final Models

For the final models to be used in the application tests, the choice takes into account the capability of those models and in which measure they can reach a fixed target in a plane. The model for the controller is the kinematic non-stationary, because offers a sufficient states definition to identify the vehicle position, although it contains a lot of assumption that simplify the plant's physical behaviour.

The non-stationary category, besides, uses the acceleration input that can be easily managed and led back to the real actuators inputs, such as the throttle valve angle for the engine, and the braking pressure, the responsible systems of the overall vehicle acceleration. Hence, the plant uses a dynamic Single Track non-stationary model, since the use of very accurate systems such as the dual track can't be justified by the research cases. In the following section there is a detailed description for both models, to understand within block schemes and codes, how they are implemented, and the successive modification that improves the robustness of the NMPC algorithm.

1.3.1 Controller

The kinematic controller model is enough to provide a correct prediction of the system states, with efficiency and robustness, indeed some variations are introduced into the plant. The code below is the Kinematic model representation for the system, that contains both the non-linear state equations (line 7 to 10), functions of the states and the inputs, and the output equations (line 12 to 14)for the basic application which consider the output to control the position X,Y and the velocity $\dot{x}$, but those last equations depend on the references used.

```
1  function [f,y]=KIN_model(t,x,u)
2
3  global  b  l
4
5  beta=atan2(b.*tan(u(2,:)),l);
6
7  f(1,:)=x(4,:).*cos(x(3,:)+beta);
8  f(2,:)=x(4,:).*sin(x(3,:)+beta);
9  f(3,:)=x(4,:).*sin(beta)./b;
10 f(4,:)=u(1,:);
11
12 y(1,:)=x(1,:);
```

```
y(2,:)=x(2,:);
y(3,:)=x(4,:);


end
```

Code 1: Kinematic Model State Equations Code

The expected limits of this model, as explained in [4], are related to the turning dynamic behaviour, since there's no description for that in the equations, hence in hard turning manoeuvres the system may produce understeering conditions. Indeed, since there's no distinction about engine and braking acceleration, their non-linear trend, and the meaning of each contribute, the algorithm may miss the objective for a certain distance, then continues to apply a negative acceleration to reach the position in reverse. This is not acceptable, hence, the solution is to modify the block system realizing some constraints that denies the negative velocity, and making the system aware about the nature of the braking acceleration contribute.

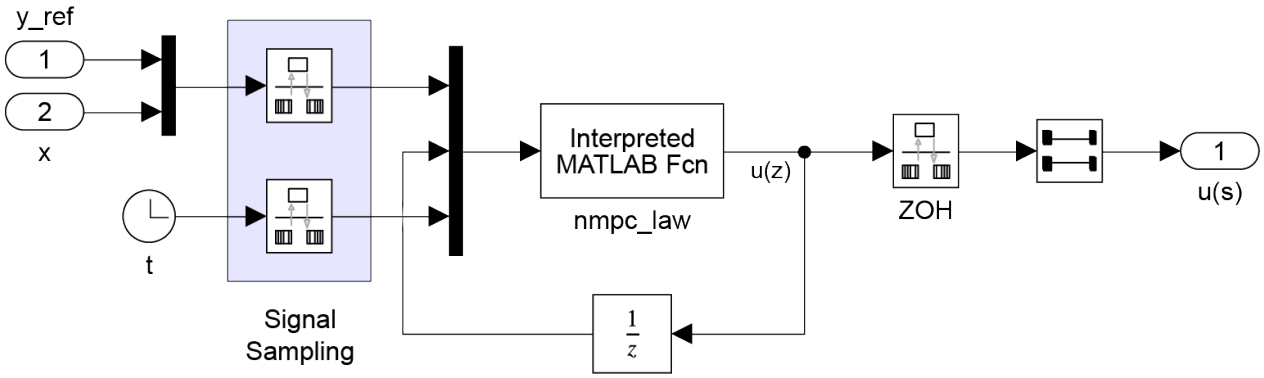


Fig. 6: Controller Block Scheme

1.3.2 Actuator

Between the controller and plant blocks, there is another important element that is the actuator, in which the incoming input signal, generated by the controller, are processed and interpreted. From the controller side of this block there is the dispatcher algorithm that may separate the engine acceleration from the braking one, and the transfer functions that compensate inputs' dynamic, Figure 7.

To separate the throttle valve input command to the braking pressure, the dispatcher takes as input the overall acceleration. An initial analysis lead to a simple threshold switch, such as the one used for Simulink block scheme. The drawback of this specific elements is the necessity of a very short simulation sampling time, that means a huge simulation time, otherwise the results are affected with a high gain noise in the switching instant.

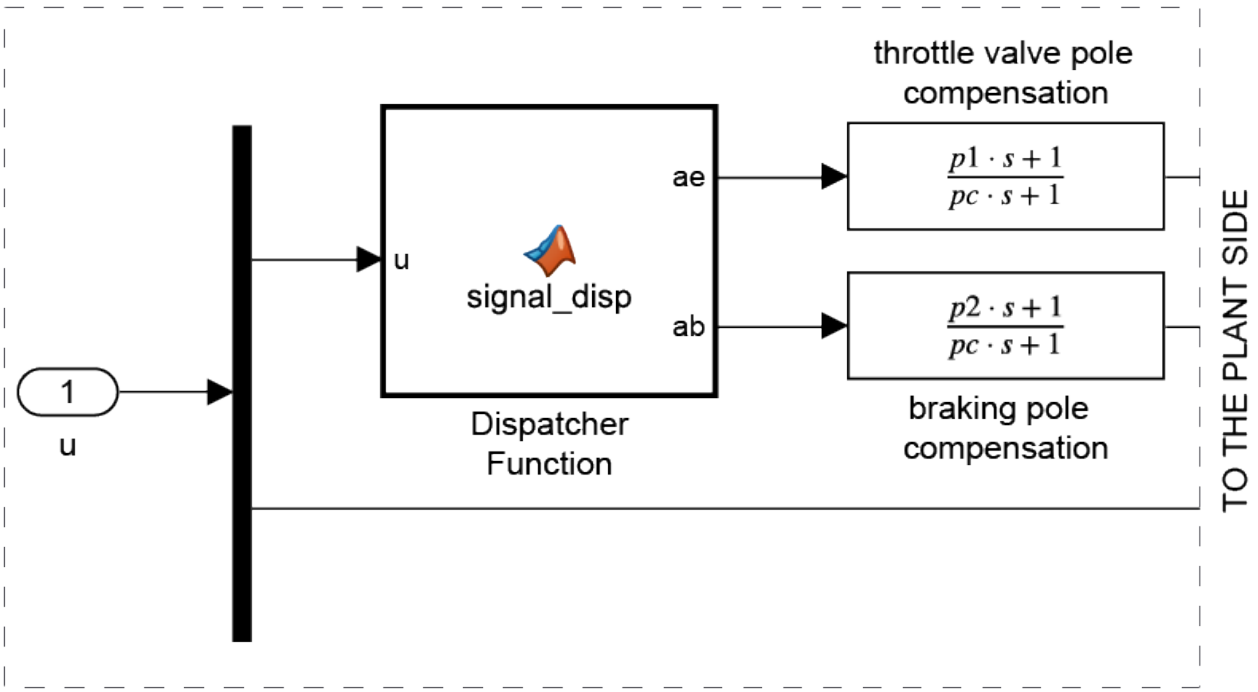


Fig. 7: Actuator Controller's Side

The solution reported uses the MATLAB function block, which contains the code for the switch algorithm.

```
function [ae,ab]=signal_disp(u)

thold=-0.2;   %acceleration threshold



if (u>=thold)
    ae=u;     %engine acceleration
    ab=0;     %braking acceleration
else
    ae=thold;
    ab=u-thold;
end


end
```

Code 2: Dispatcher Algorithm Code

The code-wise approach also allows to design a much-detailed dispatching method that can be used for both forward and reverse drive. The assumption for this basic algorithm distinguishes the purely engine acceleration field from the threshold to the maximum acceleration, while the deceleration is a mixed contribution of the engine braking (maximum at the threshold level) and the braking pressure of the brake calliper on the disk.

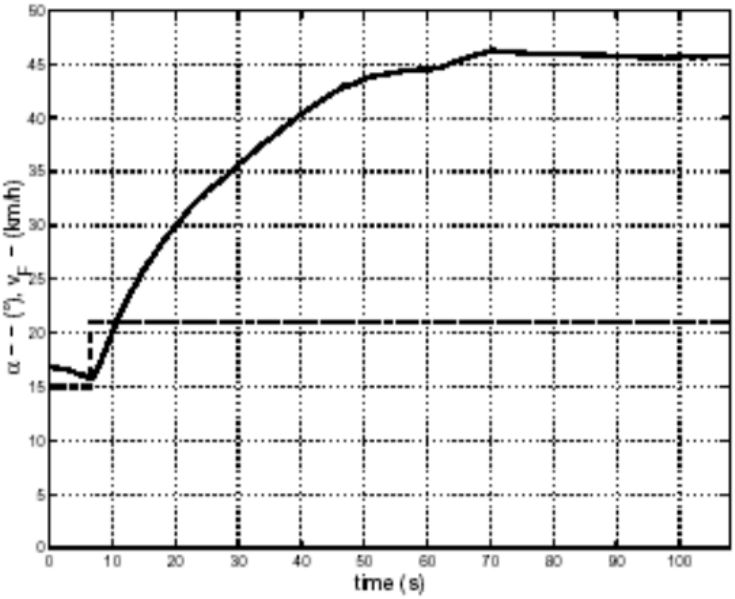


Fig. 8: First Order Velocity Dynamic

The unit measure for the inputs is an acceleration m/s^2, that must be scaled according to the actuator input argument, such as the throttle valve angle or the braking pressure. Since the study about inputs representation doesn't affect the aim of the thesis, there is a direct relationship between the requested input and the independent ones. A more accurate relationship can be examined in a different study, using this approach as the starting hypothesis.

In the end there are two transfer function blocks, one on each input branch that can compensate the dynamic response of the actuators to those impulses. The compensation law is a transfer function with a zero that cancel the current input dynamic, and one pole that substitute it with a faster one. The compensation rule is the same for the braking and acceleration signal. On the controller side the steering angle input doesn't require any modification.

On the plant side of the actuator block Figure 10, there are the transfer function that describes the dynamic of the input signal.

Gear	**Time Constant τ[s]**
1	25.7
2	35.7
3	30.1
Avg	30

Table 2: Time Constant Values

For the throttle valve signal the transfer function is the equivalent first order response with unitary gain, which time constant τ can be assumed constant as the average value of this constant for the

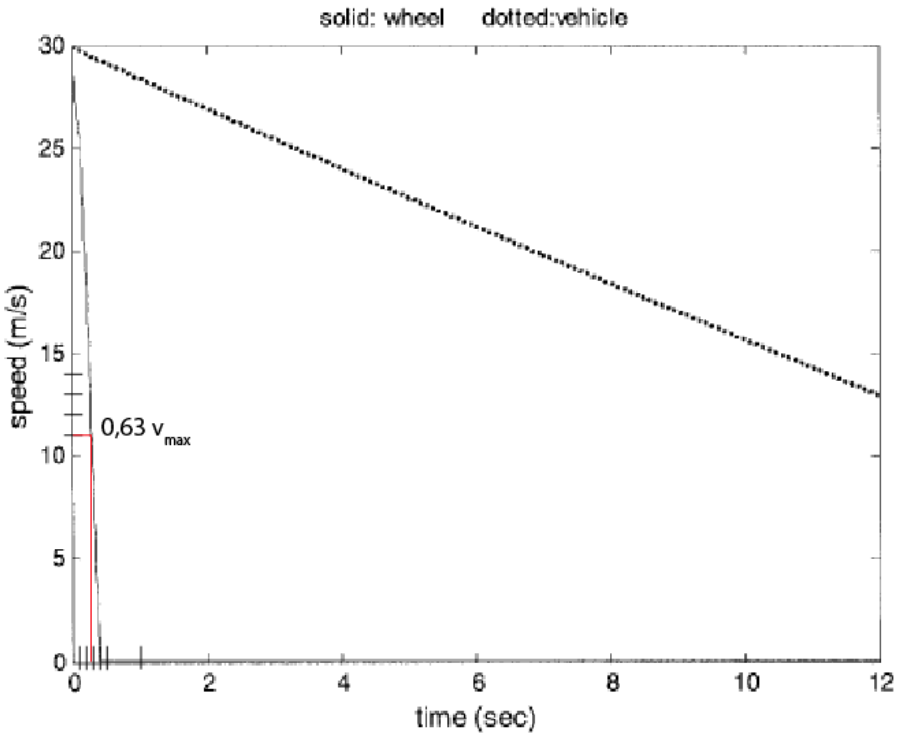


Fig. 9: Hard Braking Time Response

different gears 3. This approximation comes from the study about the cruise control [5], based on the response of a step variation on the throttle angle with a gain $K = 5$, Figure 8.

$$H(s) = \frac{1}{1 + s\tau}$$

This first order transfer function gives a simple but efficient representation for the inputs system dynamic, so, it can be also used with the braking pressure response. The transfer function's parameters depends on the response of the braking system found in literature [3]. The plot in Figure 9 refers to an hard braking time response, it means that the braking input signal is actually a step signal which can block the wheel from an initial velocity of 30 m/s in almost half second. To obtain the time constant of a step response the value must be the 63% of the final value.

$$0.63\, v_{max} = 11.1\ m/s$$

Reading this value on the plot in Figure 9 the corresponding time constant is almost $\tau_{braking} = 0.27\ s$

The steering angle input has been processed with a first order transfer function that slow down the variation on this value. The time constant value for this transfer function comes from several tries during the simulation, in order to obtain a feasible steering variation.

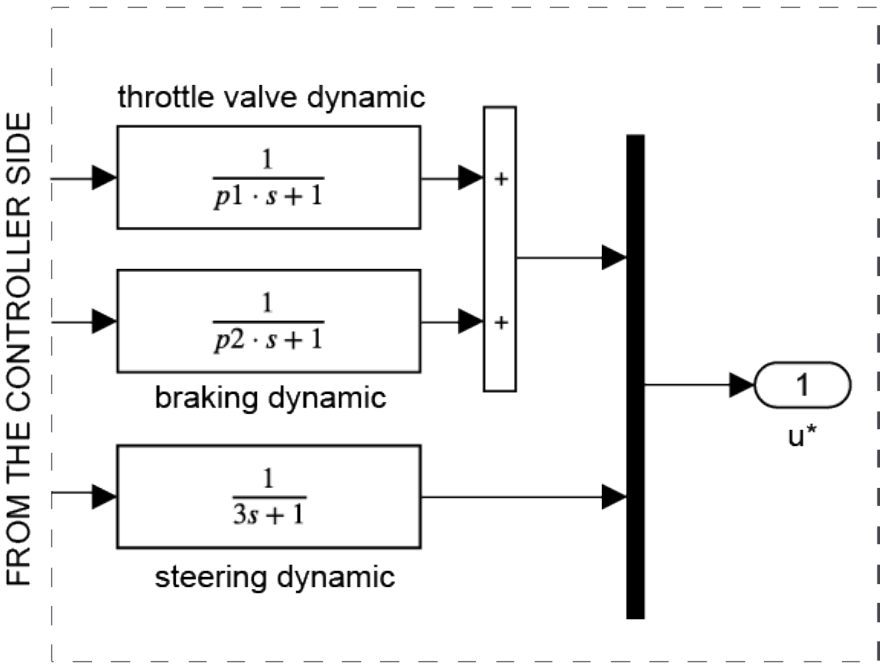


Fig. 10: Actuator Plant's Side

The following table summarise the time constant used in each first order transfer function to describe the related signal behaviour.

	Time Constants τ[s]
throttle valve (p1)	30
braking pressure (p2)	0.27
steering	3

Table 3: Time Constant Values

At the end the overall acceleration is sum of the braking and accelerating signals, thus can be the input for the single track plant model.

1.3.3 Plant

The final plant model is the single track non-stationary vehicle, which receives its inputs from the actuator block. Even if the model equations are sufficiently detailed to describe the plant, it is still necessary to modify the state equations, introducing a non-linear law during the braking. Hence, the model interprets a standing negative acceleration as the request to move in reverse when the vehicle speed is almost zero.

The braking force acts opposite with respect to the driving direction, since it is a frictional force,

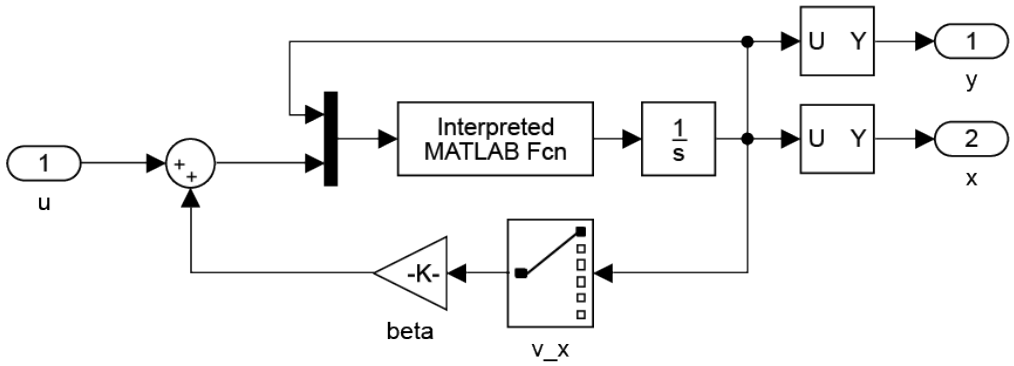


Fig. 11: Plant Block Scheme

therefore, when the vehicle stops (or its velocity is almost zero) and there is a braking pressure on the disks, the vehicle must remain stopped. This non-linear constraint has been described with an if condition in the code, when both the acceleration input is negative and the speed almost null, forcing the velocity, the longitudinal and lateral acceleration of the vehicle to zero.

```
function f=SINGLE_stateq(t,x,u)

global a b m I_zz C_yf C_yr thold

a_f=atan2(x(2,:)+a.*x(5,:),x(1,:))-u(2,:);
a_r=atan2(x(2,:)-b.*x(5,:),x(1,:));
F_cf=-C_yf.*a_f;
F_cr=-C_yr.*a_r;


if (u(1,:)<=thold && x(1,:)<=0.00001)
    x(1,:)=0;
    f(1,:)=0;
    f(2,:)=0;
else
    f(1,:)=x(5,:).*x(2,:)+u(1,:);
    f(2,:)=-x(5,:).*x(1,:)+(2/m).*(F_cf.*cos(u(2,:))+F_cr);
end


f(3,:)=x(1,:).*cos(x(6,:))-x(2,:).*sin(x(6,:));
f(4,:)=x(1,:).*sin(x(6,:))+x(2,:).*cos(x(6,:));
f(5,:)=(a.*F_cf-b.*F_cr).*(2./I_zz);
f(6,:)=x(5,:);

end
```

Code 3: Single Track Model

According to the kind of simulations, and the choice to adopt simpler models this braking behaviour must be described forcing states to zero, but this is not the best solution, as a matter of fact exist different models much complex that can use the separated inputs signal for brake and drive the car.

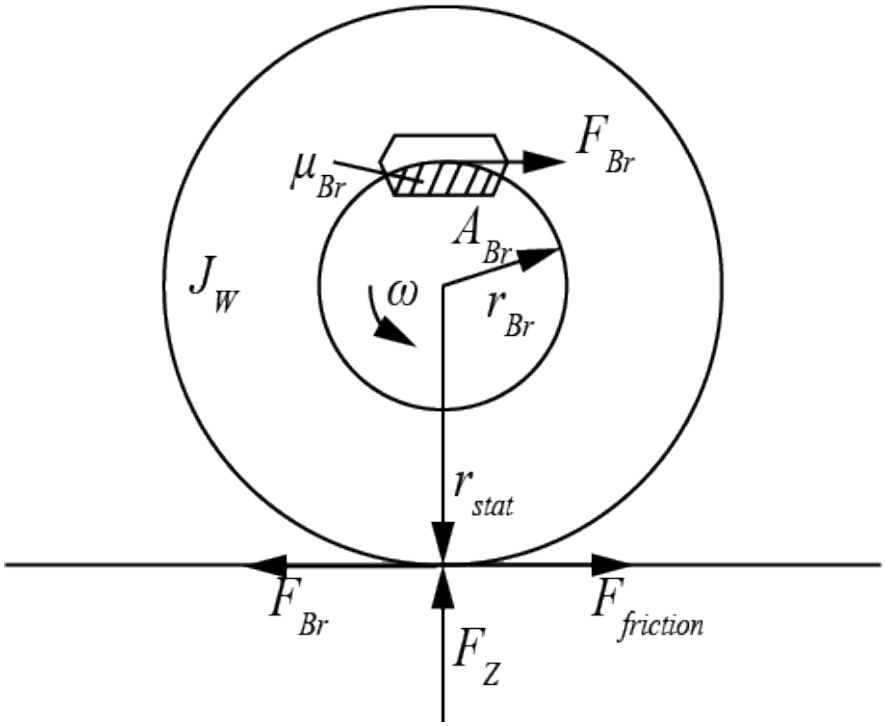


Fig. 12: Vehicle Braking Forces

Hence, it justifies why it is necessary to introduce a more accurate dispatching algorithm to setup a signal processing that fits with the plant.

In the end, inside the Simulink block scheme for the plant there is a block that contains the state equation code. The integration of its output gives the states, which provide the outputs, with a selector block, the feedback to the state equations and a disturbance that affects the input. This disturbance has been approximated as a linear function of the longitudinal vehicle velocity

$$\hat{u} = u + d$$
$$d = v_x \beta$$

Where

$$\beta \in \mathbb{R}^2$$

2 Introduction To NMPC

In the last years, the MPC has become the most standard advanced control method used in the process industries, in particular for those systems that can be adequately represented with a linear model. The increasing usage of that algorithm it's justified by its ability to handle large scale multi-variable processes with tens or hundreds of inputs and states that must fulfill physical and operational constraints.

The working principle of the MPC algorithm involves the formulation and solution of a numerical optimization problem corresponding to a finite-horizon optimal control problem at each sampling instant. Since the state of the system is updated during each sampling period, it is necessary to adopt the receding horizon approach, hence a new optimization problem must be solved at each sampling interval.

Linear models can be managed using a large variety of numerical methods and software, since the MPC problem in that cases is typically a quadratic or linear program, which is known to be convex. Unfortunately, as the linear MPC shows a numerical complexity that may result in challenging approach with the powerful computers available, the NMPC method represent a not negligible limit in its industrial impact due to the difficulties in guaranteeing a sufficiently good solution to the optimization problem within real-time requirements.

The studies concerning the nonlinear optimal control have already begun in 1950's and 1960's, followed soon by its strong characterizations such as the maximum principle, and dynamic programming. The optimal control, and in particular the Nonlinear Model Predictive Control (NMPC) practical implementation, simplifies significantly the control design as much as large and complex the system is, as a matter of fact this kind of algorithm represent an attractive solution among the alternatives.

However, this nonlinear control contains several limiting factors, from the difficulties to implement it in a real-time embedded controller, to the utilization of nonlinear dynamic system and estimated states. The nonlinear problem may require a large number of computations to perform at each sample, due to the multiple local minimal solutions that can be reached, even without guaranteeing to achieve the best optimal solution.
Hence, the NMPC involves the repetitive solution of an optimal control problem at each sampling instant in a receding horizon fashion. Unfortunately, there is no guarantee that the receding horizon implementation of a sequences of open loop optimal control solutions will perform well, or even be stable, when considering the closed loop system.

It follows that the NMPC is a powerful approach that has proven itself in several applications and with further research in the direction of numerical implementation technology and modeling and state estimation methods, it may strengthen its position as the most powerful method available for certain classes of systems [6].

2.1 Optimization Problem Formulation

While the NMPC problem formulation is driven by the specification of the control objective, constraints and dynamic model formulations, one should also consider potential numerical challenges at this point. In particular, important characteristics of the trade-off between numerical accuracy and computational complexity are determined already at the point when the NMPC optimization problem is formulation through discretization, choice of parameterizations, and choice of decision variables and constraint formulations in the optimization problem.

Let consider the MIMO non-linear system

$$\dot{x} = f(x, u)$$
$$y = h(x, u)$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^{n_u}$ is the command input and $y \in \mathbb{R}^{n_y}$ the output of the system. According to the main assumption, all the states are measured with a fixed sampling time T_s and their values are provided with the same sampling to the NMPC algorithm to obtain a prediction of the plant. The control law at each time computes a prediction of the states and the outputs over an interval $[t, t + T_p]$ where T_p represent the *prediction horizon*. Hence, the NMPC evaluate the predicted output of the system according to the initial state of that prediction interval $x(t)$, and the input for a specific time instant of that interval τ.

$$\hat{y}(\tau) = \hat{y}(\tau, x(t), u(\tau)), with \tau \in [t, t + T_p]$$

To simplify the calculation it can be assumed that the input changes only for a fraction of the prediction horizon, also known as the *control horizon* T_c, after which the input value remains constant. Both T_p and T_c are integer multiples of the sampling time T_s. Hence, $u(\tau)$ is an open loop input, since it doesn't depend anymore on the initial state.

The aim of the controller is to generate at each sampling time a prediction-based input $u*(\tau)$ that minimizes the objective function, or cost function $J(u(t : t + T_p))$. The specification of the NMPC control functionality and dynamic performance is essentially provided through the cost function and the constraints. For this algorithm the cost function is based on l_2 weighted norms of the kind

$$\| v \|_Q^2 = v^T Q v = \sum_{i=1}^{n} q_i v_i^2, Q = diag(q_1, ..., q_n) \in \mathbb{R}^{n \times n}, q_i \geq 0$$

Indeed the objective function used for this algorithm, comes from the literature [7] and is the following

$$J(u(t : t + T_\mathrm{p})) \doteq \int_t^t + T_\mathrm{p}(\| \tilde{y_p}(\tau) \|_Q^2 + (\tau) \|_R^2) d\tau + \| \tilde{y_p}(t + T_\mathrm{p}) \|_P^2$$

where

$$\tilde{y_p}(\tau) \doteq r(\tau) - \hat{y}(\tau) r(\tau) \in \mathbb{R}^{n_y}$$

The optimization problem is to minimize at each sample time the tracking error square weighted norm $\| \tilde{y}(\tau) \|_Q^2$, over a finite time interval. The other two contributes in the cost function $\| \tilde{y_p}(t+) \|_P^2$ $\| u(\tau) \|_R^2$ respectively take into account the final tracking error and how the algorithm manages the trade-off between performances and command activity. The minimization is subject by definition to the constraints

$$\begin{aligned} \dot{\hat{x}}(\tau) &= f(\hat{x}(\tau), u(\tau)), \\ \hat{x}(t) &= x(t), \\ \hat{y}(\tau) &= h(\hat{x}(\tau), u(\tau)). \end{aligned}$$

while other constraint may be used to describe particular conditions depending on the system to control

$$\begin{aligned} &\hat{x}(\tau) \in X_c, \\ &\hat{y}(\tau) \in Y_c, \tau \in [t, t + T_p] \\ &u(\tau) \in U_c, \tau \in [t, t + T_c] \end{aligned}$$

2.2 Mathematical Formulation and Control Algorithm

The optimization problem described in the previous section as:

$$\begin{aligned} u^*(t : t{+}T_p) &= \arg\min_{u(\cdot)} J(u(t : t + T_p)) \\ &s.t. \\ &\dot{\hat{x}}(\tau) = f(\hat{x}(\tau), u(\tau)), \hat{x}(t) = x(t) \\ &\hat{y}(\tau) = h(\hat{x}(\tau), u(\tau)) \\ &\hat{x}(\tau) \in X_c, \hat{y}(\tau) \in Y_c, u(\tau) \in U_c \\ &u(\tau) = u(t + T_c), \tau \in [t + T_c, t + T_p] \end{aligned}$$

can't be solved easly, since its formulation in general is non-convex and with an infinite number of decision variables. Indeed, the cost function is a function of the input, which is in turn a function of time.

The prediction time interval in which is defined the problem, is a continuous interval of time instant. In order to reformulate the problem into a finite-dimensional and practical setting, the following assumptions allow the integral and differentiation operators to be approximated by numerical integration methods.

The horizon T_p is finite and given, the input signal $u(\tau)$ is assumed to be piece-wise constant with a regular sampling interval T_s such that T_p is an integer multiple of T_s, and parameterized by a vector such that $u(\tau) = \sum_{i=1}^{m} c_i \phi_i(\tau) = c\phi(\tau)$ is piece-wise continuous. Hence, the common parametrization can be obtained through the rectangular functions

$$\phi_i(\tau) = \begin{cases} 1, & \tau \in [t + (i-1)T_s, t + iT_s] \\ 0, & \text{otherwise.} \end{cases}$$

or the polynomial functions

$$\phi_i(\tau) = (\tau - t)^{i-1}$$

The result is an optimization problem like:

$$\begin{aligned} c^* &= \arg \min_{c \in \mathbb{R}^{n_u \times m}} J(c) \\ &s.t. \\ \dot{\hat{x}}(\tau) &= f(\hat{x}(\tau), u(\tau)), \hat{x}(t) = x(t) \\ \hat{y}(\tau) &= h(\hat{x}(\tau), u(\tau)) \\ u(\tau) &= c\phi(\tau) \\ \hat{x}(\tau) &\in X_c, \hat{y}(\tau) \in Y_c, u(\tau) \in U_c \\ u(\tau) &= u(t + T_c), \tau \in [t + T_c, t + T_p] \end{aligned}$$

where the optimal parametrized input is

$$u^*(\tau) = c^* \phi(\tau)$$

Since the optimal solution to the minimization problem is an open-loop value that depends only on the initial state, the optimal input $u(\tau)$ do not perform any feedback action to increase the precision of the system. To overcome this open-loop behaviour and obtain a feedback control algorithm, at each sampling time is computed an optimal $u^*(\cdot)$ and keep constant for that time interval. NMPC is based on the receding horizon control principle, where a finite horizon open loop optimal control problem solved at each sampling instant and the optimized control trajectory is implemented until a new optimized control trajectory is available at the next sampling instant.

2.3 Parameter Design

Hence, the NMPC produces at each sampling time the solution of the optimization problem, as the optimal input that sends to the plant and keeps constant in that time sample. The resulting algorithm can be transformed in a MATLAB function to be inserted in the Simulink block scheme as shown in Figure 13, and uses a model developed by the user to compute the prediction and solve optimization.

Clearly the complexity of the model inside the controller affects its capability to reach the solution, according to the amount of calculation that have to solve in each sample time. There are also some

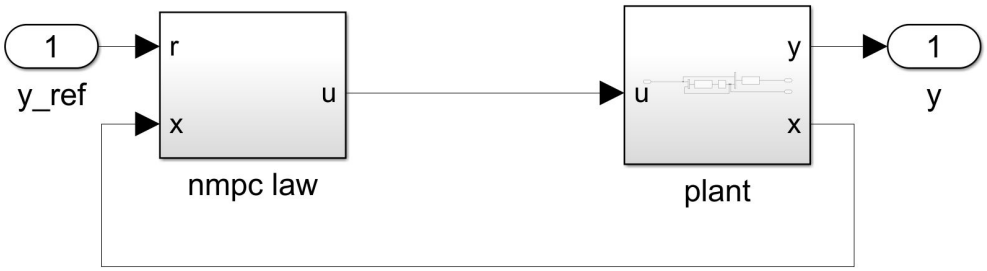


Fig. 13: Simulink® general block scheme for plant and Controller

other important parameter that influence the controller performances, such as the sampling time T_s, the prediction horizon T_p and the control horizon T_c.

In many situation the sampling time T_s depends on the hardware that hosts the controller, it means that is a fixed constraint. In some particular cases the choice is available, hence, the values for T_s should be sufficiently small to deal with the plant dynamics according to the Nyquist-Shannon sampling Theorem, but at the same time a very small value leads to numerical problems and a slow computation.

T_p and T_c indeed, can be chosen with a trial and error procedure. Usually the control horizon in two limit cases can be assumed equal to the prediction horizon or to the sampling time, while the prediction horizon should be large enough to guarantee closed loop stability properties, but larger value may reduce the accuracy on the short time interval.

It is also important to chose properly the weight matrices used inside the cost function, that are Q,R and P. Those are diagonal square matrices with dimensions $Q \in \mathbb{R}^{n_u \times n_u}$,$P \in \mathbb{R}^{n_y \times n_y}$ and $R \in \mathbb{R}^{n_y \times n_y}$, and their regulation depends on a trial and error tuning due to several simulation. Although the initial values of the diagonal elements can be chosen according to the requirements

$$Q_{ii} = \begin{cases} 1 \text{ in the presence of requirements on } x_i \\ 0 \text{ otherwise} \end{cases}$$

$$P_{ii} = \begin{cases} 1 \text{ in the presence of requirements on } y_i \\ 0 \text{ otherwise} \end{cases}$$

$$R_{ii} = \begin{cases} 1 \text{ in the presence of requirements on } u_i \\ 0 \text{ otherwise} \end{cases}$$

Later, depending on the simulation result the values on the diagonal can be modified with a trial and error procedure, using the Table 4 as a guideline.

Increasing Q_{ii},P_{ii}	$\Rightarrow$	decreasing the energy of x_i, y_i	$\Rightarrow$	reducing oscillations and converging time
Increasing R_{ii}	$\Rightarrow$	decreasing the energy of u_i	$\Rightarrow$	reducing command effort and "fuel consumption"

Table 4: Trial and error tuning hints

2.4 Robustness and stability

The NMPC control algorithm described above, can ensure the system stability, thanks to several principles refearring to literature such as adopting a parametrization sufficiently rich. It is also useful to increase the prediction horizon, since, as explained before, with T_p large enough the closed loop stability increases. Moreover, it can be used a complex cost function that approximate an infinite horizon, but introduces a very complex computational problem.

Using terminal set constraints of the type $x(t_N) \in \Omega$, that ensures that the state is regulated in order to be "close enough" to the set-point such that after T_p it is a priori known that there exists a feasible and stabilizing controller that will ensure that $x(t)$, never leaves Ω and eventually goes asymptotically to the set-point. In the end, using terminal equality constraints of the type $x(t_N) = r$, that ensures convergence in finite time, implies that the cost after time T_p is zero, and is therefore related to both an infinite-cost strategy and a stability-preserving-constraint strategy.

In real-world applications, the exact plant model is seldom known. This means that an approximated model $\hat{f}, \hat{h}$ is used for control design, instead of the "true" model f, h (this holds for any method).Practical industrial experience shows that MPC tend to be inherently robust, even without any particular consideration in the design phase beyond ensuring the accuracy of dynamic models and formulating realistic specifications in terms of operational constraints and cost function weights. A necessary condition for lack of robustness is that the value function and state feedback law are discontinuous.

There exist a wide range of NMPC formulation that include robustness into the formulation of the optimization problem. One can mainly distinguish between several robust approach, such as the min-max NMPC, the H_∞ NMPC and the parametrized controller. All those techniques typically require a high computational effort and thus cannot be applied to problems where a small T_s is required. Thanks to the receding horizon strategy, standard NMPC is in general characterized by good robustness properties.

3 NMPC Set-up For The Application Case

According to the working principle of the NMPC and the formulation of the optimization problem, to maintain high level performances, such as stability and robustness,the following pages discuss about the design of the plant model and the NMPC parameters, in order to obtain the guidelines to actuate this procedure.

The objective of the NMPC is to control the application of Autonomous driving systems dealing with the vehicle states and inputs. Summarizing, the prediction model described in Chapter 1.3.1 uses

$$x = \begin{bmatrix} X \\ Y \\ \psi \\ v \end{bmatrix}, u = \begin{bmatrix} a_x \\ \delta_f \end{bmatrix}$$

which are respectively the states and the inputs, with the system order equal to n_x, while the equations of the model provided to the controller are

$$\begin{aligned} \dot{X} &= v\cos(\psi + \beta) \\ \dot{Y} &= v\sin(\psi + \beta) \\ \dot{\psi} &= \frac{v}{l_r}\sin(\beta) \\ \dot{v} &= a \\ \beta &= \tan^{-1}\left(\frac{l_r}{l_r + l_f}\tan(\delta_f)\right) \end{aligned}$$

All the assumption about the parameters depends on that model and on the driving conditions, hence the modify of those elements may require its own review.

3.1 Sampling Time T_s, Prediction Horizon T_p and Control Horizon T_c Design

The first important parameter is the sampling time of the controller, which discretizes the continuous evolution of the system in order to represent it in a manageable code-wise fashion. This parameter usually depends on the architecture, and the complexity of the hardware that hosts the control law, which provides the computational power to optimize and solve the minimization of the cost function.
By definition this parameter can't be too small, otherwise complexity increases, but the system reaction time becomes smaller. Its value has been obtained with several repeated simulation, from a starting initial sample interval of $T_s = 0.2$ seconds,up to $T_s = 0.5$.

Those value comes from some reasonable assumed sampling interval compared with the hardware capabilities and the human reaction time. Since the worst case, with a large step interval, still

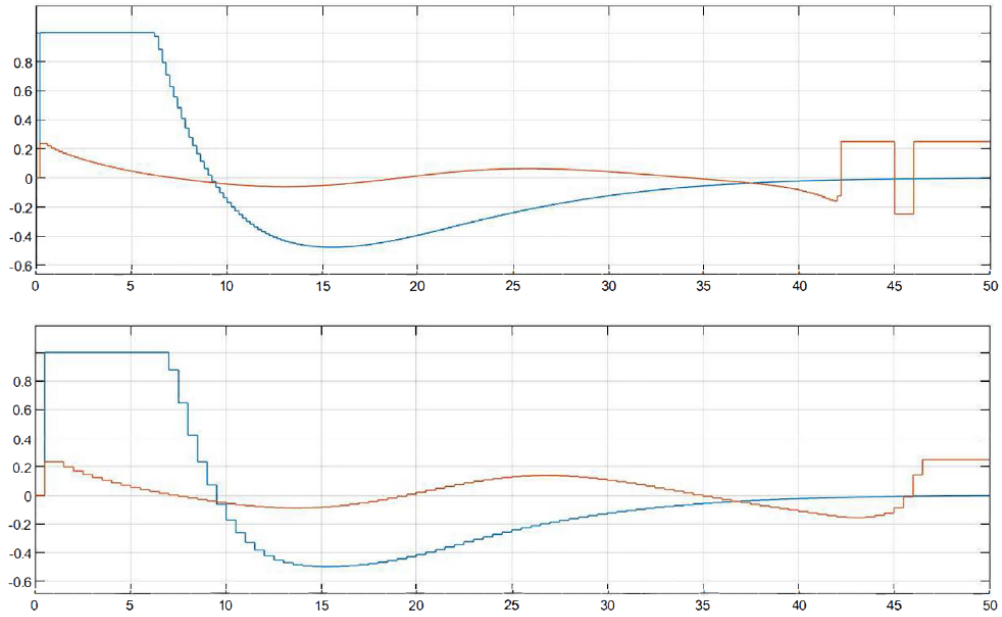


Fig. 14: Stepped Inputs with $T_s = 0.2$ (top) and $T_s = 0.5$ (bottom)

provides robust performances the final value will be $T_s = 0.5s$. According to the input parametrization, the generated values remains constant for the entire sampling interval, indeed the the input is described with m polynomial functions

$$\phi(\tau) = [\phi_1(\tau), \ldots, \phi_i(\tau), \ldots, \phi_m(\tau)]^T \in \mathbb{R}^{m \times 1}$$

with

$$\phi_i(\tau) = (\tau - t)^{i-1} = (\tau - t)^0 = 1, m = 1$$

hence,

$$u(\tau) = c_1 \phi_1(\tau) = c_1 = constant$$

This particular simple parametrization with $m = 1$ needs the assumption of $T_c = T_p$, and leads to a stepped input signal Figure14. For the basic application this configuration is very robust and allow to fast computation, although the prediction horizon can be set up to 8 seconds Figure15, ensuring the system to reach the closed loop stability.

On the contrary, when the system requires a tuning that increases the performance for the short time accuracy the sampling time and consecutively the prediction horizon must decrease, maintaining at least $T_p = 8\, T_s$.

3.2 Weight Matrices Q, P, R design

In machine learning algorithms such as in the NMPC ones, it is essential to define properly the objective function or rather the function to minimize, in order to reach with the desired behaviour

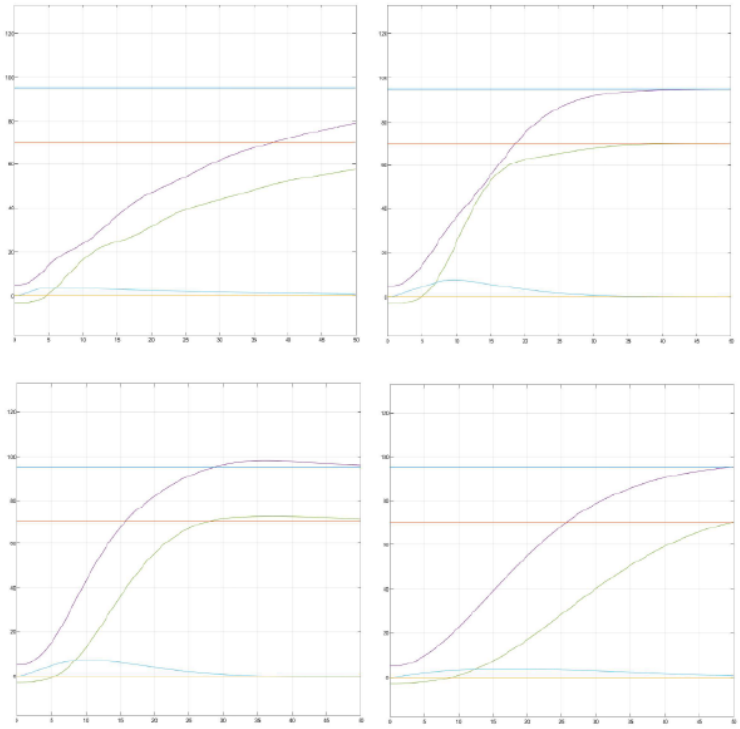


Fig. 15: Differences in the output with an increasing T_p

the references. This can be managed acting on the weight matrices inside the cost function. The three matrices are Q, P and R and operates respectively on the states outputs and inputs.

For this specific application case, the model outputs refer directly on the states, this means that weights on the states could be redundant, hence $Q = 0$. For the other matrices the tuning is rather complex and various, depending on the specific case of study, nevertheless it could be identified the main approach based on the simulation results.

The P matrix weights the outputs (at most four), that are the coordinates of the vehicle in the inertial frame, the yaw angle and the longitudinal velocity of the vehicle. The elements on the diagonal that refer to the coordinates X and Y must have always the same value, since unbalancing the priority on one coordinate, the inputs of the controller generate a trajectory that oscillate around the coordinate with less priority, beside the vehicle can miss the final position.
The best solution requires that the oscillation must be null for both coordinates otherwise the vehicles tries to reach that reference rounding around it. The weight on the yaw angle depends on the application case and refer to the orientation of the vehicle with respect to the inertial frame. However, an oscillation around that output could be accepted, than the weight may be smaller for it. Indeed, the velocity accepts oscillation around the reference value only for driving applications, since there is an higher tolerance, while reaching a fixed position stopped, an oscillation around the zero velocity means alternate forward drive and reverse one.

In conclusion the elements on the R matrix diagonal reduce the command effort respectively on

the acceleration/braking and on the steering. Hence, for the application cases studied, the weight on the acceleration/braking input is not relevant, since the actuator dynamic self-controls the inputs generated by the controller. While for the steering command usually a big variation is not acceptable, because an high lateral dynamic makes the vehicle unstable and unable to maintain the trajectory. This assumption is stronger dealing with high velocities, due to a significative contribution on the lateral dynamic.

The following tables summarizes these aspects in a nutshell.

Q	X	Tuned with the same weight of Y, must be large enough to have asymptotic convergence on the position.
	Y	Tuned with the same weight of X, must be large enough to have asymptotic convergence on the position.
	ψ	Depends on the application case, path reference requires hard weight, while without precise path low or zero weight.
	v	Usually requires an hard weight when the reference is zero no oscillation, while can be smaller whit non-zero reference.
R	a	The value is not relevant since the acceleration dynamic is regulated by the actuator.
	δ	Weight should be large enough to avoid big variation on it.

Table 5: Weights Tuning in a Nutshell

3.3 Constraint design

There are other important instruments that can be adopted to provide additional information in the controller tuning, that unquestionably depends on the application case. Those are the constraints, which, dealing with autonomous driving tasks, can be included in three main classes: the ones that bounds the trajectory such as lane keeping, the ones that describes the obstacles around the vehicle and, as the last step, the ones that refer to the limits of the vehicle itself. Obviously, some additional constraints could be adopted depending on the task to accomplish.

The constraints on the vehicle limits are usually common for all the studied cases inasmuch refers to the input of the vehicle, indeed they are:

$$a_x \in [-1.5, 1]\ m/s^2$$
$$\delta_f \in [-0.25, 0.25]\ rad$$

It can be also included in this constraint class an additional one referred on the velocity limits, that for a forward drive application are

$$\dot{x} \in [0, v_{max}]\ m/s$$

where v_{max} depends on the traffic rules.

The additional constraints as well as the trajectory and the obstacle ones, must be provided to the controller in the inequality formulation in the simulation and for the specific algorithm used. In a general application the trajectory constraints depends on the path planner and on the information captured by sensors, while obstacle detection depends only on the sensor outputs.

For the driving applications, during simulations, it has been assumed that both obstacles and path are given, so the respective information are already known and included as inequalities inside the control algorithm and will be described in the following chapter.

4 Test Design Simulation

To test and verify the performances of the control system, approaching it with the NMPC, it is necessary to design some scenarios which ask for a specific behaviour to analyze. Two main kind of request has been identified to describe some of the autonomous driving tasks, such as the fixed objective reaching and the trajectory keeping.

The vehicle control algorithms usually operate on two levels, that must work simultaneously to achieve some complex tasks, indeed one aim of this study consist in reduce as much as possible the use of multiple controls exploiting the capability of the NMPC algorithm.

Analyzing it in detail, the lower level provides the static set of references that the system must reach pointing the objective in a local subspace, hence the controller restricts to produce the command efforts for the vehicle to achieve that objective. While, operating in an higher level control, it is necessary to make the system aware about the reference variation along the time, sketching the intermediate targets that allows to reach the global one. This kind of high level control is known as path planner, which defines the reference trajectory points as a function of the states usually based on the position of the vehicle.

However the NMPC is able to 'decide' on its own the path to follow according to the minimization algorithm, that evaluate the best trajectory built on the road constraints only. As a matter of fact, the two main scenarios defined can find their solution through the two control law described, but with some differences in between, since the simplest one, the fixed objective reaching, doesn't require necessarily a path planning algorithm. While, dealing with trajectory keeping requests the path planner could be the obligatory approach, since there is no final reference to reach, but the path itself is the reference for the system.

The first two tests designed represent the scenarios described above in a specific configuration, one is the fixed objective test with some obstacles in the space that the vehicle must avoid thanks to the inputs generated by the NMPC, while the other requires to follow a sinusoidal trajectory without using a path planning algorithm.

In the end the last scenario mixes some elements from the previous cases, to simulate the control of the vehicle during an emergency maneuver, evaluating the possibility to adopt that algorithm as the basis to design an ADAS for real vehicles. This thesis centers the attention on the algorithm used for the control of the vehicle, which means that none of the reference used in the tasks adopt a path planning algorithm, hence the references are already known by the system itself, but the information is given to the controller using several constraints on the system sates. This choice is also meaningful to test the algorithm strength operating using only the low level control.

The Pages below describes in details the design and the result obtained by each simulation.

4.1 Point To Point Objective And Obstacle Managing

The first Autonomous driving request evaluate the capability of the control law to avoid some obstacles driving the vehicle from a point A to another B, as shown in Figure 16. To obtain that goal, designing the complete simulation with all the tasks were used intermediate steps and different kind of obstacles to compare the response of the control law. The simplest implementation for that kind of tasks is the one without any obstacle around, that can be used as a guideline designing all the sub-cases.

These can be defined 'point to point' driving tasks, since the vehicle starts with null velocity and must reach the target in standstill condition. For this first request the goal can be achieved without any particular attention to the orientation of the vehicle in the space. Designing the simulations it is necessary to consider the Kinematic model, since it is the one used to predict the vehicle states.

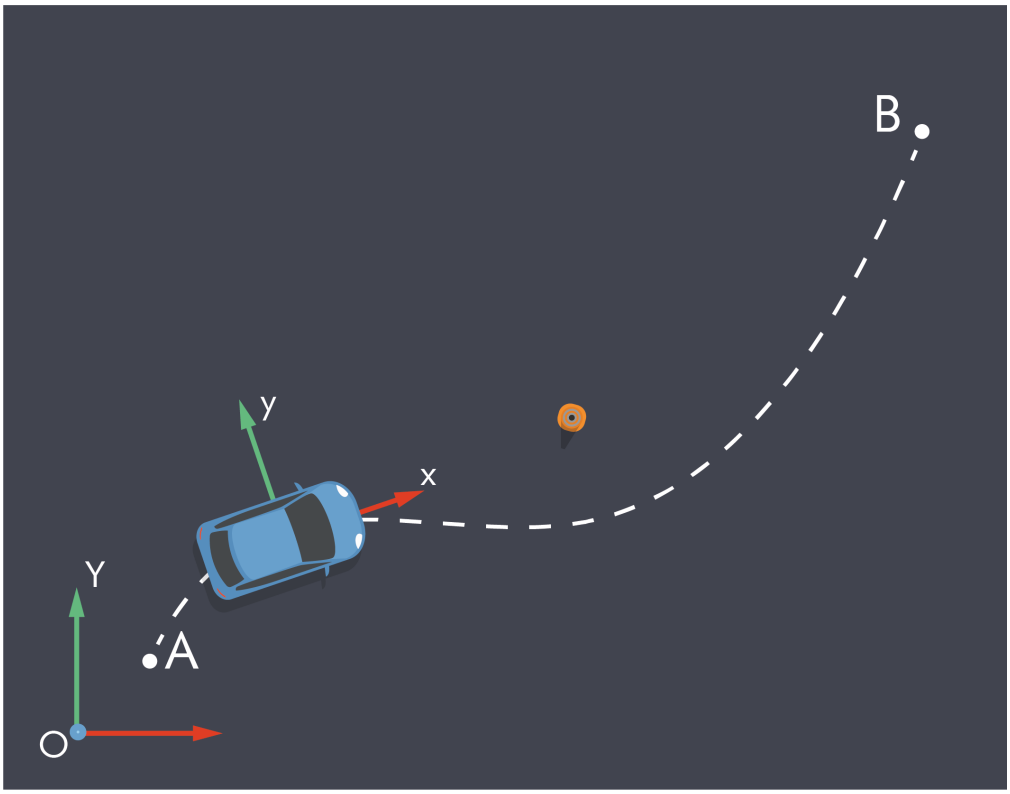


Fig. 16: Point to point case

Hence, the model the states and the output used are the following:

$$x = \begin{bmatrix} X \\ Y \\ \psi \\ v \end{bmatrix}, \qquad y = \begin{bmatrix} X \\ Y \\ v \end{bmatrix}$$

As explained in the previous chapters, the output equations are different for each task to be accomplished, since they define implicitly which kind of reference must be used to control the vehicle. In that case the first two outputs define the position of the of the CoG in $\mathbb{R}^2$, while the third is longitudinal vehicle speed. It means that the reference required are two, the position of the target B, and the null final speed.

$$\begin{aligned} X_{ref} &= 95\ m \\ Y_{ref} &= 70\ m \\ v_{ref} &= 0\ m/s \end{aligned}$$

It is also necessary to realize other hypothesis, such as the starting conditions, which are related to the dynamic vehicle model and are respectively the two velocities along the x and y axes of the body frame, the position of the CoG, the yaw rate and the yaw angle:

$$\begin{aligned} \dot{x_0} &= 0\ m/s \\ \dot{y_0} &= 0\ m/s \\ X_0 &= 5\ m \\ Y_0 &= -3\ m \\ \dot{\psi}_0 &= 0\ rad/s \\ \psi_0 &= 0.15\ rad \end{aligned}$$

The expected behaviour of this controlled system, according to the minimization algorithm used by the NMPC, will be the shorter line between the two points in absence obstacles on this trajectory. For that reason, to design that simulation, the initial yaw angle has been chosen to orientate the car in a different direction with respect to the target, this allow to formulate the problem with general initial condition.

Moreover, the hypothesis places the car on a plane without lanes that obligate its trajectory,since there's no interest in testing this capability now, than, without any obstacles around, the control law doesn't require any constraint.

The simulation time is reasonably large to observe the convergence of the outputs $T_{sim} = 50s$ to their reference value, while the controller's sampling time chosen is $T_s = 0.5s$, since the reaction time required is not too large. This sampling allows also to adopt a prediction horizon $T_p = 4s$ enough to converge asymptotically with the objective coordinates and velocity as in Figure 17, which corresponding trajectory plot is the one in Figure 18. As the last step it is necessary to configure the weight of each output and input variable, they are:

$$R = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

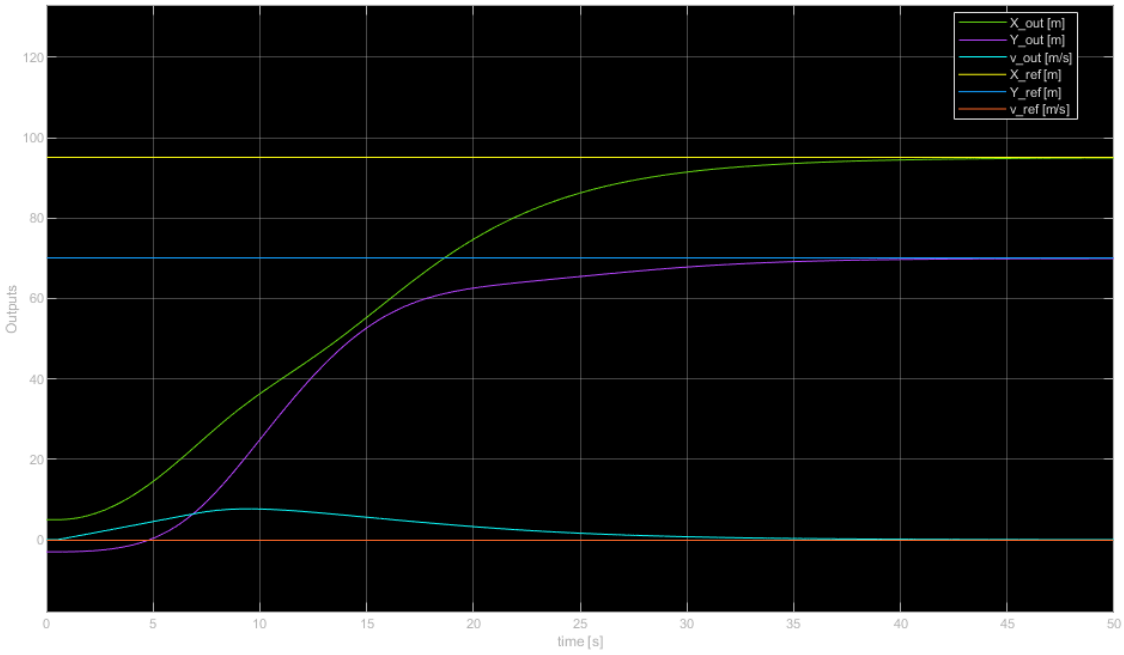


Fig. 17: Output time plot without obstacle

This choice comes from several simulations with the purpose to determine the response to the tuning, as explained in the previous chapter to illustrate the meaning of those values. The input weights are null for both the signal, since the task do not require to prevent the variation of those efforts. This can be related to the simplicity of the study case, otherwise in another context it can be useful to increment the weight on the steering command to avoid unusual steering to round the obstacles, inasmuch the high lateral dynamic of the vehicle can lead to instability of the system.

Observing the inputs generated by the NMPC time by time, those are reasonable enough and coherent with the expected trend, indeed without any weight on steering angle the convergence to the final position can be reached, that is the best compromise for the trajectory. To reduce the wide turns performed by the vehicle the weight matrix on the inputs parameters is

$$R = \begin{bmatrix} 0 & 0 \\ 0 & 100 \end{bmatrix}$$

As shown in Figure 20, the new path is more straight, but it is necessary accept a bigger error on the final position with the reference.

To increase the complexity of the simulation, some obstacles can be introduced assuming for them different dimensions and that are in standstill or moving condition. The NMPC algorithm provided manages the constraints with inequalities, so the simplest way to describe obstacles is to use the

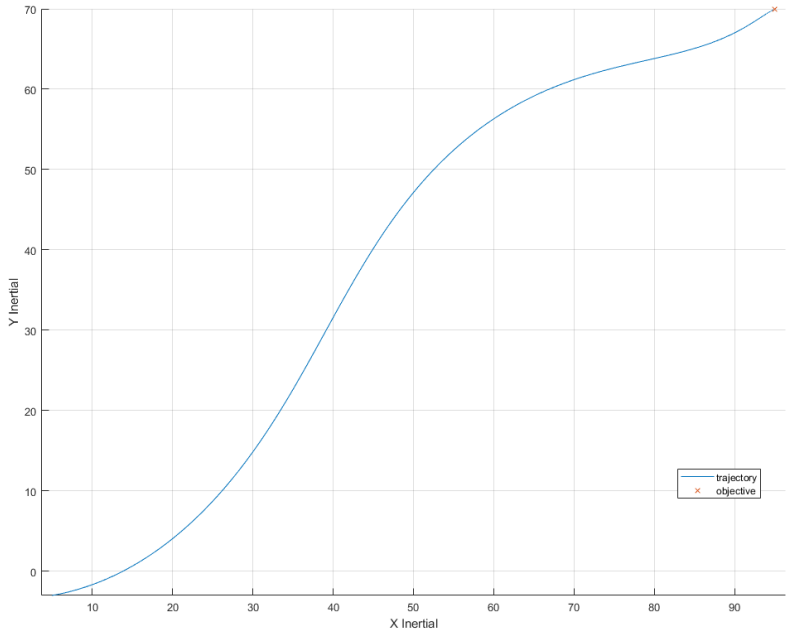


Fig. 18: Trajectory plot in the inertial RF without obstacle

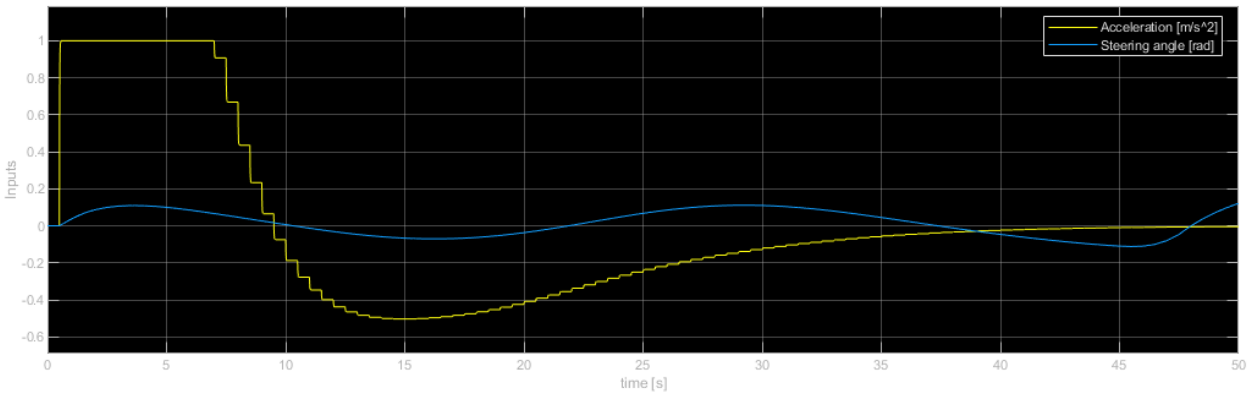


Fig. 19: Input time plot without obstacle

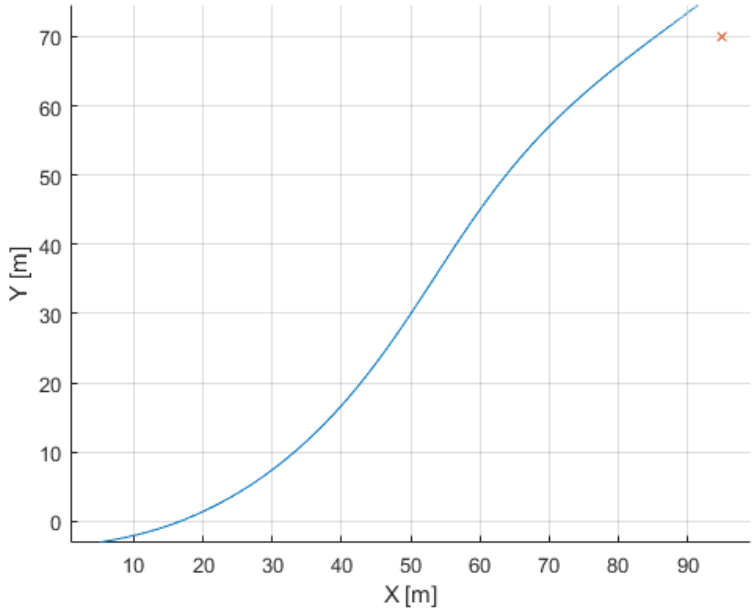


Fig. 20: Trajectory plot in the Inertial RF without obstacle and high steering weight

implicit equation of a circle

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

where the center position $C = (x_c, y_c)$ can be fixed or with a constant speed, assuming stationary motion, and the radius depends on the kind of obstacles. The simplest obstacles considered in the simulation are three as the ones in Figure 21, that are, from the smaller one to the larger one, a traffic cone, a pedestrian and a bicycle.

To force the system reaction to the presence of those obstacle, the center of the associated circles has been placed on the trajectory obtained from the previous simulation. Since the state equations concentrate the whole vehicle dimension in it's CoG it is necessary to increase the radius of each obstacle with a fixed value computed using the distance of the more distant corner of the vehicle from the CoG, and it is

$$r_{vehicle} = \sqrt{l^2 + w^2} \qquad \text{where} \begin{cases} l = 3,6m \\ w = 0.8m \end{cases}$$

than the final radius value has been computed as

$$r = (r_{obstacle} + r_{vehicle}) * 1.1$$

increasing the sum with an additional safety factor of the 10%, the result is more robust. Moreover all those cases uses the same Code 4 for the obstacle representation and the same weight's matrices:

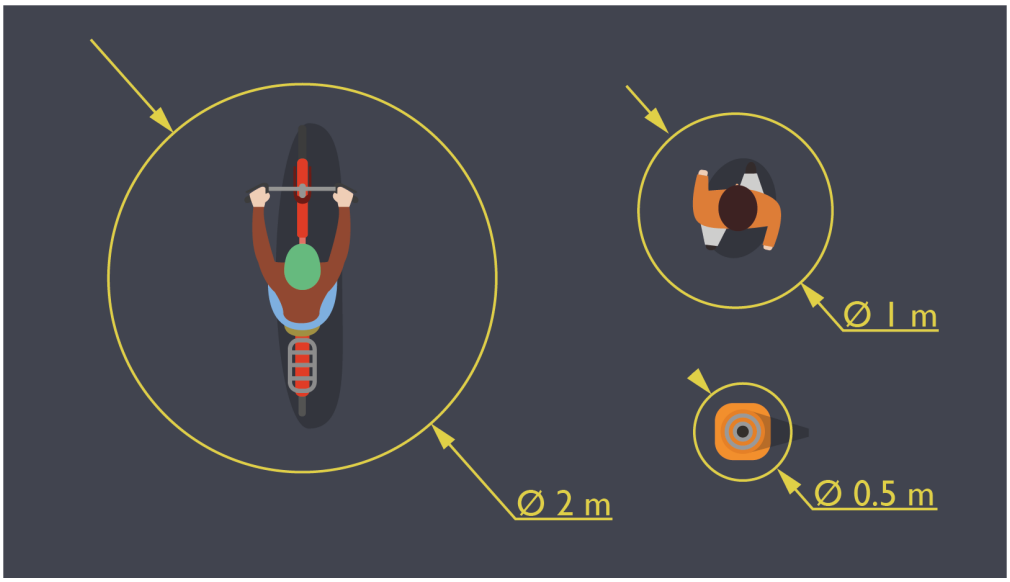


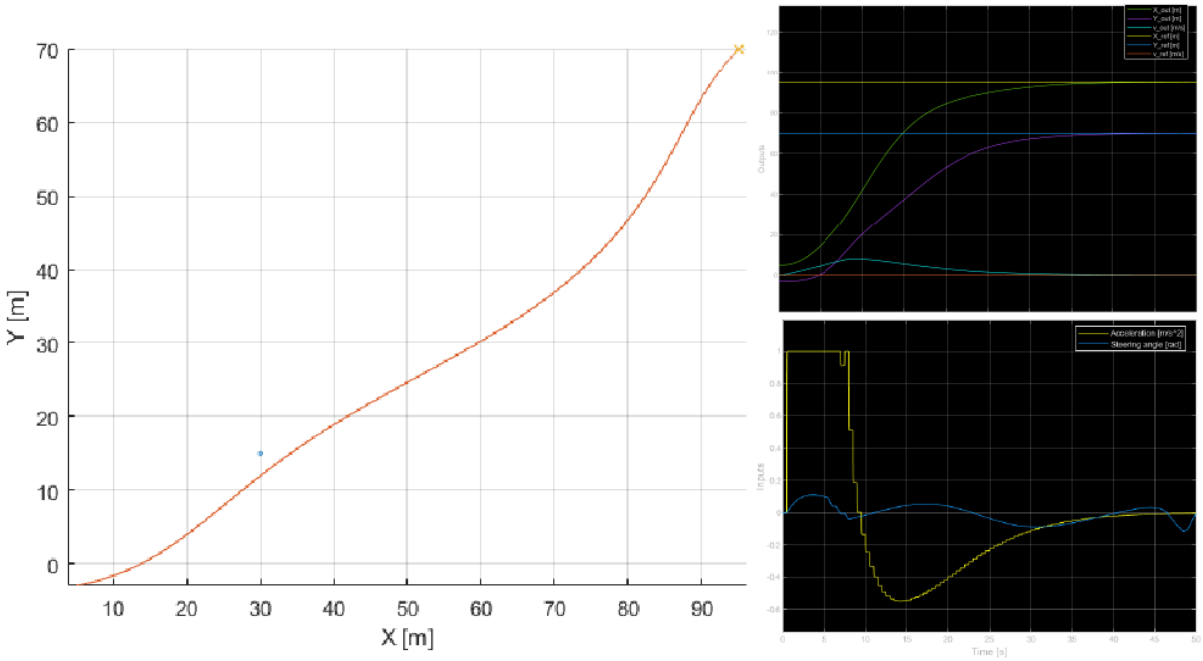
Fig. 21: Obstacles' dimension and space area

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

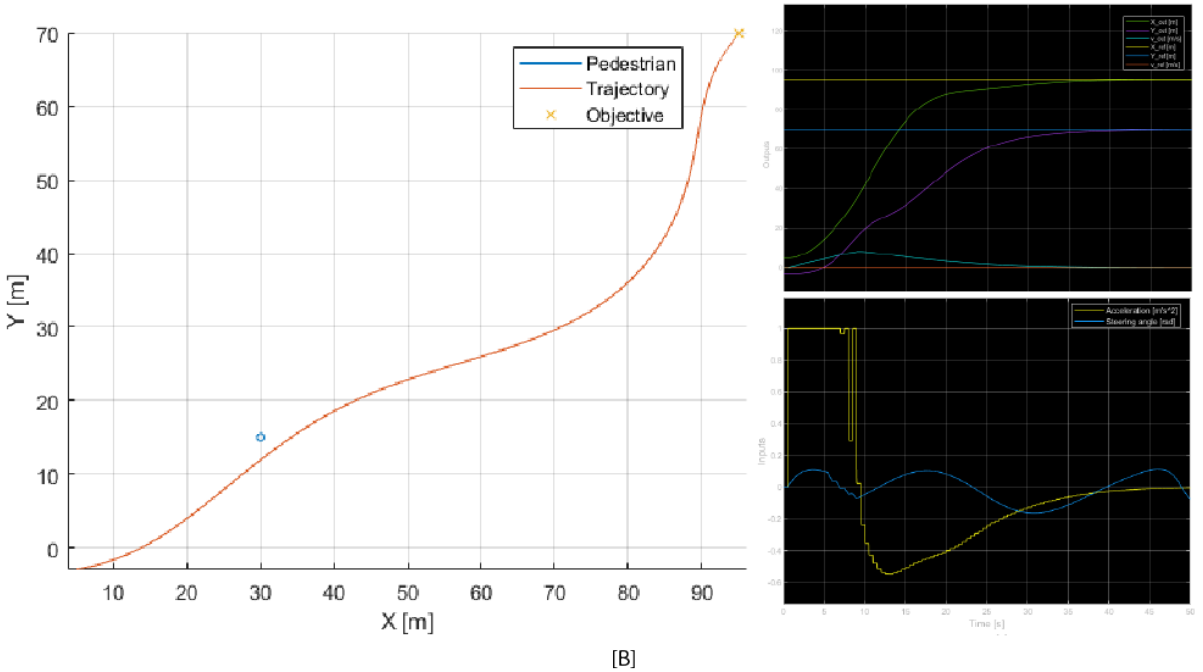
```
function F=constraint(x,y)
%Obstacles radii set
bk=1; %bicycle
pd=.5; %pedestrian
tc=.25; %traffic cone

%Obstacle augmented radius
r=(tc+1.8)*1.1;

%Obstacle center location
xc=30;
yc=15;

%vehicle speed
v=y(3,:);

% Constraints are written in the standard form F(x,y)<=0;
F=[r-sqrt((y(1,:)-xc).^2 +(y(2,:)-yc).^2),-v,v-15];
end
```

Code 4: Fixed obstacles constraint

Hence the results are represented in the plots in Figure 22


Y [m]
X [m]
Outputs
Inputs
Time [s]

[A]


Pedestrian
Trajectory
Objective
Y [m]
X [m]
Outputs
Inputs
Time [s]

[B]

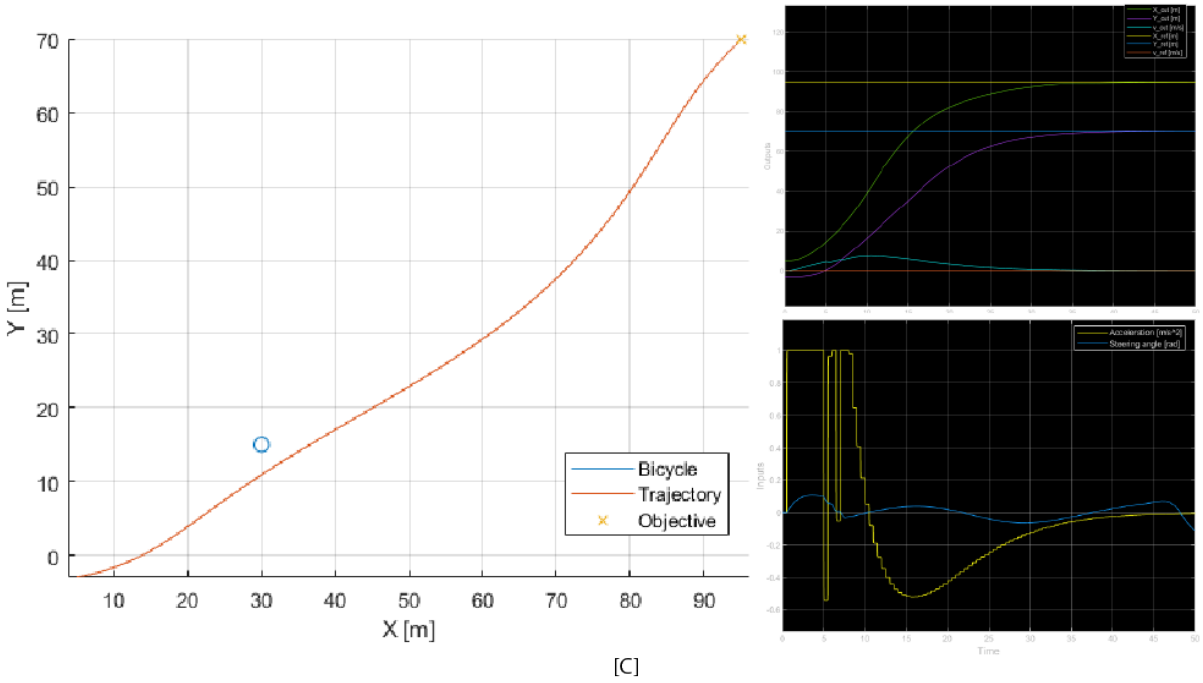


Fig. 22: Fixed Obstacles Plots: [A] Traffic Cone [B]Pedestrian [C]Bicycle

To test further the robustness of the control law, another configuration with more obstacles can be used, imagining multiple traffic cones on the road to signal a pothole, Figure 23, that still uses the same weight configuration of the previous cases.

The results show that the NMPC can accomplish the autonomous obstacle avoid task and in some cases actually, the trajectory of the vehicle is better than the one computed without constraints. The explanation for this behaviour is simple, since the first case without obstacles has more degrees of freedom, hence the minimization algorithm can be easily satisfied. Moreover that task doesn't require a strong optimization, determined by the exact tuning of the controller parameter, but is determined to demonstrate the feasibility of the point to point autonomous driving.

The last but not least assumption to be done verifies the capability of the algorithm to avoid moving obstacles during the point to point driving task. Unfortunately the NMPC algorithm is involved in the simulation environment trough a function which takes the information from a .struct file. This includes also the constraints that must be defined before the the simulation starts. It follows that the obstacle's motion is already known and can't be supplied during the simulation as the result of information elaborated by sensors. The assumption necessary to proceed bases on the capability of the vehicle to predict, with the sensors, the position of obstacles in the time domain. The final result is a trajectory that must avoid any collision. To design this simulation the constraint function takes the time as a vector of time samples (with the same sampling time of the NMPC) for all the duration of the simulation. Than, to obtain an effective response of the controller to avoid the obstacles, it is necessary to design the obstacle velocity and initial position according to the trajectory computed for unconstrained NMPC. In the end the equation to describe the movement

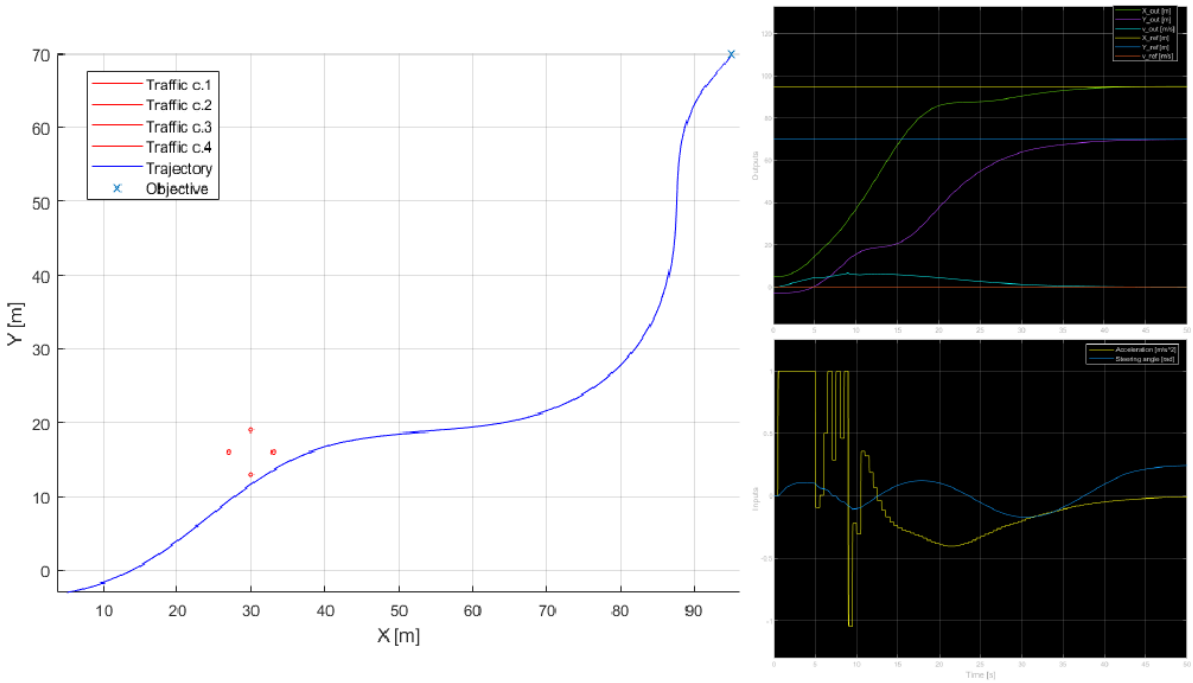


Fig. 23: Multiple Obstacles

of the obstacle center has been computed assuming it in stationary motion. It can be summarized in the following code.

```
xc1=27;
yc1=16;

xc2=30;
yc2=13;

xc3=33;
yc3=16;

xc4=30;
yc4=19;

F=[r-sqrt((y(1,:)-xc1).^2 +(y(2,:)-yc1).^2),r-sqrt((y(1,:)-xc2).^2 +(y(2,:)-yc2)
    .^2),r-sqrt((y(1,:)-xc3).^2 +(y(2,:)-yc3).^2),r-sqrt((y(1,:)-xc4).^2 +(y(2,:)-
    yc4).^2),-v,v-15];
```

Code 5: Constraint for moving obstacles

The algorithm is able to avoid those obstacles still using the same configuration with the weight matrices, moreover, since the algorithm knows a priori the trajectory of the obstacle, the result is

not sufficient to completely evaluate the reactions in this case.

Instead, the overall result for the point to point autonomous driving task, with or without obstacles, satisfies the request, since demonstrate the feasibility of the problem and the capability to reach a target avoiding obstacles. The introduction of constraints doesn't modify or make it hard to reach the target, on the contrary the constraint can be properly used to improve the performances of the control system, since delivers indirectly the information to approach the solution.

To improve the task performances is also necessary to use detailed models with the state equation to describe the lateral dynamic of the vehicle. That allows higher speed and therefore to reduce the convergence time.

In the end, using obstacles with higher dimensions the vehicle can't reach the target, but stops before the target itself. This behaviour can be justified, since to prevent the collision the controller identifies the unfeasibility and stops the vehicle, coherently with equivalent human decision. However, it is necessary study this behaviour separately and observe how the system reacts with the time evolution of those unfeasible constraints.

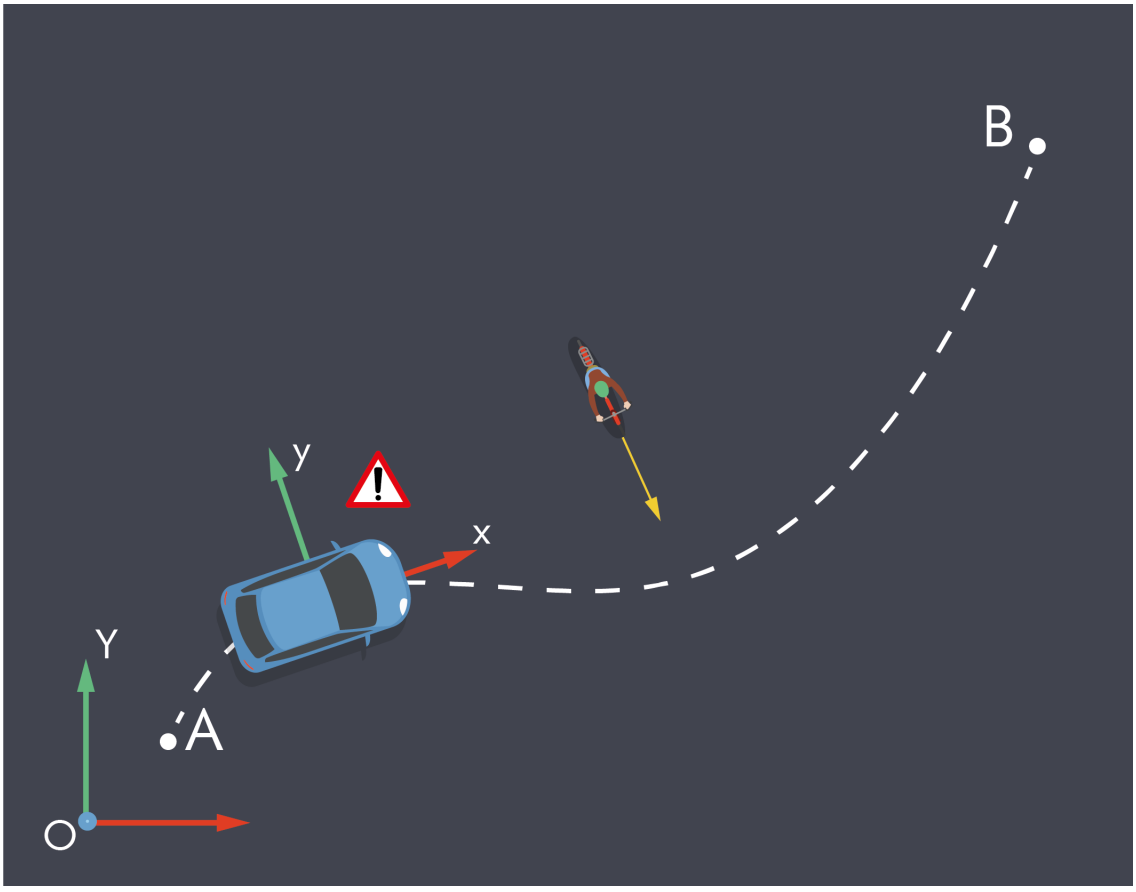


Fig. 24: Moving Obstacles scenario

4.2 Sinusoidal Lane Keeping

The objective of this simulation is to test the capability of the vehicle to proceed autonomously along the road without any specific target, using only the lane constraints. Indeed, the reference provided must only transmit the information to the vehicle necessary to proceed forward trough the path.

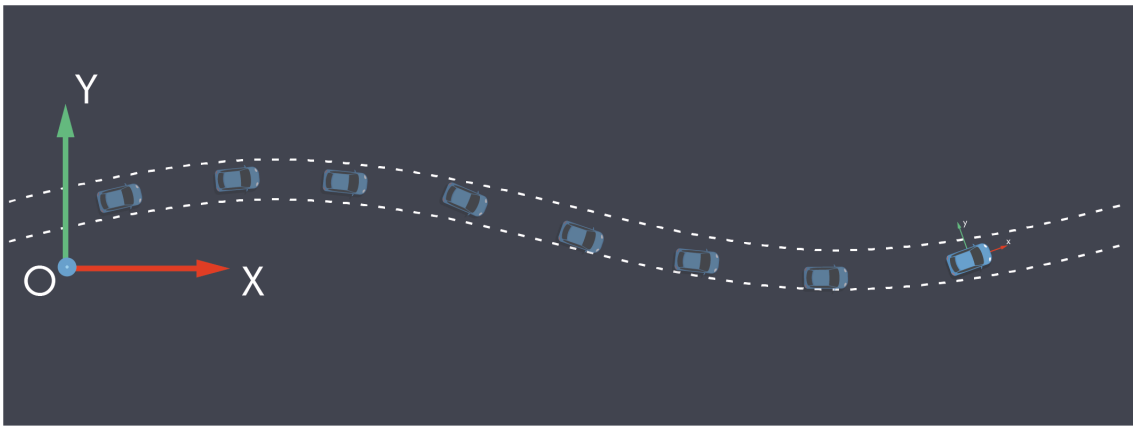


Fig. 25: Sinusoidal Lane Keeping

To provide the lane model at each sample time, designing the simulation, the path must be known, since it is necessary translate those information in the equivalent mathematical model.
This assumption is valid only for this specific simulation and represent its limits as for the previous case, linked to the constraint managing. Nevertheless, with respect to the previous simulation scenario, the constraints aren't the complete predictions of the path lane, but only the equivalent lane position on the vehicle sides.

The choice of a sinusoidal trajectory comes from the literature [4], which simulation's results study in depth the accuracy and the robustness using a kinematic prediction model. The result of that tests highlights the limits of the kinematic model, that can't predict the behaviour in lateral dynamics, hence high velocity reduces significantly the accuracy.
The equation for the sinusoidal trajectory is:

$$Y = A\sin(2\pi\frac{X}{\lambda})$$

The sinusoidal path has an amplitude of 4 meters and a wavelength of 100 meters, in order to obtain large turns, enough to follow the trajectory with moderate velocities.

To configure the simulation, the output equations can't refer to the exact position of the vehicle in the inertial RF. To manage variable reference position, it is essential to use a path planning algorithm, but since it isn't the purpose of that autonomous driving task, the outputs must be chosen as function of the available states and inputs. For that reason the outputs could be the

vehicle longitudinal speed and the yaw angle, hence, to ensure the forward driving, the reference speed must be a positive value, and the yaw set to zero. Before starting the simulation, is still necessary to define the weights on the in/out signals and the constraints.

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

The output weight is low on the velocity and null on the yaw angle, since the system do not require actually any convergence of the yaw to a fixed reference value, but in that way it is allowed to oscillate around that value. Instead, the input weight is null on the steering angle, since the vehicle should be able to react as fast as possible to compensate the lateral dynamic, while the acceleration weight must be dumped to avoid any sudden behaviour.

To design the constraints it is necessary understand the lanes' mathematical model in space, on the contrary the real approach on vehicles is based on the sensor's data, that trough cameras directly provide the lane position with respect to the car. However, to locate them into the space, it is essential to define them as the locus of points sharing the same distance with respect to the reference trajectory.

$$\begin{aligned} f(X): \quad & I \subset \mathbb{R} \to \mathbb{R} \\ & X \in I \to f(X) = Y(X), \\ & C = \{f(X), X \in I\} = f(I), \end{aligned}$$

Hence the ideal trajectory is

$$Y(X) = 4\sin\left(\frac{2\pi}{100}X\right) \qquad X \in [0, +\infty)$$

Where X and Y are the axis of the inertial RF. The tangent to the curve at each point is defined as

$$\begin{aligned} \frac{\partial Y}{\partial X} &= \frac{2\pi}{25}\cos\left(\frac{\pi}{50}X\right) \\ \alpha &= \arctan\left(\frac{\partial Y}{\partial X}\right) \\ \psi &= \frac{\pi}{2} - \alpha \end{aligned}$$

Since the lane are the set of point with the same distance d from $Y(X)$

$$Y_{lane} = Y \pm d\sin(\psi)$$

It can be noticed that this description can be used with any continuous and derivable on the set $[0, +\infty)$. The constraints that describes the lane bound of the road can be defined as a function of the states of the vehicle and provided to the NMPC algorithm as follow:

```
global d

X=x(1,:);
Y=4*sin(2*pi*X/100);

alpha=atan(2*pi/25*cos(pi*X/50));
psi=(0.5*pi-alpha);

y_s=Y+d*sin(psi);
y_i=Y-d*sin(psi);


v=y(2,:);
F=[-v,v-18,x(2,:)-y_s,-x(2,:)+y_i];
```

Code 6: Sinusoidal Lane constraints

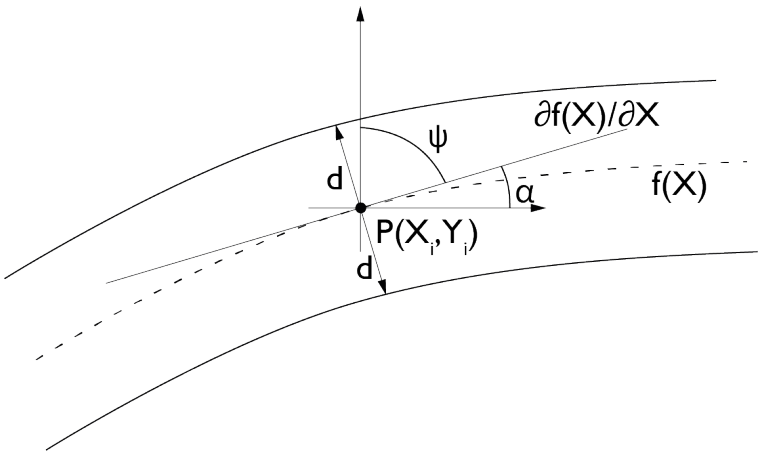


Fig. 26: Point to curve distance

In the code above the two lanes are distinguished as the superior one and the inferior one, hence, the constraints define the space between them as the available one, where the vehicle can be placed.

Once defined the constraints it is necessary to tune the controller algorithm properly, setting the best sampling time T_s and the prediction horizon T_p. The sampling used for that simulation is $T_s = 0.2s$, since the lower value allow to obtain a more accurate in the control input generation, otherwise using $T_s = 0.5s$, as shown Figure 27, the autonomous task could be achieved but with more oscillation on the trajectory.

Similarly the prediction horizon must be $T_p = 10s$, as far as possible, in order to achieve the task, otherwise, with shorter prediction the system can reach instability and after some cycle stops to follow the trajectory. While, using an higher prediction horizon the system remain accurate and stable, but the vehicle's speed is very low.

From the overall result it can be noticed that the simulation task can be achieved, the vehicle proceeds along the path as well for few sine period as for a lot of them, but with low performances.

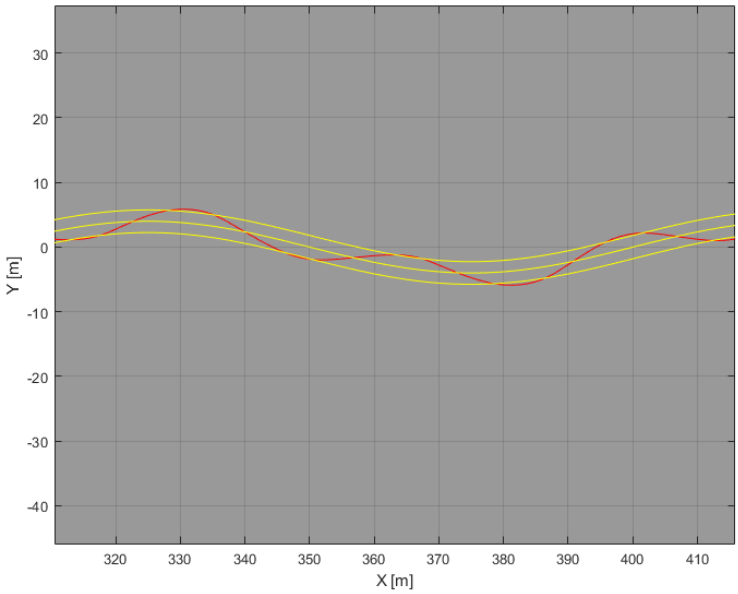


Fig. 27: Oscillating Trajectory with $T_s = 0.5\ s$

Indeed the vehicle speed for the entire simulation doesn't exceed $v = 4\ m/s$, while the reference value used in the controller configuration is set to $v = 10\ m/s$. This behaviour can be explained thanks to the references found in the literature [4], as a matter of fact the absence of the lateral dynamic in the Kinematic model used for the prediction, described in Section 1.2.3, the vehicle is able to follow the sinusoidal trajectory, but increasing the speed, the deviation from the reference position increases.

The aim of that simulation focus on the capability of the system to follow the references generated by a path planning algorithm, hence there are no constraints representing the lanes. Therefore the constraints used for the thesis study case, limits the vehicle's speed but prevent to go off the road. To improve the objective performances, as for the previous simulation's case, the prediction model should pass from a kinematic model to a dynamic one. Hence, the information about lateral dynamic allow to reach higher speed, since the NMPC algorithm, aware about those information, generates the inputs required to follow the reference speed.

4.3 Emergency Maneuver

The results coming from the previous simulation cases can be used to design an autonomous driving task for emergency application, mixing together the ability of the NMPC algorithm to stop the vehicle on a reference and the one to maintain the trajectory using the lanes as the constraints.

The goal of this simulation is to activate the autonomous driving system in order to substitute the driver in a particular condition, only for limited time in case of emergency. The autonomous

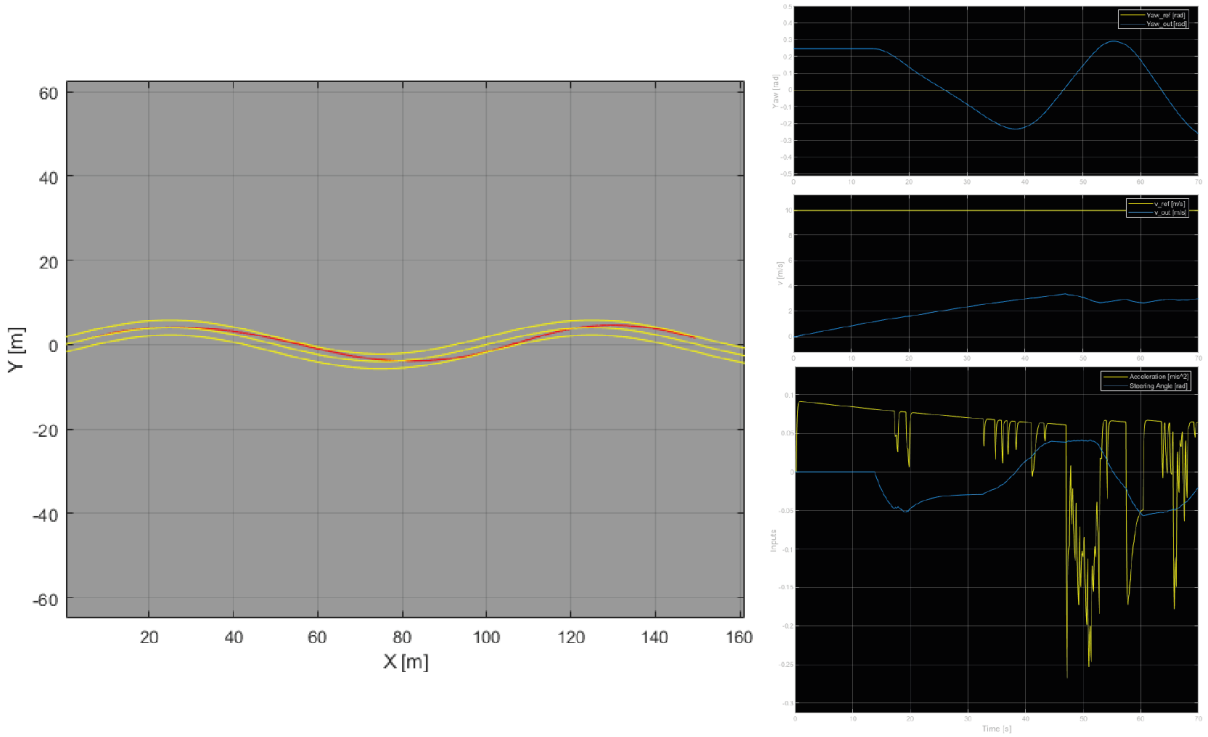


Fig. 28: Oscillating Trajectory with $T_s = 0.5\ s$

driving request is activated by the DMS (Driver Monitoring system), which contains several biometric devices, used to obtain the driver status (drowsiness, sickness, cognitive load etc.) to prevent any loss of control in driving condition. Thanks to these sensors the DMS is able to understand if the driver is capable to control, otherwise a sequence of events occurs: the DMS sends a "take over request" to recall the driver's attention, in case of no driver feedback the autonomous algorithm take the control of the vehicle to actuate a safe stop maneuver.

Actually, there are still two cases to be distinguished. When the DMS detect any driver loss of control, it must understand the specific nature of that loss. If drowsiness is detected, the NMPC engage the safe stop maneuver, otherwise if the DMS detect only a Temporary distraction, it starts an autonomous driving mode to maintain the vehicle on the trajectory and waiting for the feedback from the driver, otherwise actuate the safe stop.

The NMPC can distinguish the two cases by the required tuning and reference, but since the algorithm used for the simulations can't switch between the two configuration, it is necessary use a double trigger event and to find a compromise in the tuning that satisfies both the tasks, Table 6.

Hence, when the driver controls the vehicle the inputs are null for both the steering angle and the acceleration. For simplicity the path is a straight line with the main lane and the emergency one on the right side and there's no obstacle on the road. However the constraints in the two cases are different, since for the driving task, the car shall remain in the main lane, otherwise, for the emergency stop, both the lane represent the constraints, Figure 30.

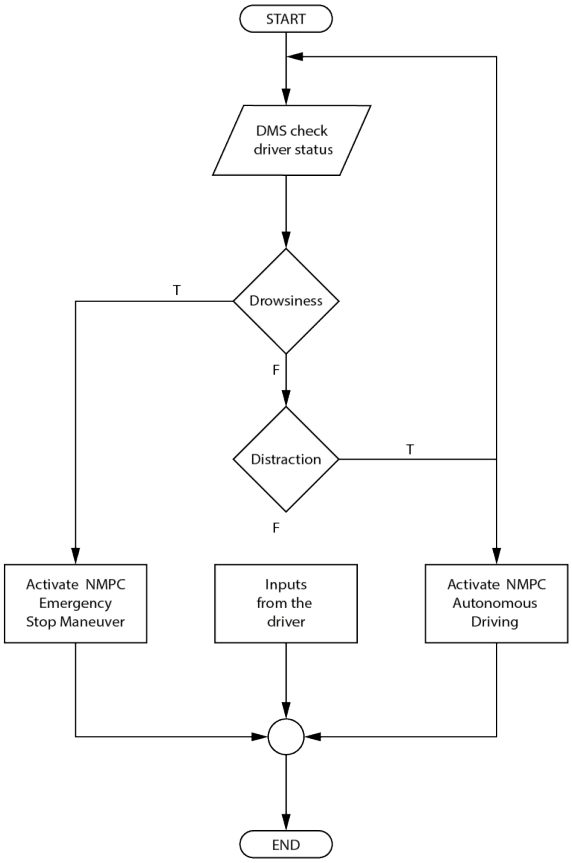


Fig. 29: Emergency Algorithm Flowchart

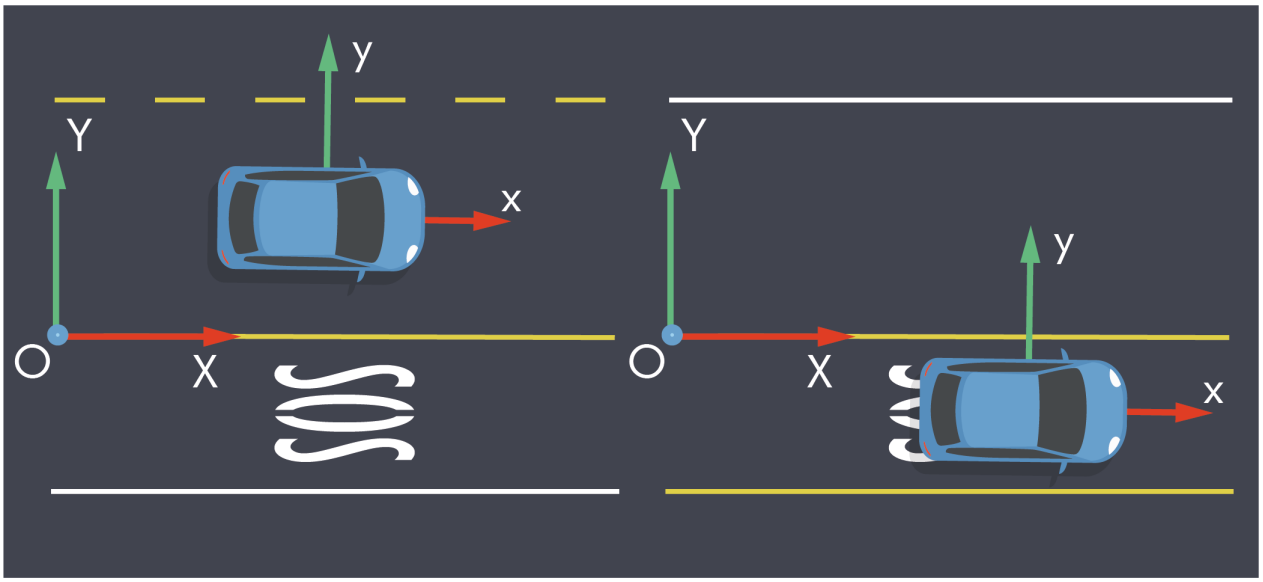


Fig. 30: Constraints for Autonomous Driving on left, constraints for Emergency Stop on right

States		Output			
Distraction	Drowsiness	NMPC	Emergency Stop	Autonomous Driving	Driver Control
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	1	0	0
1	1	1	1	0	0

Table 6: Trigger events truth table

As for the sinusoidal trajectory the straight line autonomous driving task requires mainly two references, the yaw angle and the velocity, but since the model used have to be the same for both the tasks it is necessary add one more reference that is the coordinate Y, to express the trajectory with the explicit equation. The origin of the inertial reference frame, as shown in Figure 30, is fixed on the external border of the main lane, and, to simplify the simulation, the lane is parallel to the X axis. Hence the equation of the reference path are:

$$Y_{ref}(X) = 1.875\ m \qquad \forall X \in \mathbb{R}$$

Than the other references

$$\Psi_{ref} = 0\ rad$$
$$v_{ref} = 15\ m/s$$

Y_{ref} centers the trajectory in the main lane, while the reference velocity has been set to $v_{ref} = 15m/s$, to distinguish the autonomous driving control from the regular driving condition. If the reference value was set to maintain the same speed before the detection of driver distraction, the controller can accomplish the task without generating any input.

For the testing purposes, the velocity in regular driving condition is higher with respect to the reference one, hence when the NMPC controls the vehicle the velocity must decrease and it can be easily identified from the timeplot.

Instead, the other task requires to actuate an emergency stop on the SOS lane, when the driver can't resume to control the vehicle. The new reference in this case are:

$$Y_{ref}(X) = -1.25\ m \qquad \forall X \in \mathbb{R}$$
$$\Psi_{ref} = 0\ rad$$
$$v_{ref} = 0\ m/s$$

Where Y_{ref} represent the center of the emergency lane, and the final velocity must be null, since the objective is to stop the vehicle. In that case is more important than in the other to maintain the Yaw angle at zero, since the vehicle must be parallel to the tangent of the path in its neighborhood. This assumption require to increase the weight for the output values.

To set the weight for both the tasks it is necessary to keep the hardest ones from both the configuration, in that way the resulting matrices represent the compromise between the two tuning. The weights for the emergency stop comes from the point to point simulation, where

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 14 \end{bmatrix}$$

The outputs' weight is higher on the velocity, since it decreases the oscillation around the position reference, that on contrary has a low weight, which represent only the necessity to move the vehicle on the target. The inputs' weight ha no significant values since there's no weight on the acceleration and a very low one on the steering angle.

Moving to the autonomous driving task, the weight matrices used are the same of the sinusoidal reference lane keeping, where

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}, \qquad P = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

The inputs weight is null for the acceleration, but not for the steering angle, indeed it's necessary to avoid sudden variation that can lead to instability, but still allows to variate the input enough to maintain the trajectory. The outputs' weight is low for the velocity, because there's no hard request on the reference achievement. While, concerning the yaw angle, the weight is null, but this value can't be acceptable, and should be tuned again in order to maintain the orientation of the vehicle with the trajectory.

Than, the compromise between the two tuning represent the starting point to reach the one to be used for the complete system, that is

$$R = \begin{bmatrix} 10 & 0 \\ 0 & 50000 \end{bmatrix}, \qquad P = \begin{bmatrix} 6\ 10e8 & 0 & 0 \\ 0 & 4\ 10e8 & 0 \\ 0 & 0 & 14\ 10e7 \end{bmatrix}$$

Hence, the velocity has the higher weight in the overall matrices, for an asymptotic convergence to the desired value, similarly also the position and the yaw angle, require a reasonably high weight. While, after them, the inputs' weight has ab higher value for the acceleration, and at last the steering angle.

Concerning the constraints design, two velocity limit values can be fixed, the maximum one, set to $v_{max} = 36\ m/s$, and the minimum one $v_{min} = 0\ m/s$, to avoid reverse driving condition. The lane constraint, as for the sinusoidal path, can be defined using the same mathematical model:

$$Y(X) = 1.875\ [m] \qquad \forall X$$
$$\frac{\partial Y}{\partial X} = 0$$
$$\alpha = \arctan\left(\frac{\partial Y}{\partial X}\right) \qquad \psi = \frac{\pi}{2} - \alpha$$

Since the lane are the set of point with the same distance d from $Y(X)$

$$Y_{lane} = Y \pm d\sin(\psi)$$

For both the emergency stop and the autonomous drive has been used the one coming from the emergency stop request, since there no way in the simulation environment to modify the constraint file when the simulation is in run state. Those instructions have been translated in the following code

```
function F=constraint(x,y)
sup_lane=2.75;
inf_lane=-1.5;

F=[y(1,:)-sup_lane,-y(1,:)+inf_lane,y(3,:)-36,-x(4,:)];
end
```

Code 7: Emergency Lane Stop task constraints

In the simulation model it's necessary to introduce a set of instruction to activate the NMPC and change the reference when the trigger event occurs, Figure 31.

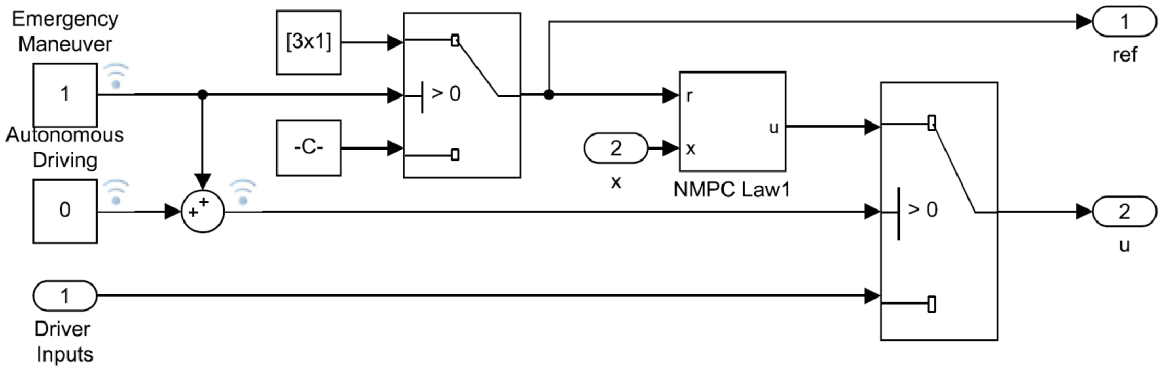


Fig. 31: Input Dispatcher

The two trigger events have been substituted by two triggers,linked to the status of the two constant values 'Autonomous Driving' and 'Emergency maneuver'. In the block scheme there are also two case switches with distinct purposes. The one on the right side controls the origin of the input signal that may be generated from the NMPC or from the driver. As shown in Table 6 this switch can be activated by both the trigger which represent the logic sum or the 'OR' logic function, sending in rest position the driver's inputs. The logic equation is

$$Tr_1 \ + Tr_2$$

Where Tr_1 corresponds to Autonomous Driving trigger and Tr_2 corresponds to the Emergency Stop trigger. Otherwise, the other switch block selects the reference that must be used from the NMPC

concerning to the trigger selected. The rest position gives back the 'Autonomous Driving' reference values, while the Emergency Maneuver reference can be obtained using its own trigger, demonstrated by

$$(Tr_1 \times Tr_2) + (\bar{T}r_1 \times Tr_2) = Tr_2 \times (Tr_1 + \bar{T}r_1) = Tr_2$$

The sampling time and the prediction horizon tuning, are the same used in the sinusoidal lane keeping

$$T_s = 0.2\ s$$
$$T_p = 10\ s$$

to obtain accurate input and output values for the vehicle.
The starting conditions are

$$\dot{x_0} = 20\ m/s$$
$$\dot{y_0} = 0\ m/s$$
$$X_0 = 0\ m$$
$$Y_0 = 1.875\ m$$
$$\dot{\psi_0} = 0\ rad/s$$
$$\psi_0 = 0\ rad$$

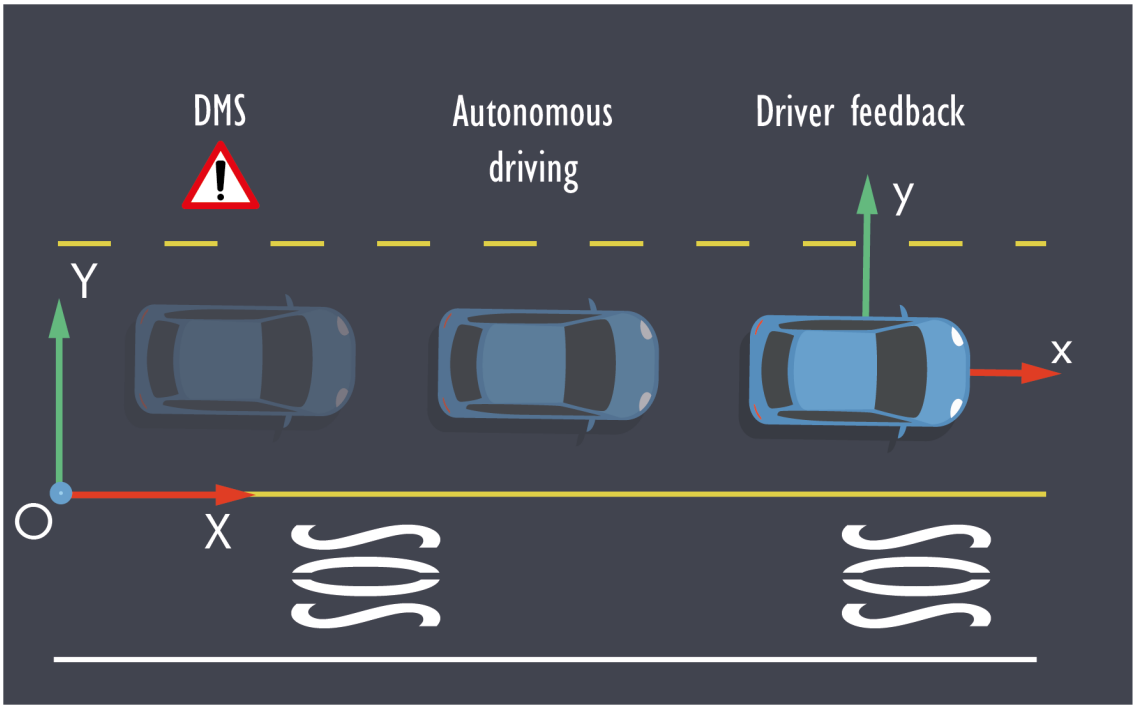


Fig. 32: Case 1: DMS detects distraction, Auton. driving activated, than driver feedback occurs

Summarizing, to accomplish the simulation request, it is necessary to simulate the system in the three possible scenario that can occur:

- the DMS detects driver's distraction, starts the autonomous driving task and alerts the driver, than the driver feedback to the incitement and resume driving;
- the DMS detects driver's distraction, starts the autonomous driving task and alerts the driver, but no feedback occurs, hence the Emergency Maneuver task is executed;
- the DMS detects driver's drowsiness, starts the Emengency Maneuver task.

For all the scenarios the simulation time has been set to infinite and the event trigger actuated manually by the user in the simulation environment itself.

The scenario results shows a negligible variation on the yaw angle Ψ and the position Y, due to the lateral dynamic, but it doesn't affect the result, while the rising time of the velocity to the target value is $t_{r,10-90} = 21.65\ s$, with very low oscillation around the reference value.

Concerning to the Inputs, the result is a moderate brake with a maximum deceleration , that slowly decrease until the target velocity has been reached, meanwhile the steering input remains at zero.

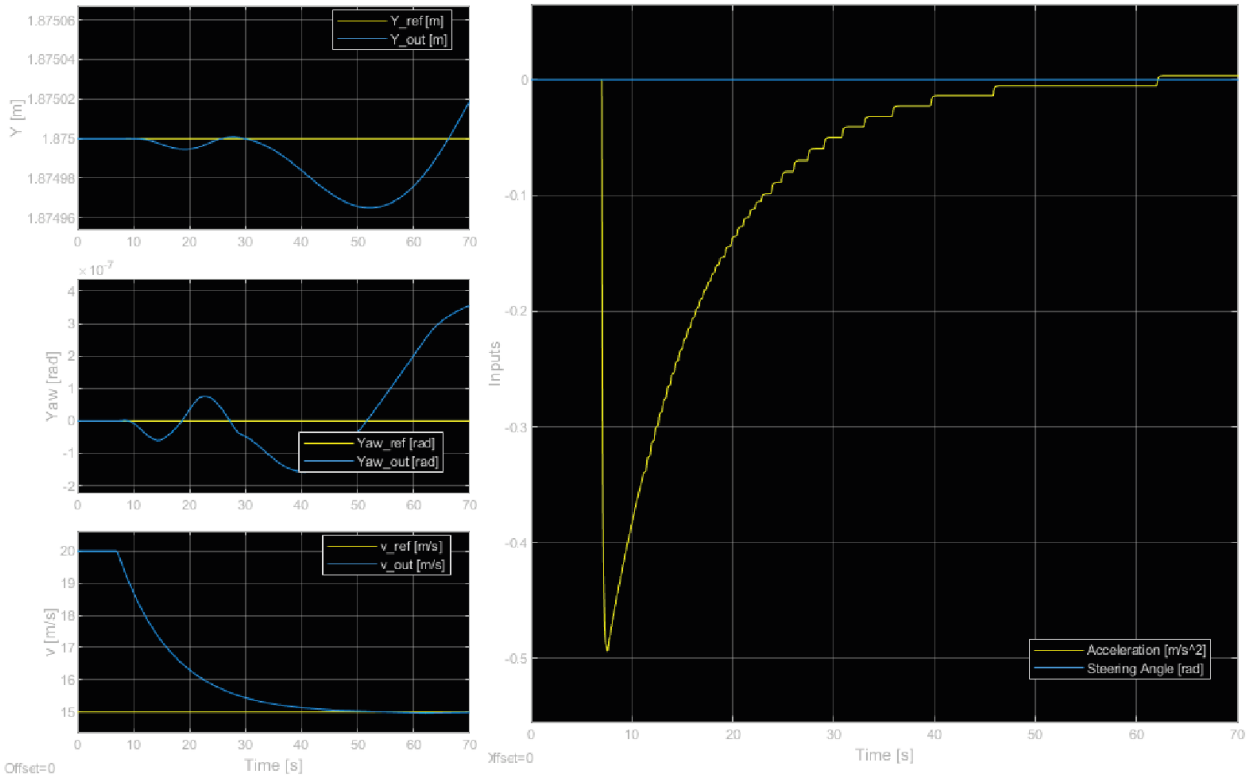


Fig. 33: Output and input time plots for Emergency Maneuver case 1

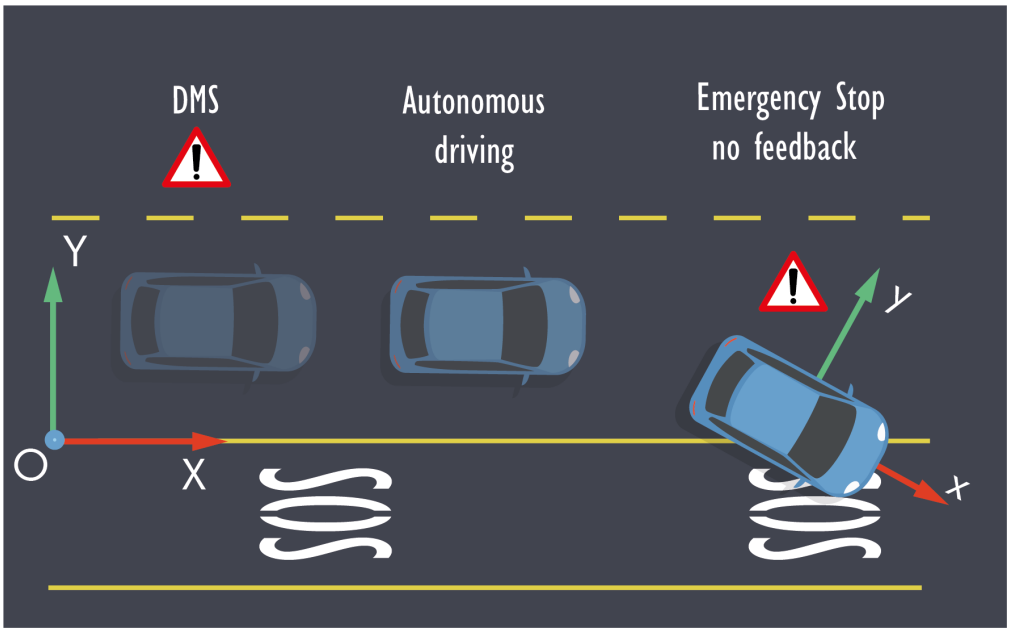


Fig. 34: Case 2: DMS detects distraction, Auto. Driving activated, no feedback requires Em. Stop

In the next scenario the first section is similar to the previous one, but after almost 10 seconds of autonomous driving the DMS detects no feedback from the driver, hence the Emergency maneuver starts. The event triggers come arbitrary during the simulation, since there's no information about the amount of time required, but their sequential activation is delayed enough to distinguish the different behaviour.

The outputs referring to Ψ and Y don't change significantly after the activation of the autonomous driving trigger, at time $t = 5\ s$, indeed, as for the previous scenario, the velocity starts to decrease according to the braking request that comes from the controller. At time $t = 15\ s$ the second trigger is activated, the new reference has been set, hence the vehicle decelerates to stop with a rise time $t_r = 30\ s$.

Therefore, since the trigger has been activated, the vehicle moves to the emergency lane in $t_r = 20\ s$ with very low oscillation around the reference value. The Yaw dynamic increases significantly during the lane changing, with a maximum absolute deviation of 0.03 radiant. Since the vehicle reaches the emergency lane, with a yaw angle almost at 0.02 radiant, the settling time at 10% of the starting value is almost 25 seconds, while the peak values has been damped at very low level.

At the same time the inputs during the autonomous driving request are null steering and a generous brake that decreases slowly, but, when the references for the Emergency stop have been set, the vehicle receive an hard brake input at the maximum decelerating level, persistent for more than one second, then this value converges slowly to zero, in almost 25 seconds but with a noisy response. Meanwhile the steering angle variation necessary to move the vehicle between the lane is not too big, with a maximum peak of 0.005 radiant.

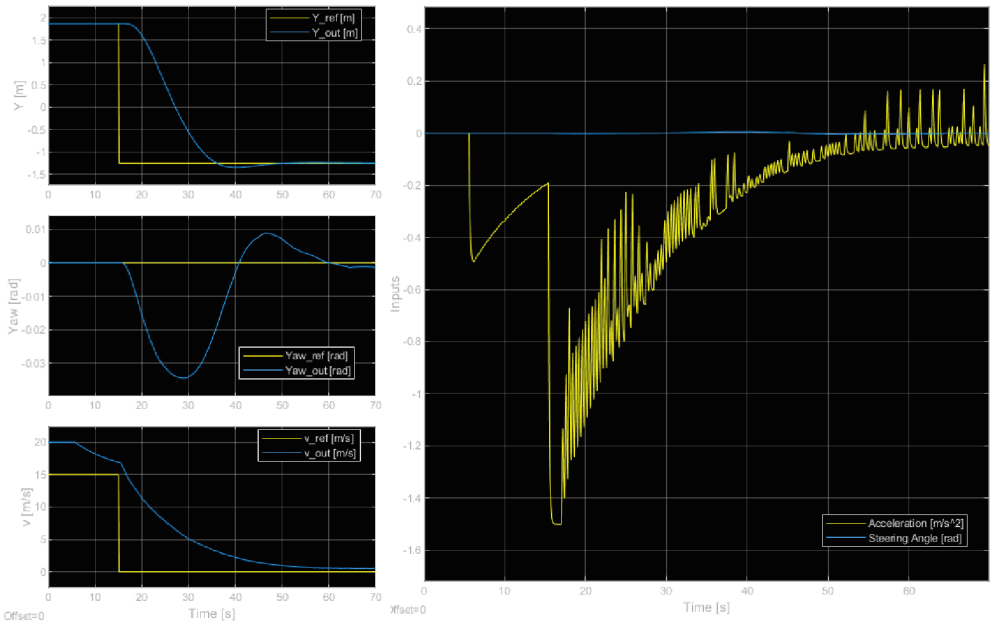


Fig. 35: Output and input time plots for Emergency Maneuver case 2

In the last scenario, where the DMS requires only the Emergency maneuver, the output variation is not too different. Once the maneuver has began, starting from an initial velocity of 20 m/s, the speed decreases asymptotically to zero, with a rise time $t_{r,10-90} = 25\ s$, the same time interval required for the output pose Y, which converges oscillating around the reference value with a low amplitude deviation. Instead, the yaw angle goes at most to 0.029 during the maneuver,than converges rapidly to zero as for the previous scenario.

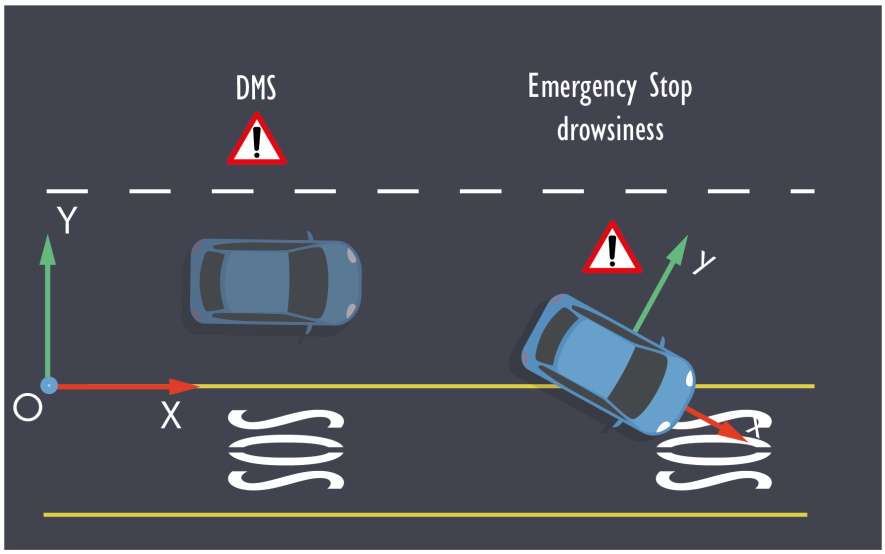


Fig. 36: Case 3: DMS Detects drowsiness, vehicle directly effectuate the Emergency Stop

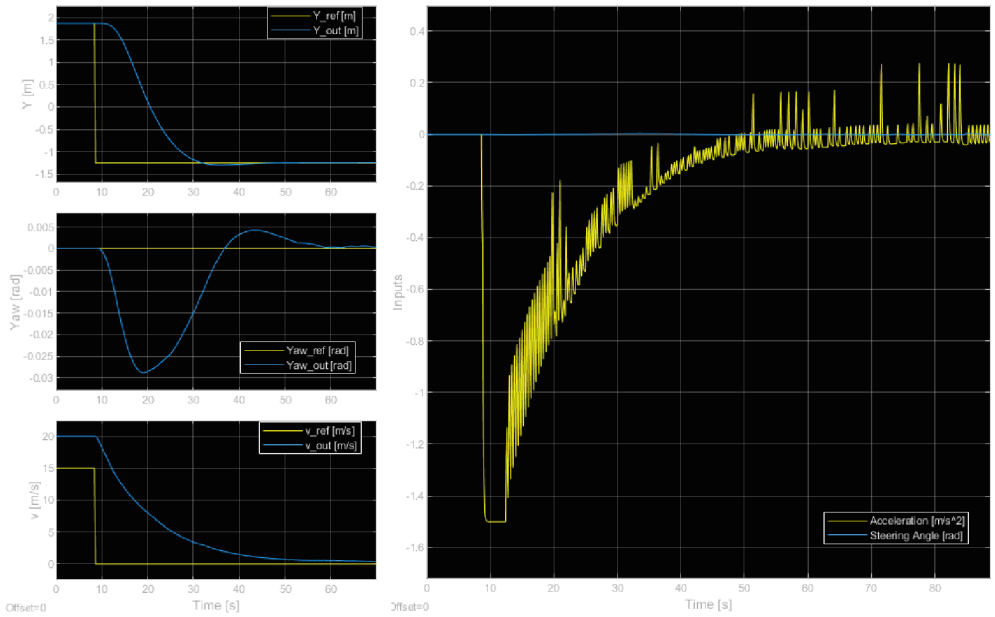


Fig. 37: Output and input time plots for Emergency Maneuver case 3

In the end, the input signals shows an hard brake operation, to stop the vehicle as soon as possible, braking it with the maximum acceleration for 1,5 seconds, than the braking slowly decreases to zero but the input generated has a noisy response that must be filtered. The steering angle uses very small angles to move the vehicle, since it moves at high speed.

Thanks to the data acknowledged from this simulation, it's possible evaluate the performances of the control law achieving the whole simulation.

From an external point of view, the robustness of the algorithm is a powerful tool that allows to achieve simultaneously several tasks, and it find the best explanation in this last simulation. In particular, the approach used in this last task, describes a reliable control system, that can be actually used to manage autonomously a car, thanks to the high performances shown.
Obviously, the control model used in that particular case of study represent the prompt in designing a more complex ADAS. To accomplish this implementation, first of all, it is necessary realize a complete study that involves also other vehicles or obstacles.

Therefore, the algorithm must be improved and modified to receive and elaborate, during the execution, the constraints from the sensors, in order to verify and test the possibility to avoid obstacles in movement.

At the same time, the model used by the NMPC for the prediction purpose should be modified into a dynamic model, since, using a model capable to understand and compensate the lateral behaviour of the car, the performances in lane keeping autonomous task improve.

Conclusion

Summarising, this thesis remarks how the study of the models used in the NMPC control law is directly influenced by the autonomous driving task to reach. As a matter of fact it is essential to delve into the models study.

This detailed study doesn't have to focus only on the models already used in these simulations, but should be dedicated to the ones that have shown a better representation of the real case, despite the limitations found (such as the dynamic models), in order to find the best compromise between accuracy in the mathematical representation and simplicity due to the algorithm feasibility.

Nevertheless the results demonstrates the ability of the NMPC algorithm to simply rearrange the objective by managing with the tuning and the constraints, and the possibility to use the same algorithm for different tasks. This solution can be very useful to reduce the amount of the control devices involved in the autonomous driving systems for future applications.

The solutions analysed, focus on the usage of these controller skills to build an autonomous driving algorithm based on safe working conditions, such as the emergency maneuver, that represent the most significant result in this paper. Indeed the NMPC adapts the same algorithm for multiple applications maintaining robust performances and a quite accurate result.
The results show also the necessity to design the objective for the autonomous vehicle in a smart way, in order to maintain a feasible system still using several constraints.

To improve the performance of the control law, it can be useful to adopt a more detailed NMPC based software, according to the possibility to manage and limit directly the variation of the inputs generated, or with other features that improve the optimization. Than, it can be also necessary to develop a detailed translation of the input commands to the real one managed by the actuators, using an alternative code in the provided dispatcher block.

The overall study has accomplished its aim, showing the efficiency of an NMPC based control system applied for autonomous driving applications and providing the tools necessary to operate with the simulation environment, in order to develop detailed analysis on the vehicle dynamics control.
To validate the whole study it is necessary to extend the experimental simulation into a more detailed vehicle plant, managing with measured states and more complex dynamics.

In the end, the proposed NMPC based control algorithm, represents one of the possible solutions for vehicle conditional automation [1], and it is one step forward in developing smarter solutions for the future society.

References

[1] S. International, "Automated driving standard j3016," 2014. [Online]. Available: https://www.sae.org/misc/pdfs/automated_driving.pdf 2, 5, 59

[2] A. Polli, "Intelligenza artificiale. guida autonoma, lo stato dell'arte," 2018, iT. [Online]. Available: https://www.prismacompany.it/intelligenza-artificiale-guida-autonoma-lo-stato-dellarte 5

[3] R. Rajamani, *Vehicle Dynamics and Control*, 1st ed. Springer, 2006. 7, 14, 19

[4] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *The Computer Journal*, vol. 27, no. 2, pp. 97–111, 1984. 16, 45, 48

[5] S. Malan, "Lecture notes fondamenti di controlli automatici." 19

[6] T. Johansen, *Selected Topics on Constrained and Nonlinear Control*. STU Bratislava - NTNU Trondheim, 2011, ch. 5. 23

[7] C. Novara, "Lecture notes nonlinear control and aerospace applications," March-June 2018. 24

Ringraziamenti

Al termine di questo importante percorso di studi, che oggi mi vede giunto al suo traquardo con la giusta dose di impegno e sacificio, rivolgo i miei sentiti e dovuti ringraziamenti a tutti coloro che in questi anni ne hanno preso parte.

Un generoso ringraziamento in particolare va al Prof. Carlo Novara, che mi ha seguito in questi ultimi sette mesi durante la stesura della mia tesi di laurea, fornendomi gli spunti e le nozioni necessarie a creare un valore aggiunto al lavoro svolto, nonchè per la disponibilità concessa nel vagliare le proposte e le soluzioni necessarie per portare a termine il progetto con perizia.

Ringrazio i miei compagni di università Ruggero Giorgio Carlo Davide Jennifer Simone e Andrea, con i quali ho potuto confrontarmi e collaborare nel corso di questi anni, in particolare a Davide Jennifer Simone Ruggero e Giorgio i quali hanno saputo regalarmi tanti momenti di leggerezza.

Ringrazio di cuore il mio amico coinquilino e compaesano Salvatore per l'aiuto e il sostegno datomi nello svolgimento di questa tesi, e per questi anni di serena convivenza al di fuori della nostra piccola realtà che mi hanno permesso di sentirmi a casa anche a centinaia di chilometri di distanza.

Un grande grazie anche a Marco un amico e fratello che mi ha aiutato a crescere nella mia passione per la musica, e per la tenacia con cui mi ha sempre sostenuto in questi anni.

Ringrazio tutti gli amici di sempre Giuseppe Michele Ciccio Antonio, che nonostante la distanza non hanno mai smesso di essermi vicino grazie all'affetto e l'amicizia che ci lega da tanti anni.

Un caloroso grazie a tutti gli zii e cugini con i quali sono cresciuto e che non hanno mai smesso di essere parte viva della mia famiglia. In particolare grazie a zio Dino per il costante supporto morale e l'aiuto economico che mi ha permesso di concentrare la mia attenzione sullo studio, e a mio cugino Angelo, a cui devo un importante aiuto nella cura grafica della tesi, che mi è stato a fianco come un fratello maggiore per tutti questi anni e mi ha aiutato ad imparare e ad interessarmi con occhio critico al mondo tecnologico.

Infine, il ringraziamento più importante va alla mia famiglia, ai miei genitori Lucio ed Anna che, al di là dell'aiuto economico, mi hanno sempre supportato e sopportato, e a cui devo la mia crescita e formazione semplicemente per tutto ciò cui sono potuto diventare oggi, a mia sorella Alessandra e a Claudio, per l'affetto il sostegno e la guida che continuano a darmi sempre, giorno dopo giorno.

Fabio Faliero