CROSSWIND STABILIZATION CONTROL SYSTEMS FOR HEAVY VEHICLES

Applied Automatic Control
Electronic Engineering for Intelligent Vehicles
Autonomous Driving Engineering
University of Bologna

A.A. 2022-2023

Federico Rovighi, Valerio Tiri and Francesco Lobriglio

August 25, 2024

Contents

List of Figures

List of Tables

Abstract

In a world in continuous evolution in many sectors new technologies are always needed; the automotive field is no exception. A goal that may be pursued in the future is to obtain fully autonomous vehicle and in order to achieve that many smaller steps are needed. For this reason many control systems have been created and some still are under development or not even conceived.

In this scenario this study focuses on the design of an automatic control that aims to increase the safety of many vehicles (the one on which it's mounted on and also all the other road users). The basic idea is to bring a road vehicle that has deviated from its original trajectory back to its intended path. The control system was designed thanks to 'MATLAB' and 'SIMULINK' and its purpose is to stabilize the trajectory of a heavy vehicle (a bus in this example) through the control of its steering wheel (steering angle) when it's proceeding on a road that is subject to strong crosswinds. This was performed through the design of a PI controller. In addition to this the system is also employed for lane change manoeuvres.

Chapter 1

Introduction

1.1 Motivations

It's common knowledge that in the last couple decades vehicles have undergone a lot of changes; especially due the fact that many new technologies were developed: electric vehicles, new ADAS and also hydrogen based systems just to cite some of them.

One aspect that can be seen as one of the most critical is the safety one, that needs a lot of attention and precision, since the smallest error can lead to destructive outcomes. For these reasons a lot of the newest ADAS or electronic systems are focused on improving the safety of all the road users.

Under these conditions it's quite clear why many car manufacturers and also factories linked to the automotive decided to create ADAS, control systems and others technologies. Many of these systems are focused on correcting the human error or reacting to particular situation on behalf of the driver in order to avoid causing further distraction or stress and allow him to focus on the road. Some systems created for this purpose already exist, such as emergency braking or adaptive cruise control.

A type of vehicles that can cause enormous damage are especially the heavy ones such as vans, buses and trucks, so it's important to develop control systems for them. These vehicles are typically used for very long trips on high-speed roads like highways, where strong winds are often present due to their frequent passage through open spaces or areas where wind forces are amplified, such as bridges. This inevitably increases fatigue and stress for the driver. Under these conditions, the vehicle may easily deviate from its path due to a strong wind gust, potentially causing the driver to lose control. Without a quick reaction to adjust the maneuver, this could lead to a road accident and potential fatalities.

In this scenario, the importance of control systems becomes evident, yet few manufacturers have implemented such systems in their products. The exception is Mercedes, which has addressed this issue with a system called '*Crosswind Stabilization/Assist*'.

1.2 Contributions

The purpose of this work is to develop a control system for a '*Setra S 515 HD*' bus shown in Figure 1.1. The aim of this system is to control the steering angle of the bus in order to return the vehicle to its original path from which it deviated due to two different factors: wind speed and wind angle.

When the wind hits the bus at a certain angle, it causes a yaw rotation, which is the rotation around the Z axis, leading the vehicle to deviate from its original trajectory. Since a yaw rotation also changes the direction in which the vehicle was heading, adjusting the steering angle can counteract the wind's effect on the yaw angle. This helps to stabilize the vehicle and improve both safety and stability. Another effect that wind has on the bus, depending on both the speed and the angle at which it hits the vehicle, is to alter its velocities, both lateral and longitudinal (Y and X axis speeds). This can make the vehicle even more dangerous if not controlled promptly.

1.3 State of art and literature comparison

Regarding the issue presented so far, given its importance, several studies have been conducted, which have been very useful in the development of this system. Many different models were addressed and various possible solutions were proposed based on different ideas and principles.

Figure 1.1: Setra S 515 HD

In the work of Jung and Kim [1] is introduced a novel compensation control strategy and disturbance observer for mitigating crosswind effects on vehicles. It employs adaptive disturbance estimation and two control modes (with and without driver intervention) using LQR (Linear-Quadratic Regulator) and different state-space models.

Kim et al. [2] focused more on the winds in order to find a method to assess the frequency of exposure to hazardous crosswinds by estimating the value for NC (Numerical Control). It considers the impact of 16 wind directions on different vehicle types and applies the method to evaluate a bridge example across multiple traffic lanes. Wind tunnel tests measured wind speed effects on the bridge deck.

Kahveci [3] developed a robust adaptive control design using the steering angle as the control input, accounting for changing environmental conditions and control input constraints. The yaw rate control strategy is validated through simulations, demonstrating efficient rejection of lateral forces, minimization of actuator failure effects and improved vehicle stability and maneuverability.

Additionaly Hübner et al [4] focused on car-trailer combinations using a linear single-track model, verified against a nonlinear double-track model under parameter variations and crosswind disturbances. The results show effective lateral stabilization potential. Practical implementation challenges include dealing with sensor noise and unmeasurable variables, requiring observer design and signal filtering.

The idea of Huang [5] was to create a predictive compensation-based handling stability control system for autonomous vehicles to handle transient crosswinds. This system integrates steering and speed control strategies to manage coupling interference. The predictive compensation control strategy effectively constrains interference and reduces control error, outperforming the Fuzzy/PID (Proportional Integral Derivative) and NNPC (Neural Network-based Predictive Control) methods in meeting control requirements.

Another study about this topic was performed by Zhang and Xie [6] comparing vehicle stability with and without AFS (Active Front Steering) control under crosswind conditions. Proving that with AFS control many dynamic parameters (yaw angle, slip angle, lateral accelerations and lateral offsets) are significantly reduced.

Finally, as previously mentioned, Mercedes-Benz is the only car manufacturer, among the leading ones, to have developed such a system. Their system, called 'Crosswind Stabilization/Assist' [7], was developed through simulations in virtual and real wind tunnels. It operates using the ESP (Electronic Stability Program), which continuously monitors the driver's steering inputs, lateral accelerations and yaw rotations using existing sensors designed to detect skids. To achieve this, it utilizes the adaptive braking systems by slightly braking the wheels on the side from which the wind is coming.

1.4 Organisation of the manuscript

This document is divided into four main parts, with the current being the first one as an introduction.

➢ Chapter 2: the model and equations are defined alongside explanatory images. This chapter details

all the essential instruments and elements required for the control system, including matrices representing states, disturbances, errors and measurements. After providing precise definitions and explanations, the chapter proceeds with the linearization of the model and an analysis of its eigenvalues, reachability and observability. It concludes with the definition and illustration of the control system to be presented.

➢ Chapter 3: focuses on the '*SIMULINK*' project with an in-depth analysis of all the conceived models, including the blocks used and all the parameters. This part also includes the results obtained from the simulations of the aforementioned models and an analysis of them as well as the tuning procedure and all the tests that were performed.

➢ Chapter 4: discussion on problematic aspects and possible improvements, along with the conclusion of the manuscript.

1.5 List of the symbols

Symbol	Description	Unit
m	Vehicle mass	kg
ψ	Yaw angle	rad
ω	Yaw rate	$\frac{rad}{s}$
ρ	Air density	$\frac{kg}{m^3}$
S_x	Frontal section of the vehicle	m^2
S_y	Lateral section of the vehicle	m^2
C_f	Aerodynamic front drag coefficient	$adim$
C_s	Aerodynamic side drag coefficient	$adim$
W	Wind speed	$\frac{m}{s}$
ξ	Wind incidence angle	rad
g	Gravitational acceleration	$\frac{m}{s^2}$
w_{part}	Weight partitioning	$adim$
δ	Steering angle	rad
P^I_{By}	Body y position in inertial coordinates	m
P^I_{Bx}	Body x position in inertial coordinates	m
l_x	Longitudinal distance from C.G. to wheels	m
l_y	Lateral distance from C.G. to wheels	m
V^I_{By}	Body y speed in inertial coordinates	$\frac{m}{s}$
V^I_{Bx}	Body x speed in inertial coordinates	$\frac{m}{s}$
C_1, C_2, C_3	Road friction coefficient	$adim$
ϵ	Infinitesimal value to avoid errors	$adim$
ω_{wheel}	Wheel angular speed	$\frac{rad}{s}$
r_{wheel}	Wheel radius	m
I^B_z	Yaw moment of inertia	$kg * m^2$

Table 1.1: List of symbols

The Table 1.1 defines all the symbols that will be used for this project providing also a definition and the measurement unit.

Chapter 2

Crosswind stabilization control system

2.1 Model and Problem Formulation

2.1.1 Case study

The main objective of the project is to stabilize the bus system when subjected to strong gusts of lateral winds. To achieve this, the steering angle and the angular speed of the vehicle's wheels are adjusted to counteract the yaw rotation induced by the wind.

Let's start the analysis of the system by focusing on the vehicle that, as previously said, is a '*Setra S 515 HD*'. An even weight distribution is assumed in the model, but the parameter that handles this distribution can be adjusted as needed. The bus is considered to be a rear-wheel traction vehicle, with the wheels numbered clockwise starting from the front-left one. A front and rear overhang are also considered, so the vehicle's axis positions are not exactly at the ends of the vehicle.

The vehicle is assumed to be traveling at high speed in a straight horizontal line on a highway (or similar road) with no slopes, bumps or roughness when, at a certain point, the wind gust blows. The small effects of the wind on roll and pitch are considered negligible to simplify the model, as these contributions are minimal. Furthermore, the wind is modeled as a single gust coming horizontally from a fixed direction perpendicular to the vehicle, with no possibility of orientation changes. Depending on this angle, the wind could either slow down or speed up the vehicle, in addition to inevitably causing lateral displacement, unless it comes directly from behind or in front.

A four-wheel model is considered, with different contributions for all four wheels, including friction coefficients, velocities, positions and more. Therefore, the considered model is neither the simplest (such as a bicycle model simplification) nor the most complex.

In this system three different kind of reference planes are considered:

- Inertial reference plane: the world reference, which is fixed and cannot change as it is bound to the world itself.
- Body reference plane: the bus reference, which is bounded to the vehicle. It has its origin at the center of gravity of the bus and its orientation depends on the orientation of the vehicle itself.
- Wheel reference plane: each wheel has its own reference plane, which is bounded to the wheel. Similar to the body reference plane, the orientation of each wheel reference plane depends on the orientation of the wheel itself.

Given the presence of different planes, rotation matrices are required to transition between them. Three matrices are needed: the first matrix, Matrix 2.1a, facilitates the transition from the inertial plane to the body plane. The other two matrices represent rotations from the body plane to the wheels plane: one for the front wheels, Matrix 2.1b, and one for the rear wheels, Matrix 2.1c. The notation R_{XY} is used to denote a rotation matrix from plane Y to plane X, while the transposed matrix R^t_{XY} is used for the reverse rotation, from X to Y.

$$R_{BI} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1a}$$

$$R_{W_{1,2}B} = \begin{bmatrix} \cos\delta & \sin\delta & 0 \\ -\sin\delta & \cos\delta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1b}$$

$$R_{W_{3,4}B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1c}$$

In the Figure 2.1 a full bus model is shown; here different forces act on the wheels and the vehicle. For example, there is drag caused by the wind, friction on the wheels and various torques and moments that contribute to rotations. The aim of the project is to counteract these effects by adjusting the steering angle and the angular speed of the wheels (and therefore the speed of the vehicle) to achieve a balance at an equilibrium point.

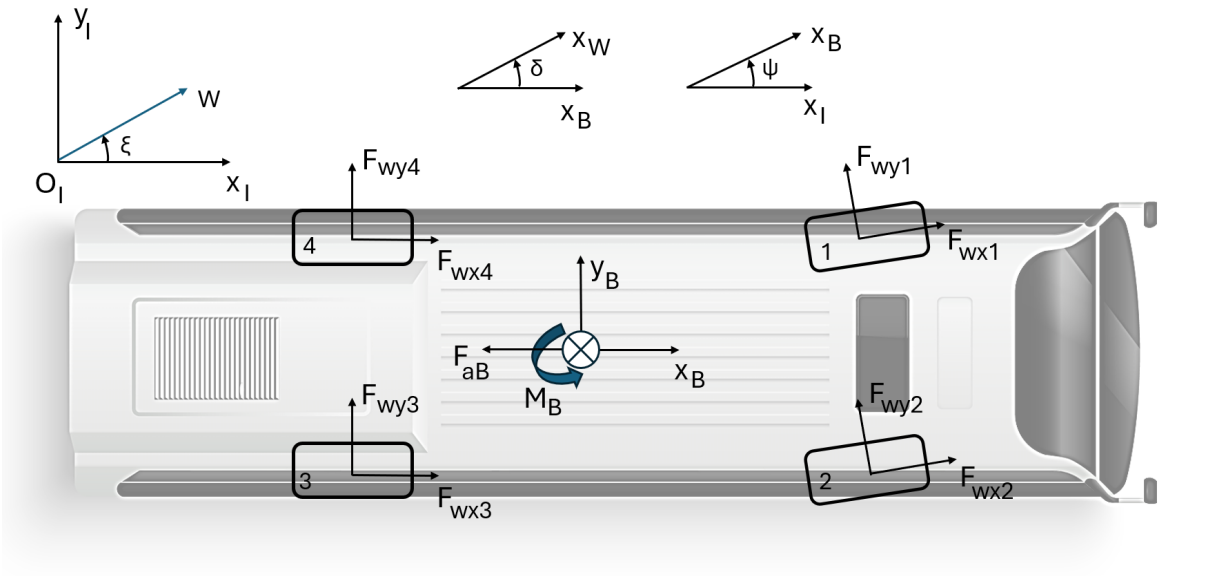


Figure 2.1: Bus model with all the reference planes, forces, speeds etc.

One of the most complex parts of the model is designing the tires, involving intricate calculations such as those for slip angle and friction coefficient. These calculations are derived from experimental studies and simulations, where formulas are defined based on highly complex parameters or functions [8]. Their model is shown in Figure 2.2.

To better comprehend the behavior of the system, it is essential to define all the forces, moments, torques and accelerations to which the bus is subjected. However, before delving into these details, the first step, given that a 3-dimensional world is considered, is to define some matrices and vectors. To facilitate understanding of the following equations, let's clarify the notation: variables will be defined in the format A_X^Y where the variable A refers to the element X in the Y reference system.

During the practical implementation in MATLAB and SIMULINK, whenever a *sign* or *abs* operation was required, problems were encountered with the Jacobian computation. To address these issues, alternative versions were chosen to replace them, avoiding these complications. These operations are shown in Equation 2.2.

$$sign\_approx(x) = \tanh 1000x \qquad abs\_approx(x) = \sqrt{x^2 + \epsilon} \tag{2.2}$$

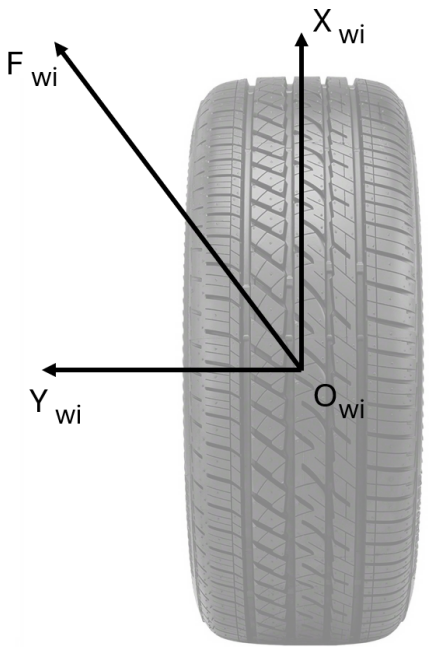


Figure 2.2: Wheel model

Now, the detailed calculations can be performed, starting with the vectors needed to locate the wheel positions within the vehicle, which will therefore be in the body reference system, these are the vectors from 2.3a to 2.3d.

$$P_{W_1}^B = \begin{bmatrix} l_x \\ l_y \\ 0 \end{bmatrix} \tag{2.3a}$$

$$P_{W_2}^B = \begin{bmatrix} l_x \\ -l_y \\ 0 \end{bmatrix} \tag{2.3b}$$

$$P_{W_3}^B = \begin{bmatrix} -l_x \\ -l_y \\ 0 \end{bmatrix} \tag{2.3c}$$

$$P_{W_4}^B = \begin{bmatrix} -l_x \\ l_y \\ 0 \end{bmatrix} \tag{2.3d}$$

These vectors are defined using l_x and l_y which, as explained in table 1.1, represent the x and y distances from the center of gravity (C.G.) to the wheels. They form a rectangle that covers the majority of the bus body (not entirely, due to the previously mentioned overhang).

Another vector that is needed is the one representing the vehicle speed in inertial coordinates that is showed in the vector 2.4.

$$V_B^I = \begin{bmatrix} V_{B_x}^I \\ V_{B_y}^I \\ 0 \end{bmatrix} \tag{2.4}$$

Thanks to these vectors it's now possible to define the wheel position equation in inertial coordinates shown in equation 2.6. Here, the position of the bus in inertial coordinates is defined as P_B^I (vector 2.5); to this, the position of the wheel inside the bus, $P_{W_i}^B$, are added.

$$P_B^I = \begin{bmatrix} P_{B_x}^I \\ P_{B_y}^I \\ 0 \end{bmatrix} \tag{2.5}$$

$$P_{W_i}^I = P_B^I + R_{BI}^t P_{W_i}^B \quad \text{for} \quad i = \{1, 2, 3, 4\} \tag{2.6}$$

By deriving the equation 2.6 it's possible to obtain the expression of the wheel speed that is expressed in equation 2.7

$$V_{W_i}^I = \frac{\partial P_B^I}{\partial t} + \frac{\partial R_{BI}^t}{\partial \psi} \cdot \frac{\partial \psi}{\partial t} \cdot \frac{\partial P_{W_i}^B}{\partial t} \quad \text{for} \quad i = \{1, 2, 3, 4\}$$

$$V_{W_i}^I = V_B^I + R_{BI}^{t'} P_{W_i}^B \omega \quad \text{for} \quad i = \{1, 2, 3, 4\} \tag{2.7}$$

A new notation for a matrix is introduced: $R_{BI}^{t'}$. This denotes the derivative of the transposed matrix R_{BI}^t with respect to ψ. The derivative of that matrix in its non-transposed form is shown as matrix 2.8

$$R'_{BI} = \begin{bmatrix} -\sin\psi & \cos\psi & 0 \\ -\cos\psi & -\sin\psi & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.8}$$

To define the friction coefficients, a change of coordinates is needed; a translation from the inertial reference system to the wheel reference one is needed. This requires multiplication by two rotation matrices, as shown in equation 2.9.

$$V_{W_i}^{W_i} = R_{W_i B} R_{BI} V_{W_i}^I \quad \text{for} \quad i = \{1, 2, 3, 4\} \tag{2.9}$$

Now, thanks to the speed previously defined, it is possible to derive the expressions needed to determine the actual friction coefficient, as it depends on specific variables. These variables are the longitudinal slip ratio λ and the slip angle β of the wheels. Their expression are shown in equation 2.10 and 2.11.

$$\lambda_i = \frac{\omega_{wheel} r_{wheel} - V_{W_i}^{W_i}(1)}{\epsilon + \max\{|\omega_{wheel} r_{wheel}|, \left|V_{W_i}^{W_i}(1)\right|} \quad \text{for} \quad i = \{3, 4\} \tag{2.10}$$

The λ values are computed only for wheels 3 and 4. The reason is that the λ_1 and λ_2 pertain to the front wheels, and since the vehicle has rear-wheel traction, the front wheels are non-driven wheels and their λ are zeros.

$$\beta_i = \arcsin \frac{V_{W_i}^{W_i}(2)}{\left|\left|V_{W_i}^{W_i}\right|\right|} \quad \text{for} \quad i = \{1, 2, 3, 4\} \tag{2.11}$$

With these two variables defined the two friction coefficient, μ_x and μ_y, can be obtained. These are expressed as $\mu_x(\lambda_i)$ and $\mu_y\left(-\frac{\beta_i}{\frac{\pi}{2}}\right)$, respectively. Thus, μ_x is a function of the longitudinal slip ratio, while μ_y is a function of the slip angle. The two expression are presented in equation 2.12a and 2.12b.

$$\mu_{x_i} = \text{sign}(\lambda_i) C_1 \left(1 - e^{-C_2|\lambda_i|}\right) - C_3 \lambda_i \qquad \text{for} \quad i = \{1,2,3,4\} \tag{2.12a}$$

$$\mu_{y_i} = \text{sign}\left(-\frac{\beta_i}{\frac{\pi}{2}}\right) C_1 \left(1 - e^{-C_2\left|-\frac{\beta_i}{\frac{\pi}{2}}\right|}\right) - C_3 \left(-\frac{\beta_i}{\frac{\pi}{2}}\right) \qquad \text{for} \quad i = \{1,2,3,4\} \tag{2.12b}$$

The three coefficient C_1, C_2 and C_3 are constant values that may vary based on road conditions, such as dry asphalt, wet asphalt, snow and more. Table 2.1 shows some possible values for these constants while the trend of the μ based on the different C values can be seen in Figure 2.3.

Road Conditions	C_1	C_2	C_3
Dry asphalt	1.2801	23.990	0.5200
Wet asphalt	1.1973	25.186	0.5373
Snow	0.1964	94.129	0.0646
Ice	0.0500	306.390	0.0000

Table 2.1: Friction coefficient possible values

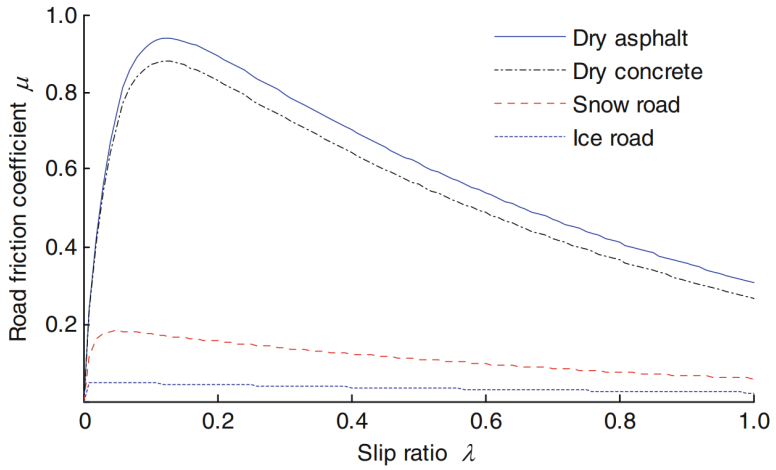


Figure 2.3: Friction coefficient with different C values

As previously mentioned, the bus model initially assumes that the weight is perfectly partitioned equally among all wheels. However, the parameter w_{part} allows to adjust this by specifying the longitudinal partitioning ratio. This means that the rear and front wheels can bear different weights. For wheels on the same axle (front pair and rear pair), there is no weighting parameter, so they will each bear half of the axle's total weight.

The normal forces on all wheels are computed according to equations 2.13a and 2.13b.

$$N_1 = N_2 = \frac{mgw_{\text{part}}}{2} \tag{2.13a}$$

$$N_3 = N_4 = \frac{mg(1 - w_{\text{part}})}{2} \tag{2.13b}$$

Now that both the normal forces and the friction coefficients are defined, it is possible to compute the friction forces on all the wheels in the wheel reference system with a simple multiplication. There will be two components: the lateral and longitudinal contributions of these forces (equations 2.14a and 2.14b), which can be combined into the vector $F_{W_i}^{W_i}$ (vector 2.14c).

$$F_{xW_i}^{W_i} = N_i \mu_{x_i} \qquad \text{for} \quad i = \{1,2,3,4\} \tag{2.14a}$$

$$F_{yW_i}^{W_i} = N_i \mu_{y_i} \qquad \text{for} \quad i = \{1,2,3,4\} \tag{2.14b}$$

$$F_{W_i}^{W_i} = \begin{bmatrix} Fx_{W_i}^{W_i} \\ Fy_{W_i}^{W_i} \\ 0 \end{bmatrix} \qquad \text{for} \quad i = \{1,2,3,4\} \tag{2.14c}$$

Since the final forces equation requires the friction forces to be defined in inertial reference system coordinates, all the components need to be multiplied by the two rotation matrices that allows to convert to inertial reference coordinates. Equation 2.15 shows the sum of all the wheel forces in the inertial reference system.

$$F_W^I = R_{BI}^t \sum_{i=1}^{4} R_{W_i B}^t F_{W_i}^{W_i} \tag{2.15}$$

Now that the friction forces have been completely modeled, the focus can be moved on the contribution of the wind. Figure 2.4 shows the Beaufort scale [9], which is the universal classification for wind speeds. Firstly, the vector representing the wind speed in inertial coordinates needs to be defined. This definition is straightforward because the wind speed and the wind incidence angle are the two disturbance parameters. Therefore, by multiplying the wind speed by the sine and cosine of the incidence angle, it is possible to obtain the lateral (equation 2.16b) and longitudinal (equation 2.16a) components of the wind contribution. These components can be joined together to form the wind vector W^I (vector 2.16c).

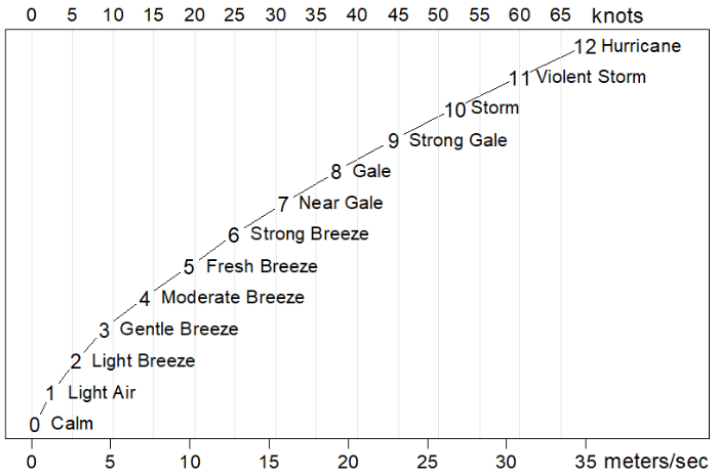


Figure 2.4: Beaufort wind force scale

$$W_x^I = |W| \cos \xi \tag{2.16a}$$

$$W_y^I = |W| \sin \xi \tag{2.16b}$$

$$W^I = \begin{bmatrix} W_x^I \\ W_y^I \\ 0 \end{bmatrix} \tag{2.16c}$$

Simply by adding the vectors 2.4 and 2.16c (representing respectively the vehicle speed and wind speed both in inertial coordinates) the resulting speed vector, being vector V_a^I (equation 2.17), is obtained.

$$V_a^I = V_b^I + W^I \tag{2.17}$$

With the wind effect applied to the vehicle is now possible to model the longitudinal and lateral aerodynamic forces, respectively shown in equations 2.18a and 2.18b.

$$F_{Ax}^I = \frac{\rho S_x C_f \operatorname{sign}(V_{ax}^I) {V_{ax}^I}^2}{2} \tag{2.18a}$$

$$F_{Ay}^I = \frac{\rho S_y C_s \operatorname{sign}(V_{ay}^I) {V_{ay}^I}^2}{2} \tag{2.18b}$$

By combining these two components into a single vector and multiplying this vector by the rotation matrix from the inertial reference system to the body reference system, the aerodynamic forces vector in body reference system (vector 2.19) is obtained.

$$F_A^B = R_{BI} \begin{bmatrix} F_{Ax}^I \\ F_{Ax}^I \\ 0 \end{bmatrix} \tag{2.19}$$

The last step regarding the forces equation is to apply Newton's second law: $F = ma$, to define the inertial acceleration of the vehicle, as shown in equation 2.20.

$$A_B^I = \frac{1}{m} F_A^I F_W^I \tag{2.20}$$

After the forces definition lets focus on the moments; the first step is to define a vector containing all the dimension of the vehicle which is the vector 2.21.

$$l_B = \begin{bmatrix} l_x \\ l_y \\ 0 \end{bmatrix} \tag{2.21}$$

As before, both an aerodynamic contribution and another related to the wheels are defined, all obtained through cross products. In this case, unlike before, the contributions must be defined in body reference system, as the moments are all applied to the vehicle. Therefore, it is formally correct to do so.

Let's start with the aerodynamic contribution, which is simply defined as the cross product between the body lengths vector and the aerodynamic forces (equation 2.19) in body coordinates, as shown in equation 2.22.

$$M_A^B = l_B \times F_A^B \tag{2.22}$$

Moving to the wheels, there are four contributes, one for each of them. Since the wheel forces are defined in wheel reference coordinates, a rotation is needed before performing the actual cross product with the wheel position in the body reference system, as described by equation 2.23.

$$M_{W_i}^B = P_{W_i}^B \times \left(R_{W_i B}^t F_{W_i}^{W_i} \right) \quad \text{for} \quad i = \{1, 2, 3, 4\} \tag{2.23}$$

Finally, the last step in defining the mathematical model is to utilize both the moments defined previously—the aerodynamic moment (equation 2.22) and the wheels moment (equation 2.23)—to obtain the yaw rotational acceleration, as shown in equation 2.24.

$$A_\psi = \frac{1}{I_z^B}\left(M_A^B + \sum_{i=1}^{4} M_{w_i}^B\right) \tag{2.24}$$

2.1.2 Model signals vectors

In the control world, it is customary to define vectors containing key quantities. The vectors that need to be defined are:

- ➢ The state vector contains the states that need to be continuously monitored and are useful for understanding the behavior of the system.
- ➢ The control vector, containing the controllable variables on which the system can act.
- ➢ The disturbance vector, composed of signals that the system cannot act on.
- ➢ The reference vector includes the target values that the system aims to achieve.
- ➢ The measurement vector, containing the variables measured by the chosen sensors.
- ➢ The error vector, which measures how much the measured values differ from the references.

In this project the state vector has a dimension of $n = 6$ and is shown in vector 2.25

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} P_{B_x}^I \\ P_{B_y}^I \\ V_{B_x}^I \\ V_{B_y}^I \\ \psi \\ \omega \end{bmatrix} = \begin{bmatrix} \text{Vehicle x position in inertial coordinates} \\ \text{Vehicle y position in inertial coordinates} \\ \text{Vehicle x speed in inertial coordinates} \\ \text{Vehicle y speed in inertial coordinates} \\ \text{Vehicle yaw angle} \\ \text{Vehicle yaw rate} \end{bmatrix} \tag{2.25}$$

Each state variable represents a different dimension related to the vehicle, specifically its position, speed and rotation.

Moving to the control vector, it has a dimension of $p = 2$ and represents the vehicle's steering angle, which can be adjusted via the steering wheel, and the wheel angular velocity, which can be modified through the accelerator. This values are shown in vector 2.26.

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \delta \\ \omega_{wheel} \end{bmatrix} = \begin{bmatrix} \text{Vehicle steering angle} \\ \text{Wheel angular velocity} \end{bmatrix} \tag{2.26}$$

Regarding the disturbances their vector has a dimension of $n_d = 2$ and its contributes are composed only by wind related elements as shown in vector 2.27.

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} W \\ \xi \end{bmatrix} = \begin{bmatrix} \text{Wind speed} \\ \text{Wind incidence angle} \end{bmatrix} \tag{2.27}$$

Regarding the references, the model has just a single one: the lateral position that the vehicle should reach; therefore, it has a dimension of $m = 1$. This perfectly summarizes the aim of the project, as the model must be capable of returning to and maintaining a certain position even in presence of disturbances. The reference is shown in the vector 2.28.

$$\mathbf{r} = \begin{bmatrix} r_1 \end{bmatrix} = \begin{bmatrix} y_{pos_{ref}} \end{bmatrix} = \begin{bmatrix} \text{Y reference position} \end{bmatrix} \tag{2.28}$$

As previously mentioned, a vector of sensor measurements is also needed in this project. Sufficient sensors have been employed so that each original state is measured by a dedicated sensor. In fact, the vector dimension is $q = n = 6$, as shown in vector 2.29. In this way, the system has been designed to operate with full-state feedback, allowing all state variables to be monitored and controlled directly for optimal performance.

The chosen sensors include:

- ➢ *Global Navigation Satellite System* (GNSS) for measuring the longitudinal position of the vehicle (x position), as well as both lateral and longitudinal speeds.
- ➢ Magnetometer for measuring the yaw angle.
- ➢ Gyroscope for measuring the yaw rate.
- ➢ For the lateral position (y), a vehicle position detection method proposed by Ali and Hussein [10] is used. This method employs cameras and visual systems to detect the vehicle's position relative to reference points, typically road lanes.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} P^I_{B_x} + \nu_1 \\ P^I_{B_y} + \nu_2 \\ V^I_{B_x} + \nu_3 \\ V^I_{B_y} + \nu_4 \\ \psi + \nu_5 \\ \omega + \nu_6 \end{bmatrix} = \begin{bmatrix} \text{Vehicle x position measured by GNSS} \\ \text{Vehicle y position measured by experimental method} \\ \text{Vehicle x speed measured by GNSS} \\ \text{Vehicle y speed measured by GNSS} \\ \text{Vehicle yaw angle measured by magnetometer} \\ \text{Vehicle yaw rate measured by gyroscope} \end{bmatrix} \tag{2.29}$$

All the ν_i are the different sensor noises that will be later addressed in subsection 3.1.2.

The error vector has the same dimension as the reference vector, $m = 2$, as they are closely related since the error represents the deviation of the measured vehicle y position from the reference value. This can be observed in vector 2.30.

$$\mathbf{e} = \begin{bmatrix} e_1 \end{bmatrix} = \begin{bmatrix} P^I_{B_y} - y_{pos_{ref}} \end{bmatrix} = \begin{bmatrix} \text{Error on vehicle y position (measured - reference)} \end{bmatrix} \tag{2.30}$$

It's common practice to combine three distinct vectors — the disturbances, sensor noises and references — to form a new vector (shown as vector 2.31 and all the components shown in vector 2.32), known as the exogenous vector. This name is apt because all the elements it contains are contributions that cannot be controlled or altered by the system itself. Its dimension is determined by the sum of the dimensions of the three vectors that compose it, which are $r = n_d + q + m = 2 + 6 + 1 = 9$.

$$\mathbf{w} = \begin{bmatrix} \mathbf{d} \\ \nu \\ \mathbf{r} \end{bmatrix} \tag{2.31}$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} W \\ \xi \\ \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \\ \nu_6 \\ y_{pos_{ref}} \end{bmatrix} = \begin{bmatrix} \text{Wind speed} \\ \text{Wind incidence angle} \\ \text{GNSS x position noise} \\ \text{Experimental method noise} \\ \text{GNSS x speed noise} \\ \text{GNSS y speed noise} \\ \text{Magnetometer noise} \\ \text{Gyroscope noise} \\ \text{Y reference position} \end{bmatrix} \tag{2.32}$$

Finally, the last vector is the derivative of vector 2.25, denoted as $\dot{\mathbf{x}}$ and depicted in vector 2.33.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} V^I_{B_x} \\ V^I_{B_y} \\ A^I_{B_x} \\ A^I_{B_y} \\ \omega \\ A_\psi \end{bmatrix} = \begin{bmatrix} \text{Vehicle x speed in inertial coordinates} \\ \text{Vehicle y speed in inertial coordinates} \\ \text{Vehicle x acceleration in inertial coordinates} \\ \text{Vehicle y acceleration in inertial coordinates} \\ \text{Vehicle yaw rate} \\ \text{Vehicle yaw acceleration} \end{bmatrix} \tag{2.33}$$

This vector is crucial because, alongside the $\mathbf{y}$ vector and $\mathbf{e}$ vector, it enables the solution of a set of ordinary differential nonlinear equations in the 'Input-State-Output' (or State Space) representation. These are elaborated in detail in equation 2.34.

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u}, \mathbf{w}) \\ \mathbf{e} &= h_e(\mathbf{x}, \mathbf{u}, \mathbf{w}) \end{aligned} \tag{2.34}$$

2.2 Model Analysis

In many engineering and scientific contexts, like this one, physical systems exhibit nonlinear behaviors, making their analysis and control particularly complex. Linearization allows the use of linear analysis tools and control techniques, making the design of controllers and stability analysis more manageable. While linearization reduces the complexity of the problem, it is important to note that the linear approximation is valid only in the vicinity of the chosen operating point. Nonetheless, this simplification is often sufficient to achieve satisfactory system performance in many practical applications.

Starting from equation 2.34, three integral curves $\mathbf{u}^*$, $\mathbf{w}^*$ and $\mathbf{x}^*$ can be defined, resulting in the new equation 2.35.

$$\begin{aligned} \dot{\mathbf{x}}^* &= f(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) \qquad \mathbf{x}^*(t_0) = \mathbf{x_0}^* \\ \mathbf{y}^* &= h(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) \\ 0 &= h_e(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) \end{aligned} \tag{2.35}$$

By doing the difference between equation 2.34 and equation 2.35, new error signals were defined, represented by the character ˜, as shown in equation 2.36.

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{x} - \mathbf{x}^*, & \tilde{\mathbf{u}} &= \mathbf{u} - \mathbf{u}^* \\ \tilde{\mathbf{y}} &= \mathbf{y} - \mathbf{y}^*, & \tilde{\mathbf{w}} &= \mathbf{w} - \mathbf{w}^* \\ \dot{\tilde{\mathbf{x}}} &= \dot{\mathbf{x}} - \dot{\mathbf{x}^*} \end{aligned} \tag{2.36}$$

To derive the expression for $\dot{\tilde{\mathbf{x}}}$, the starting point is the equation.

$$\begin{aligned} \dot{\tilde{\mathbf{x}}} &= f(\mathbf{x}, \mathbf{u}, \mathbf{w}) - f(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) \\ &= f(\mathbf{x}^* + \tilde{\mathbf{x}}, \mathbf{u}^* + \tilde{\mathbf{u}}, \mathbf{w}^* + \tilde{\mathbf{w}}) - f(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) \end{aligned}$$

Given that f is differentiable, it can be expressed as

$$f(\mathbf{x}^* + \tilde{\mathbf{x}}, \mathbf{u}^* + \tilde{\mathbf{u}}, \mathbf{w}^* + \tilde{\mathbf{w}}) = f(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*) + \nabla f_x \tilde{\mathbf{x}} + \nabla f_u \tilde{\mathbf{u}} + \nabla f_w \tilde{\mathbf{w}} + o\left(||\tilde{\mathbf{x}}||^2, ||\tilde{\mathbf{u}}||^2, ||\tilde{\mathbf{w}}||^2\right)$$

By substituting these results into the previous equation, the following expression is obtained

$$\dot{\tilde{\mathbf{x}}} = \nabla f_x \tilde{\mathbf{x}} + \nabla f_u \tilde{\mathbf{u}} + \nabla f_w \tilde{\mathbf{w}} + o\left(||\tilde{\mathbf{x}}||^2, ||\tilde{\mathbf{u}}||^2, ||\tilde{\mathbf{w}}||^2\right)$$

By applying the same process to the signals $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{e}}$, the resulting equations are presented in equation 2.37.

$$\begin{aligned}
\dot{\tilde{\mathbf{x}}} &= A(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} + B_1(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} + B_2(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}} \\
\tilde{\mathbf{y}} &= C(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} + D_1(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} + D_2(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}} \\
\mathbf{e} &= C_e(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} + D_{e_1}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} + D_{e_2}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}}
\end{aligned} \tag{2.37}$$

With the matrices A, B_1, B_2, C, D_1, D_2, C_e, D_{e_1} and D_{e_2} defined in equations 2.38

$$\begin{aligned}
A(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} &= \nabla f_x|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} \\
B_1(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} &= \nabla f_u|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} \\
B_2(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}} &= \nabla f_w|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*}
\end{aligned}$$

$$\begin{aligned}
C(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} &= \nabla h_x|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} & C_e(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{x}} &= \nabla h_{e_x}|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} \\
D_1(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} &= \nabla h_u|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} & D_{e_1}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{u}} &= \nabla h_{e_u}|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} \\
D_2(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}} &= \nabla h_w|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*} & D_{e_2}(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)\tilde{\mathbf{w}} &= \nabla h_{e_w}|_{\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*}
\end{aligned} \tag{2.38}$$

With the theoretical background now established, the analysis of these matrices within the previously defined model can begin. The focus will first be on matrix A, which is presented in matrix 2.39 in a form containing all the symbolic values.

$$\begin{aligned}
A &= \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_6} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_6} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial x_2} & \frac{\partial f_6}{\partial x_3} & \cdots & \frac{\partial f_6}{\partial x_6}
\end{bmatrix} \\
A &= \begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & \alpha & \beta & \gamma & \delta \\
0 & 0 & \epsilon & \zeta & \eta & \theta \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & \iota & \kappa & \lambda & \mu
\end{bmatrix}
\end{aligned} \tag{2.39}$$

All the Greek letters represent various contributions that are extensive, spanning several pages each. Some of them are even too large to be fully displayed in MATLAB's command window, hence they cannot be fully written out in this report.

The same approach is applied to the matrices B_1 and B_2, shown in matrix 2.40 and matrix 2.41, respectively.

$$\begin{aligned}
B_1 &= \begin{bmatrix}
\frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\
\frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\
\vdots & \vdots \\
\frac{\partial f_6}{\partial u_1} & \frac{\partial f_6}{\partial u_2}
\end{bmatrix} \\
B_1 &= \begin{bmatrix}
0 & 0 \\
0 & 0 \\
\nu & \xi \\
\pi & \rho \\
0 & 0 \\
\sigma & \tau
\end{bmatrix}
\end{aligned} \tag{2.40}$$

Also in this case, all the Greek letters represent different contributions and they are still too extensive to be fully represented.

$$B_2 = \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \frac{\partial f_1}{\partial w_2} & \frac{\partial f_1}{\partial w_3} & \cdots & \frac{\partial f_1}{\partial w_9} \\ \frac{\partial f_2}{\partial w_1} & \frac{\partial f_2}{\partial w_2} & \frac{\partial f_2}{\partial w_3} & \cdots & \frac{\partial f_2}{\partial w_9} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial w_1} & \frac{\partial f_6}{\partial w_2} & \frac{\partial f_6}{\partial w_3} & \cdots & \frac{\partial f_6}{\partial w_9} \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Delta & \Gamma & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Omega & \Xi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \Sigma & \Psi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.41}$$

This time, the contributions are "smaller" in size, allowing them to be displayed. However, they are still too large to fit on a single line. Therefore, they will be divided into parts to form the final expression. Nonetheless, they are not as long as the previous ones that spanned multiple pages.

Starting with the Δ contribution:

$$\Delta_1 = \frac{C_f S_x W \rho \cos\xi \tanh\left(1000 V_{B_x}^I + 1000 \cos\xi \sqrt{W^2+\epsilon}\right)\left(V_{B_x}^I + \cos\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Delta_2 = \frac{C_f S_x W \rho \cos\xi \left(\tanh\left(1000 V_{B_x}^I + 1000 \cos\xi \sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{B_x}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Delta = \frac{\Delta_1 - 500\Delta_2}{m}$$

Then moving to the Γ:

$$\Gamma_1 = \frac{C_f S_x \rho \tanh\left(1000 V_{B_x}^I + 1000 \cos\xi \sqrt{W^2+\epsilon}\right)\sin\xi\sqrt{W^2+\epsilon}\left(V_{B_x}^I + \cos\xi\sqrt{W^2+\epsilon}\right)}{m}$$

$$\Gamma_2 = \frac{C_f S_x \rho \sin\xi \left(\tanh\left(1000 V_{B_x}^I + 1000 \cos\xi \sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{B_x}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2}{m}$$

$$\Gamma = -\Gamma_1 + 500\Gamma_2$$

The next contribution, denoted by Ω, is quite similar to that of Δ, differing only in certain parameters, as illustrated by the following definitions:

$$\Omega_1 = \frac{C s S_y W \rho \sin\xi \tanh\left(1000 V_{B_y}^I + 1000 \sin\xi \sqrt{W^2+\epsilon}\right)\left(V_{B_y}^I + \sin\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Omega_2 = \frac{C s S_y W \rho \sin\xi \left(\tanh\left(1000 V_{B_y}^I + 1000 \sin\xi \sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{B_y}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Omega = \frac{\Omega_1 - 500\Omega_2}{m}$$

Similar to the preceding contribution, the current one involving Ξ shares analogous characteristics with another previously defined contribution (designated as Γ), as illustrated in the following lines:

$$\Xi_1 = \frac{CsS_y\rho \tanh\left(1000V_{B_y}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)\cos\xi\sqrt{W^2+\epsilon}\left(V_{B_y}^I + \sin\xi\sqrt{W^2+\epsilon}\right)}{m}$$

$$\Xi_2 = \frac{CsS_y\rho\cos\xi\left(\tanh\left(1000V_{B_y}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{B_y}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2}{m}$$

$$\Xi = \Xi_1 - 500\Xi_2$$

Now, only the last two contributions remain, which are the longest and most complex among all. Beginning with the first, denoted as Σ:

$$\Sigma_1 = \frac{C_f S_x W\rho \tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)\cos\xi\sin\psi\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_2 = \frac{C_f S_x W\rho\cos\xi\sin\psi\left(\tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_3 = \frac{C_s S_y W\rho \tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)\cos\psi\sin\xi\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_4 = \frac{C_s S_y W\rho\cos\psi\sin\xi\left(\tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_5 = \frac{C_f S_x W\rho \tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)\cos\xi\cos\psi\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_6 = \frac{C_f S_x W\rho\cos\xi\cos\psi\left(\tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_7 = \frac{C_s S_y W\rho \tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)\sin\xi\sin\psi\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)}{\sqrt{W^2+\epsilon}}$$

$$\Sigma_8 = \frac{C_s S_y W\rho\sin\xi\sin\psi\left(\tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2}{\sqrt{W^2+\epsilon}}$$

$$\Sigma = -\frac{l_x(\Sigma_1 - 500\Sigma_2 - \Sigma_3 + 500\Sigma_4) + l_y(\Sigma_5 - 500\Sigma_6 + \Sigma_7 - 500\Sigma_8)}{I_z^B}$$

Finally, concluding with the last contribution, denoted as Ψ:

$$\Psi_1 = C_f S_x \rho \tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)\sin\xi\sin\psi\sqrt{W^2+\epsilon}\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)$$

$$\Psi_2 = C_s S_y \rho \tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)\cos\xi\cos\psi\sqrt{W^2+\epsilon}\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)$$

$$\Psi_3 = C_f S_x \rho \sin\xi \sin\psi \left(\tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2$$

$$\Psi_4 = C_s S_y \rho \cos\xi \cos\psi \left(\tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2$$

$$\Psi_5 = C_f S_x \rho \tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)\cos\psi\sin\xi\sqrt{W^2+\epsilon}\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)$$

$$\Psi_6 = C_f S_x \rho \cos\psi \sin\xi \left(\tanh\left(1000V_{Bx}^I + 1000\cos\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{Bx}^I + \cos\xi\sqrt{W^2+\epsilon}\right)^2$$

$$\Psi_7 = C_s S_y \rho \tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)\cos\xi\sin\psi\sqrt{W^2+\epsilon}\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)$$

$$\Psi_8 = C_s S_y \rho \cos\xi \sin\psi \left(\tanh\left(1000V_{By}^I + 1000\sin\xi\sqrt{W^2+\epsilon}\right)^2 - 1\right)\sqrt{W^2+\epsilon}\left(V_{By}^I + \sin\xi\sqrt{W^2+\epsilon}\right)^2$$

$$\Psi = \frac{l_x\left(\Psi_1 + \Psi_2 - 500\Psi_3 - 500\Psi_4\right) + l_y\left(\Psi_5 - 500\Psi_6 - \Psi_7 + 500\Psi_8\right)}{I_z^B}$$

The tanh operation that appears in matrix 2.41 is a result of substituting the *sign* operation with the *sign_approx* function, showed in equation 2.2.

Those were all the necessary definitions for the matrix B_2, thus concluding the definitions related to the function f. Now, the remaining definitions concern h and h_e, with their matrices being much simpler.

A key first step is to define the two functions, h and h_e, which are represented as vector 2.42 and vector 2.43.

$$h = \begin{bmatrix} P_{B_x}^I + \nu_1 \\ P_{B_y}^I + \nu_2 \\ V_{B_x}^I + \nu_3 \\ V_{B_y}^I + \nu_4 \\ \psi + \nu_5 \\ \omega + \nu_6 \end{bmatrix} = \begin{bmatrix} \text{Vehicle x position + GNSS x position noise} \\ \text{Vehicle y position+ experimental method noise} \\ \text{Vehicle x speed + GNSS x speed noise} \\ \text{Vehicle y speed + GNSS y speed noise} \\ \text{Vehicle yaw angle + magnetometer noise} \\ \text{Vehicle yaw rate + gyroscope noise} \end{bmatrix} \tag{2.42}$$

$$h_e = \left[P_{B_y}^I - y_{pos_{ref}}\right] = \left[\text{Error on the lateral position}\right] \tag{2.43}$$

Beginning with the function h, three matrices must be defined: C, D_1 and D_2. These are represented as matrix 2.44, matrix 2.45 and matrix 2.46.

$$C = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} & \cdots & \frac{\partial h_1}{\partial x_6} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} & \cdots & \frac{\partial h_2}{\partial x_6} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_6}{\partial x_1} & \frac{\partial h_6}{\partial x_2} & \frac{\partial h_6}{\partial x_3} & \cdots & \frac{\partial h_6}{\partial x_6} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.44}$$

It is evident that this matrix is an identity matrix, as expected, since all the measured states correspond exactly to the signals defined as states in vector 2.25, which represents the state vector.

$$\begin{aligned} D_1 &= \begin{bmatrix} \frac{\partial h_1}{\partial u_1} & \frac{\partial h_1}{\partial u_2} \\ \frac{\partial h_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \vdots & \vdots \\ \frac{\partial h_6}{\partial u_1} & \frac{\partial h_6}{\partial u_2} \end{bmatrix} \\ D_1 &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{2.45}$$

$$\begin{aligned} D_2 &= \begin{bmatrix} \frac{\partial h_1}{\partial w_1} & \frac{\partial h_1}{\partial w_2} & \frac{\partial h_1}{\partial w_3} & \cdots & \frac{\partial h_1}{\partial w_9} \\ \frac{\partial h_2}{\partial w_1} & \frac{\partial h_2}{\partial w_2} & \frac{\partial h_2}{\partial w_3} & \cdots & \frac{\partial h_2}{\partial w_9} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_6}{\partial w_1} & \frac{\partial h_6}{\partial w_2} & \frac{\partial h_6}{\partial w_3} & \cdots & \frac{\partial h_6}{\partial w_9} \end{bmatrix} \\ D_2 &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{2.46}$$

Among the values of this matrix, the only contributions present are those caused by the noises from all the sensors, as represented by the vector w (shown in vector 2.32).

Finally, the matrices derived from the function h_e can be defined, namely C_e, D_{1_e} and D_{2_e}. These are represented as matrix 2.47, matrix 2.48 and matrix 2.49.

$$\begin{aligned} C_e &= \begin{bmatrix} \frac{\partial h_e}{\partial x_1} & \frac{\partial h_e}{\partial x_2} & \frac{\partial h_e}{\partial x_3} & \cdots & \frac{\partial h_e}{\partial x_6} \end{bmatrix} \\ C_e &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{2.47}$$

$$\begin{aligned} D_{e_1} &= \begin{bmatrix} \frac{\partial h_e}{\partial u_1} & \frac{\partial h_e}{\partial u_2} \end{bmatrix} \\ D_{e_1} &= \begin{bmatrix} 0 & 0 \end{bmatrix} \end{aligned} \tag{2.48}$$

$$\begin{aligned} D_{e_2} &= \begin{bmatrix} \frac{\partial h_e}{\partial w_1} & \frac{\partial h_e}{\partial w_2} & \frac{\partial h_e}{\partial w_3} & \cdots & \frac{\partial h_e}{\partial w_9} \end{bmatrix} \\ D_{e_2} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{aligned} \tag{2.49}$$

The contents of these matrices are quite predictable. For matrix 2.47, the only non-zero value is in the second position, representing the vehicle's y position in the state vector. This position is one of the only two variables appearing in the function h_e. The other variable is the negative y reference position, which causes the appearance of -1 in matrix 2.49 in the last position. This position corresponds to the last value of the exogenous vector, as shown in vector 2.32, which is the sole reference in the project.

These matrices provide insight into how different elements affect the system. For example, matrix A shows how the system's dynamics are influenced by the states. Similarly, matrices B_1 and B_2 illustrate how the controls and exogenous inputs, respectively, impact the system's dynamics. Matrices C, D_1 and D_2 describe how the states, controls and exogenous inputs modify the controlled outputs. Finally, matrices C_e, D_{e_1} and D_{e_2} detail how these elements affect the error dynamics.

2.2.1 Linearization

All the matrices in equation 2.37 are time-varying if at least one element of their triplets, namely $(\mathbf{x}^*, \mathbf{u}^*, \mathbf{w}^*)$, is time-varying. Consequently, all the matrices are constant if and only if all the elements of the aforementioned triplet are constant. Lets define the following triplet $(\mathbf{x_0}, \mathbf{u_0}, \mathbf{w_0})$ and refer to it as the equilibrium point for the Linear Time Invariant (LTI) system, represented by equation 2.37.

Lets now define the equilibrium point for this project defined by matrices 2.50 alongside the project parameter values showed in Table 2.2.

$$\mathbf{x_0} = \begin{bmatrix} 0 \\ 0 \\ 25 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{u_0} = \begin{bmatrix} 0 \\ 100 \end{bmatrix} \qquad \nu_\mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.50}$$

$$\mathbf{r_0} = \begin{bmatrix} 0 \end{bmatrix} \qquad \mathbf{d_0} = \begin{bmatrix} 25 \\ \frac{\pi}{2} \end{bmatrix} \qquad \mathbf{e_0} = \begin{bmatrix} 0 \end{bmatrix}$$

Symbol	Value			Unit
m	22240			kg
ρ	1.225			$\frac{kg}{m^3}$
S_x	7.5			m^2
S_y	36			m^2
C_f	1			$adim$
C_s	1.35			$adim$
g	9.81			$\frac{m}{s^2}$
w_{part}	0.5			$adim$
l_x	5			m
l_y	1.25			m
C_1, C_2, C_3	1.2801	23.990	0.5200	$adim$
ϵ	10^{-8}			$adim$
r_{wheel}	0.25			m

Table 2.2: Parameters values

The chosen reference point effectively places the vehicle at the origin of the coordinate system. This means the vehicle is moving with a longitudinal velocity of 25 $\frac{m}{s}$ (90 $\frac{km}{h}$) while experiencing no lateral movement. As previously mentioned, this is the point around which the system aims to be maintained, assuming a flat and straight road. The goal is to minimize the error relative to the center of the lane and to maintain a zero steering angle at steady state. To achieve this, the presence of wind blowing at an average speed of 25 $\frac{m}{s}$ is assumed, coming from the vertical direction in the positive semi-plane with respect to the x-axis.

Now that the equilibrium point has been defined, the linearization process can continue with the definition of the matrices. These matrices are computed using the values of the newly defined vectors and are represented as matrices 2.51, 2.52 and 2.53.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.4894 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5694 & 15.9084 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -0.0010 & 0.0270 & -0.1365 & -1.2844 \end{bmatrix} \tag{2.51}$$

$$B_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0.1249 \\ 7.9542 & 0 \\ 0 & 0 \\ 3.2210 & 0 \end{bmatrix} \tag{2.52}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2582 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0669 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0270 & 0.0261 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.53}$$

Then, a simulation of the model was also conducted and the results are shown in subsection 2.2.5.

2.2.2 Reachability

A system is defined as reachable if it is possible to reach a certain state from a different one within a finite time. The reachability problem involves finding all the final states that can be reached from a given initial state. In this context, a state $\mathbf{x}_{t_1}$ is defined as reachable from another state $\mathbf{x}_{t_2}$ in the time interval $[t_1, t_2]$ if there exists an input function $\mathbf{u}(\cdot)$ such that $\mathbf{x}_{t_1} = \psi(t_1, t_2, \mathbf{x}_{t_0}, \mathbf{u}(\cdot))$ [11].

From a mathematical perspective, a system is considered fully reachable if the rank of the matrix R, represented as vector 2.54, is equal to the dimension of the matrix A, which, as a reminder, is $n = 6$.

$$R = \begin{bmatrix} B_1 & AB_1 & A^2B_1 & \cdots & A^{n-1}B_1 \end{bmatrix} \tag{2.54}$$

To check if a system is fully reachable using MATLAB, the command *ctrb* is used, which computes the controllability matrix. Here is an example of the code used to check for reachability:

```
Reach = ctrb(A, B1);
lenA = length(A, B1);
if rank(Reach) == lenA
  disp('(A,B1) FULLY REACHABLE')
else
  disp('(A,B1) NOT FULLY REACHABLE')
end
```

Running this code in MATLAB confirms that the rank of the reachability matrix 'R' is indeed 6, which matches the dimension of matrix 'A'. As mentioned earlier, this indicates that the control system can achieve any state from any initial condition. Moreover, it provides insights into the state space connectivity, ensuring that transitions between any two desired states are possible.

Given these observations, it can be concluded that there is no requirement for a Kalman decomposition to identify a subsystem that is reachable. The full reachability of the system implies that the entire state space is accessible under suitable control inputs, facilitating control over all system states.

2.2.3 Observability

Certainly, after addressing the reachability problem, the next step is to consider the observability of the system. Observability in system theory pertains to determining whether the internal state of a system can be inferred or observed based on its external outputs. It involves assessing whether all internal states of the system can be uniquely determined or reconstructed from the available measurements or outputs. This property is crucial for designing effective state estimation and observer-based control strategies.

The observability problem is analogous to the reachability problem but focuses on determining a set of observable states from a given initial state. The formulation is as follows: a state $\mathbf{x}_{t_2}$ is defined as observable from another state $\mathbf{x}_{t_1}$ in the time interval $[t_1, t_2]$ if there exists an input function $\mathbf{u}(\cdot)$ such that $\mathbf{x}_{t_1} = \psi(t_1, t_2, \mathbf{x}_{t_0}, \mathbf{u}(\cdot))$ [11].

A system can be considered fully observable if the rank of the observability matrix O, represented as vector 2.55, is equal to the dimension of the matrix A, which is $n = 6$.

$$O = \begin{bmatrix} C & AC & A^2C & \cdots & A^{n-1}C \end{bmatrix} \tag{2.55}$$

In MATLAB, the observability of the system can be checked with a code similar to the one used in the previous subsection to check for reachability. This time, the command used is *obsv*:

```
O = obsv(A, C);
n_o = length(A, B1);
if rank(O) == n_o
  disp('(A,C) FULLY OBSERVABLE')
else
  disp('(A,C) NOT FULLY OBSERVABLE')
end
```

If this code is run, it is proven that the system is indeed fully observable since the rank of the observability matrix is, in fact, 6. This means that there are no 'hidden' states and all the states of the system can be inferred from the outputs. Consequently, this ensures that the entire state vector can be accurately estimated from the observed outputs. This level of observability is crucial for effective state estimation and feedback control, as it guarantees that all internal dynamics of the system can be monitored and controlled. Additionally, full observability negates the need for further decomposition techniques, such as the Kalman decomposition, to isolate observable subsystems. This comprehensive observability, combined with full reachability, signifies that the system is both fully controllable and fully observable, laying a strong foundation for robust control and estimation strategies.

2.2.4 Eigenvalues

A paramount aspect of investigating a control system is the evaluation of eigenvalues. This term provides important information regarding the system's stability properties and is closely related to the characteristic equation $det(\lambda I - A)$, where λ represents the eigenvalues of the matrix A. This equation can be considered as the representation of how the eigenvalues of the control system are related to the dynamic system behavior, with the roots of this equation, when set to zero, being the eigenvalues.

The eigenvalues give the following information based on their nature. Firstly, it is known that if the eigenvalues are real, there is an exponential growth or decay of the system over time. Particularly, if the eigenvalues are negative, then the behavior will be stable, whereas positive ones correspond to unstable behavior and the response seems to be diverging or growing boundlessly. Lastly, a real eigenvalue equal to 0 indicates that a critically stable system remains stable, neither decaying nor growing.

When dealing with complex eigenvalues, which are common in automatic control systems, each component of the eigenvalue provides distinct insights. The real part of the eigenvalue determines the stability of the system: if it is negative, it indicates that the system is stable. The imaginary part, however, reflects the oscillatory nature of the system's response, leading to periodic fluctuations around a reference state. The damping ratio, which is the ratio of the magnitude of the real part to the imaginary part, indicates how quickly these oscillations decay and the overall shape of the response. If an eigenvalue has an algebraic multiplicity greater than one, it signifies repeated modes and characteristics in the system. The desired outcome is to ensure that the real parts of the eigenvalues are negative.

In MATLAB, the eigenvalues can be computed using the function '*eig*'. When applied to the matrix A, the result is shown in matrix 2.56.

$$eig(A) = \begin{bmatrix} 0 \\ 0.2518 \\ -1.0528 + 0.5382i \\ -1.0528 - 0.5382i \\ 0 \\ -0.4894 \end{bmatrix} \tag{2.56}$$

We can observe that only a few eigenvalues have negative real parts, indicating that the system is not inherently stable. This issue will be addressed later in Subsection 2.3.3, where the stabilizing effect of the controller will be demonstrated.

2.2.5 Model validation

To prove the validity of the model, some simulations were performed to validate both the non-linear model and the linearized model. This section presents the details of the simulation setup and the results obtained.

The objective was to verify the mathematical and physical correctness of the system. To achieve this, a SIMULINK model was created, capable of performing a real simulation using a MATLAB function block that incorporates all the necessary system equations. The complete model is illustrated in Figure 2.5.

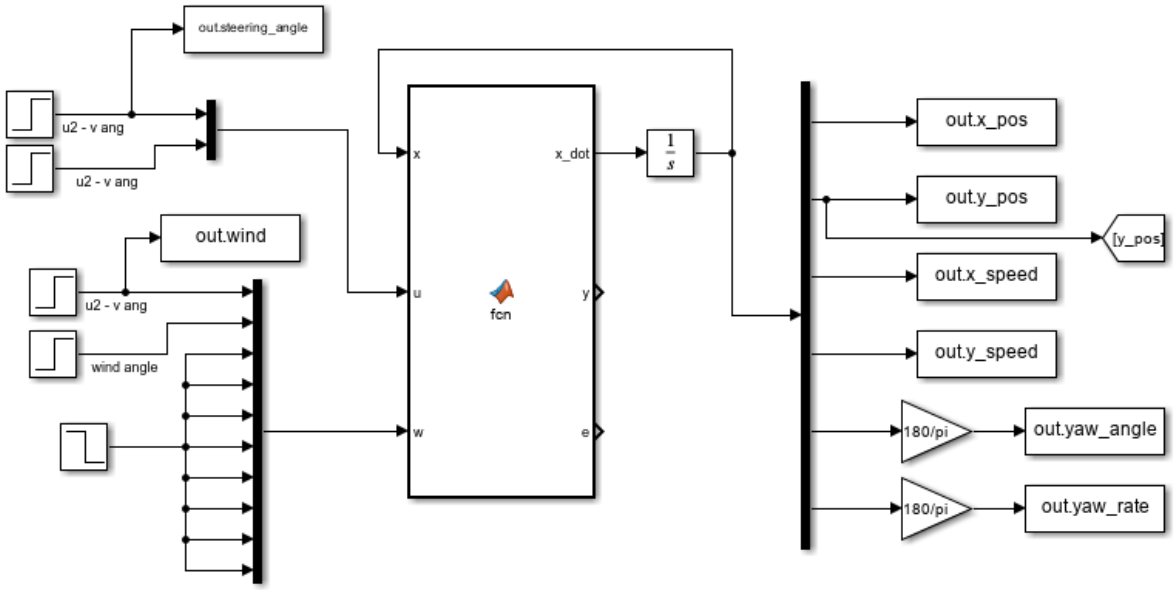


Figure 2.5: SIMULINK model used to verify the non-linear model correctness

This was done because the linearization process, by definition, works in a neighborhood of the chosen point, thereby flattening the behavior of the equations. Instead, this simulation is performed on a non-linear system to show its non-linear behavior.

The MATLAB block takes as input the state vector $\mathbf{x}$, the control vector $\mathbf{u}$ and the exogenous vector $\mathbf{w}$. The outputs are the $\dot{\mathbf{x}}$ vector, the measured output vector $\mathbf{y}$ and the error vector $\mathbf{e}$. Among all the outputs, the only one that matters is the $\dot{\mathbf{x}}$ vector, since, being in full-state feedback, all the states are observed by the sensors.

The initial condition of all the simulations are showed as vectors 2.57.

$$\mathbf{x_{init}} = \begin{bmatrix} 0 \\ 0 \\ 25 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{u_{init}} = \begin{bmatrix} 0 \\ 100 \end{bmatrix}$$

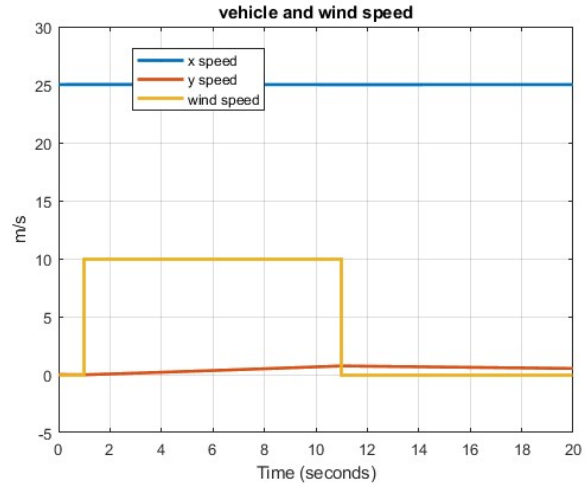
$$\mathbf{r_{init}} = \begin{bmatrix} 0 \end{bmatrix} \qquad \mathbf{d_{init}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.57}$$

Then, three correctness tests were performed with the following conditions: a simulation time of 20 seconds, no controller present so the system is operating in free evolution. These tests were:

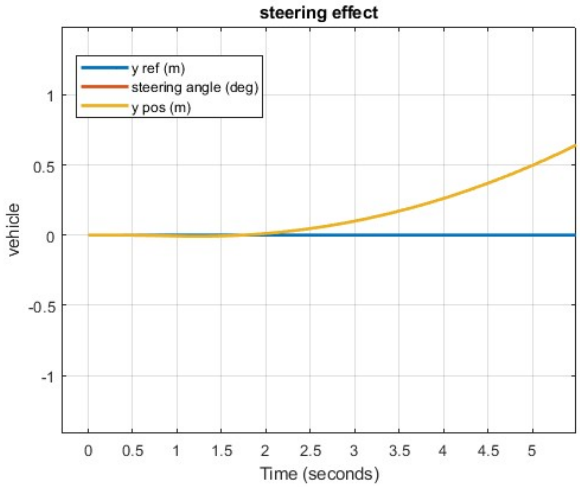
- 10 $\frac{m}{s}$ wind with incidence angle $\frac{\pi}{2}$ from $t = 1$ to $t = 11$.
- 25 $\frac{m}{s}$ wind with incidence angle $-\frac{3\pi}{4}$ from $t = 1$ to $t = 11$.
- At $t = 1$ a δ is applied as a line with a slope of 0.2 till reaching $\delta = 5$

In the figure from 2.6 to 2.8 all the simulation results can be seen.

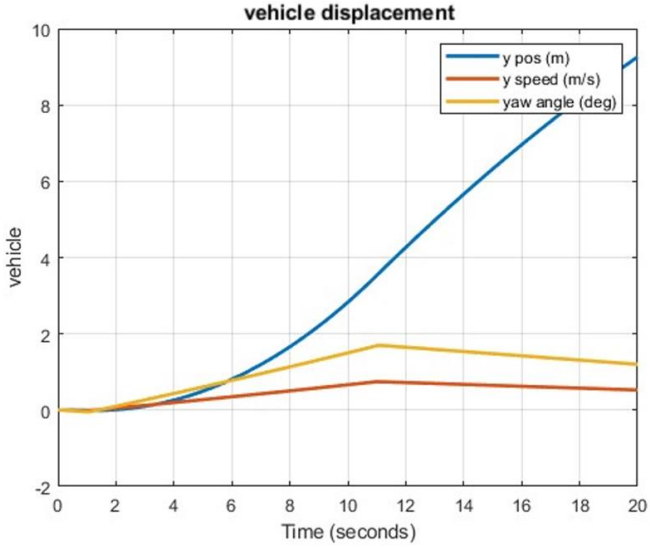
In the first simulation, it is observed that the Y position shifts in the positive direction, as expected, due to the wind coming from the right, causing the vehicle to move to the left. The second simulation shows the symmetric case, where the wind comes from the left, resulting in the vehicle shifting to the right. Additionally, a slight deviation in the X speed is noted due to an additional frontal drag from the wind, as the wind angle is not perfectly perpendicular to the X direction. In the third simulation, with no disturbances and a constant steering angle, the yaw angle describes an oblique line, leading the vehicle to gradually perform a U-turn. By the 15-second mark, the Y speed drops to zero while the X speed reaches a minimum value of -25 $\frac{m}{s}$, indicating that the vehicle has completed a 180° turn.



(a) Speeds

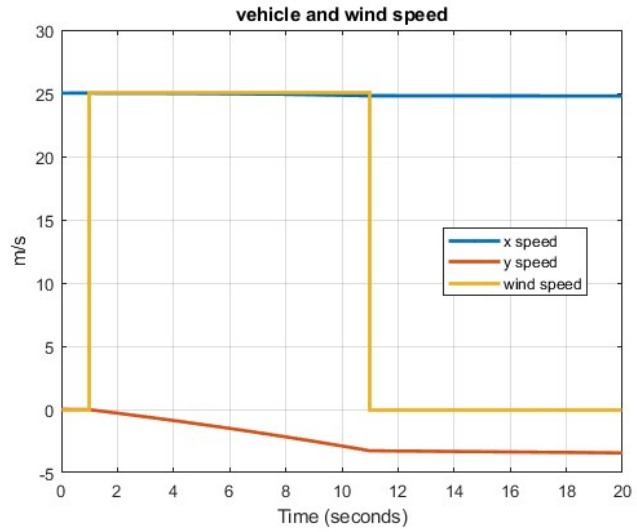


(b) Steering angle

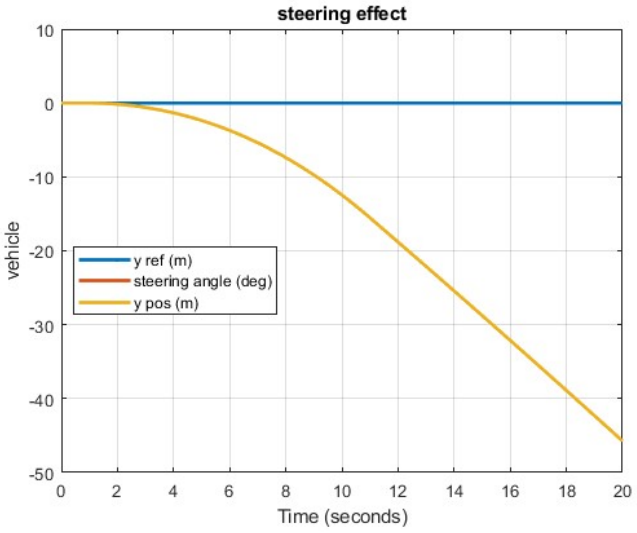


(c) General vehicle information

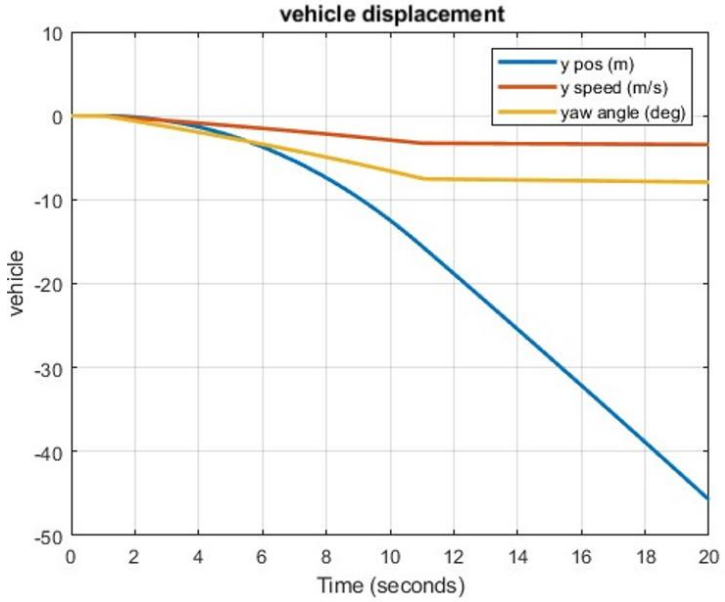
Figure 2.6: First correctness simulation for the non-linear system



(a) Speeds

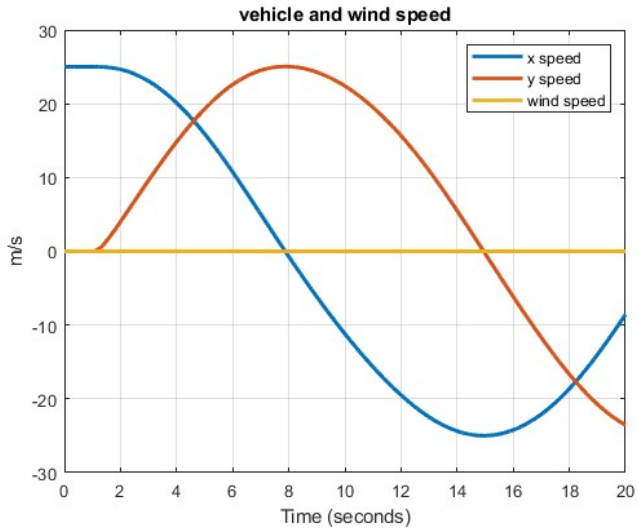


(b) Steering angle

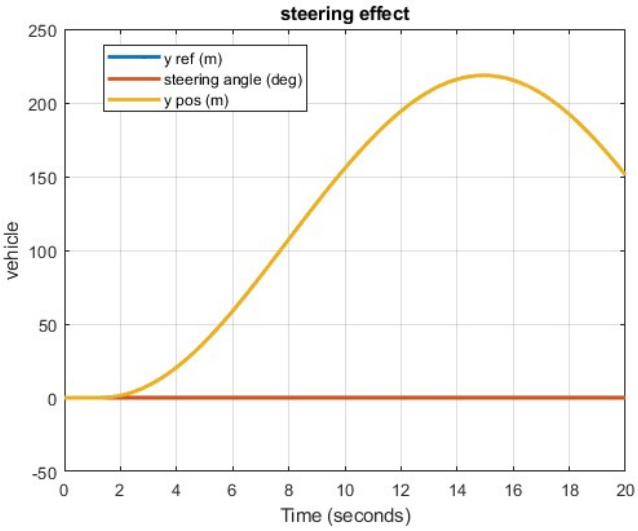


(c) General vehicle information

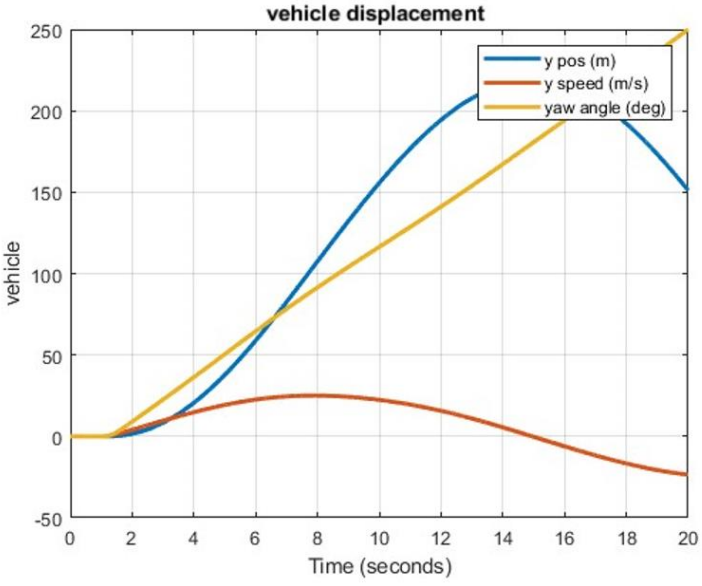
Figure 2.7: Second correctness simulation for the non-linear system



(a) Speeds



(b) Steering angle



(c) General vehicle information

Figure 2.8: Third correctness simulation for the non-linear system

Concerning the linearized model, where the SIMULINK file is shown in Figure 2.9 and the starting point is shown in vectors 2.58, the test performed are the following:

➢ Free-evolution condition with no disturbances.

➢ Free-evolution with a wind of 25 $\frac{m}{s}$ with an incidence angle of $\frac{\pi}{2}$ starting from $t = 1$ till $t = 11$.

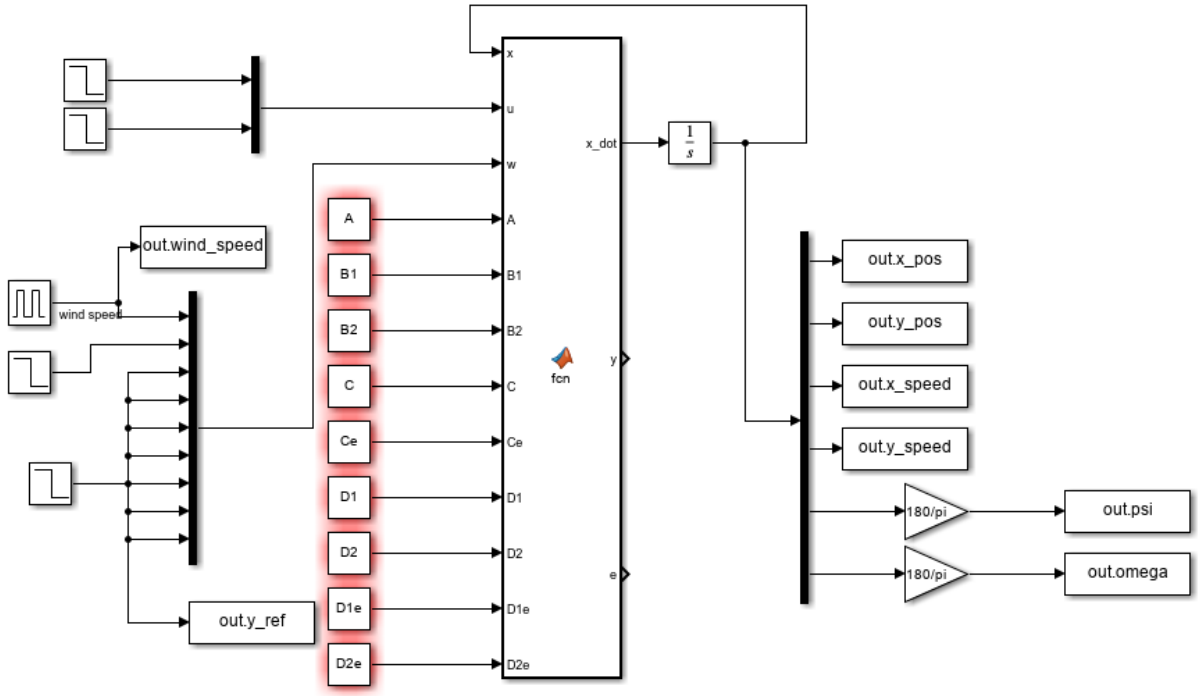


Figure 2.9: SIMULINK model used to verify the linearized model correctness

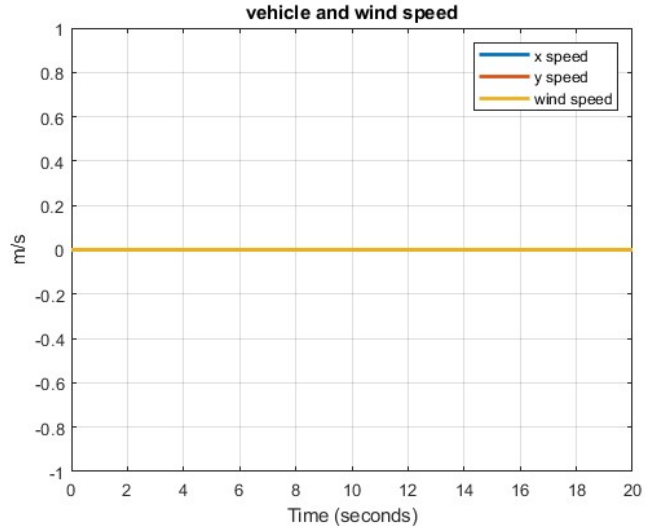
$$\mathbf{x_{init}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{u_{init}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.58}$$

$$\mathbf{r_{init}} = [0] \qquad \mathbf{d_{init}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

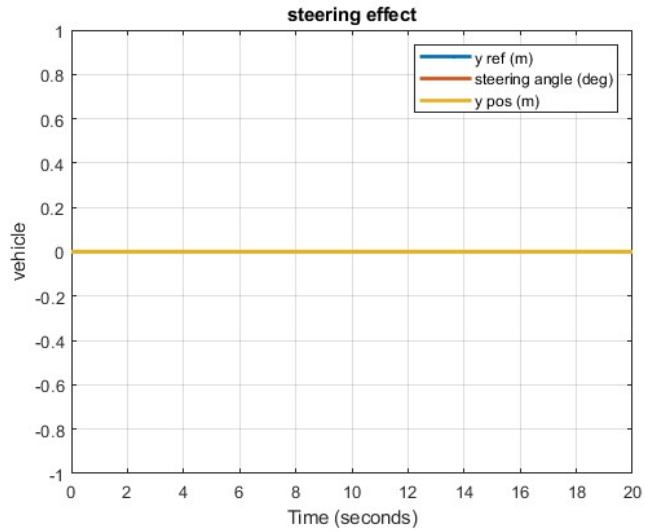
In the Figure 2.10 and Figure 2.11 the simulation results can be seen.

The first set of graphs may initially appear meaningless as they display only flat lines. However, these graphs actually highlight an important aspect of the system: its inherent stability in the absence of disturbances and controls. The flat lines indicate that the system does not deviate from its trajectory, confirming that, under steady conditions, the linearized system maintains its stability and remains in equilibrium without external influences. The X speed which is 25 $\frac{m}{s}$, is shown as remaining constant at zero in the vehicle and wind speed graph. This is because the graph displays deviations relative to the linearization point, which, as previously mentioned, is outlined in matrices 2.50.

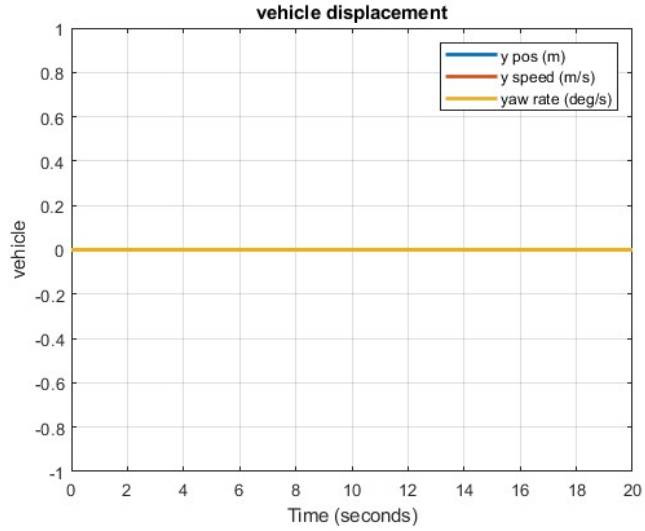
In the second set of graphs, wind is applied, and without any corrective action from the control system, the vehicle begins to deviate from its initial position as soon as the wind starts. This deviation grows continuously, affecting all states of the system, even after the wind stops. This occurs because the vehicle's states drift so far from the linearization point that the system no longer behaves as expected. When the linearized model is no longer valid the abnormal behavior displayed in the graphs shows off.



(a) Speeds

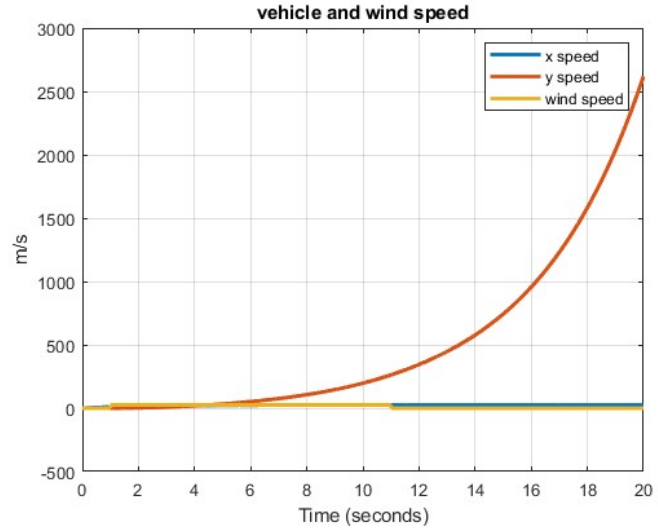


(b) Steering angle

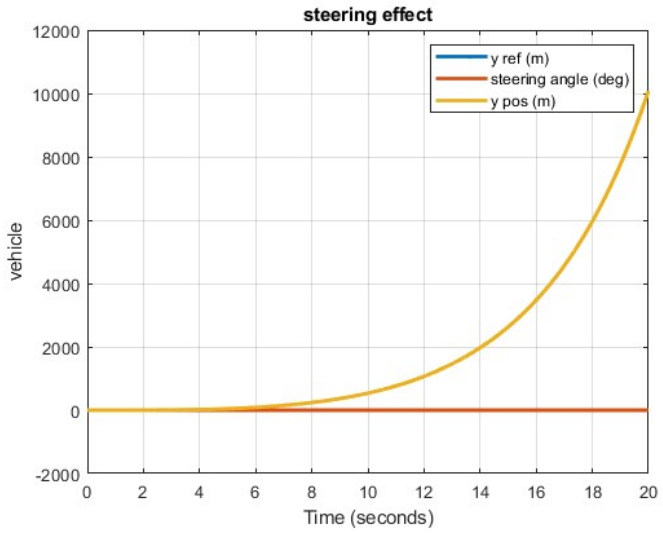


(c) General vehicle information

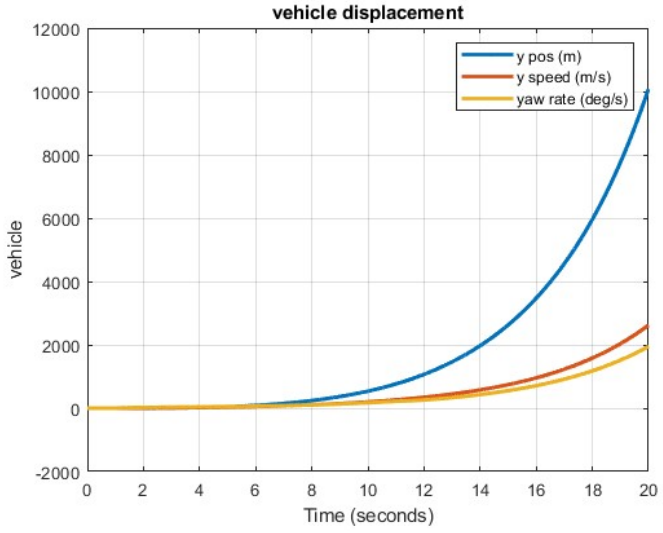
Figure 2.10: First correctness simulation for the linearized system



(a) Speeds



(b) Steering angle



(c) General vehicle information

Figure 2.11: Second correctness simulation for the linearized system

2.3 Proposed solution

The main objective of this project is to develop the control system that performs the actions described in the previous sections. The proposal is presented in the following pages, detailing its structure and the tools that were used.

2.3.1 General structure

To achieve the objective, the project was divided into two main contributions: *proportional action* and *integral action*. This led to the structure shown in Figure 2.12.

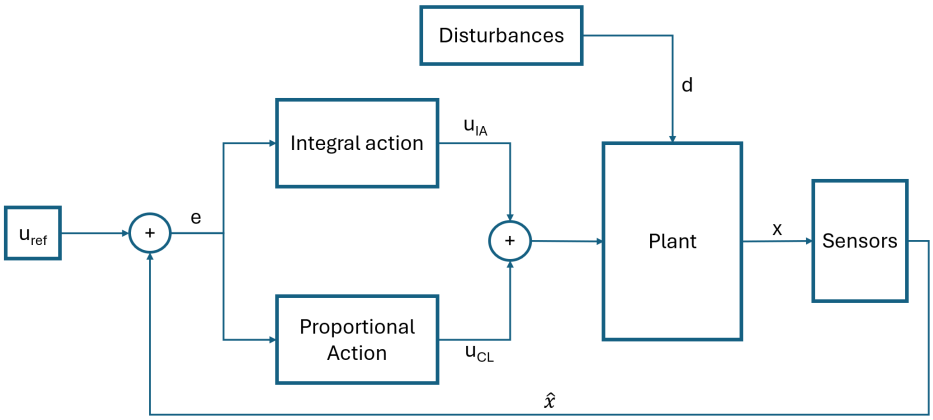


Figure 2.12: Solution control scheme

A PI controller was selected, which consists of both proportional and integral components. This approach was chosen because it strikes a good balance between simplicity and performance, making it well-suited for this kind of application. The PI controller is especially effective in minimizing steady-state error, ensuring a stable and precise response to lateral wind disturbances. Unlike a PID controller, which includes a derivative component, the PI controller simplifies the design process and reduces sensitivity to noise, which is an important consideration in a system subject to disturbances and varying driving conditions. Additionally, the PI controller offers a smoother response than a PID, reducing the risk of oscillations and overshoots, which could negatively impact vehicle comfort and safety.

Now lets analyze the 2 main contribution:

- **Proportional action**: it leverages real-time information about a system's actual behavior to improve stability and performance. By continuously adjusting the system's input based on feedback, it ensures that the desired output is achieved reliably and efficiently.
- **Integral action**: by integrating the error over time, it effectively eliminates steady-state errors, ensuring that the actual output aligns precisely with the reference set point. This leads to improved accuracy, robustness and overall system performance.

2.3.2 Feedback loop

Feedback (closed-loop) control is a fundamental concept in control theory used to regulate the behavior of dynamic systems. Unlike open-loop control, which does not utilize feedback from the system, closed-loop control continuously monitors the output of a system and adjusts the input accordingly to achieve the desired performance.

The closed-loop control system consists of several key components:

- **Controller**: The brain of the system, which determines the corrective actions based on the error signal. The controller processes the error signal to compute the necessary adjustments to the input.
- **Plant**: The process or system being controlled, such as a motor, furnace, or any mechanical or electrical system. It represents the part of the system where control actions are applied.

- **Sensors**: Devices that measure the actual output of the plant and provide feedback to the controller. These measurements are crucial for assessing the system's performance and determining the error.
- **Actuators**: Mechanisms that implement the controller's decisions by adjusting the input to the plant. While the specific role of actuators may vary depending on the control strategy, they generally influence the plant's behavior.
- **Reference Signal**: The desired value or set point that the system aims to achieve. It serves as the target for the system's output.
- **Error Signal**: The difference between the reference signal and the actual output, which drives the controller's corrective actions.

The feedback loop operates as follows: sensors measure the actual output of the plant and compare it to the reference signal to generate an error signal. The controller then processes this error signal and determines the necessary adjustments. These adjustments are implemented by the actuators, which modify the plant's input to reduce the error. This process continuously repeats, forming a loop that adapts to maintain the desired output.

Feedback control offers numerous benefits, including:

- **Stability and Accuracy**: by continuously correcting errors, feedback control helps stabilize the system against disturbances and model inaccuracies, enhancing its accuracy in achieving the desired output.
- **Robustness**: it increases the system's robustness to environmental changes and variations in system parameters, allowing the system to maintain performance despite unforeseen changes.
- **Responsiveness**: the system remains responsive to changes in the reference signal and external disturbances, improving overall performance.

An important characteristic of a system concerning closed-loop control is reachability, which was addressed in Subsection 2.2.2. It was concluded that the system is fully reachable. Therefore, it is known that a control law u can be designed to modify the behavior of the system. In the initial definition of the control system, it is assumed that the system's behavior information is available within the plant. It is established that there exists a matrix K_r that can transform the coordinate system to make the matrix $(A + B_1K_r)$ Hurwitz[1], or to achieve different eigenvalues. Thus, a state feedback control law, which ensures that the reachable system is stable in a closed loop, requires the definition of the K_r matrix.

To achieve this, the quadratic cost function is utilized and is aimed at being minimized. The cost function J is given by equation 2.59.

$$J = \int_0^1 \left(\tilde{\epsilon}^T Q \tilde{\epsilon} + \int_0^1 \tilde{u}^T R \tilde{u}\, dt \right) dt \tag{2.59}$$

Where J represents the cost function. To minimize the error, the value of Q should be increased, as higher Q places a greater penalty on any non-zero error $\tilde{\epsilon}$. Furthermore, Q should be positive semi-definite; otherwise, a negative value would result in the error $\tilde{\epsilon}(t)$ becoming infinite. Similarly, R pertains to the cost of the control input $\tilde{u}$ and it should not be zero. If R was zero, it would fail to represent a realistic actuator.

The relative "weight" of Q and R determines the size of the control action and the speed of the response; therefore, compromises have to be made. Due to the dimension of $\tilde{\epsilon}$, Q is a [7×7] matrix, while R is a matrix with dimensions just [1×1]. The cost function parameters Q and R are defined as the following code:

[1] A matrix is said to be Hurwitz if all its eigenvalues have strictly negative real parts. This implies that the system will be stable because any perturbations or deviations from equilibrium will decay over time and the system will return to its equilibrium state.

```
Aext = [A zeros(6,1);
        Ce zeros(1,1)];
B1ext = [B1;
         D1e];

Qe = diag([qe_x_pos qe_y_pos qe_x_speed qe_y_speed
           qe_yaw_angle qe_yaw_rate qe_error_steer]);
Re = re_steer;

[Ke, SolAre, poles] = lqr(Aext, B1ext, Qe, Re);

K = Ke(:, 1:lenA);
KI = Ke(1, lenA+1:end);
```

where the values for all the parameters forming the Qe and Re will be shown in Subsection 3.2.1.

2.3.3 Integral action

Integral action continuously sums (integrates) the error over time and adjusts the control input based on this accumulated error. By doing so, it ensures that any persistent, small error is gradually eliminated, leading to precise adherence to the desired set point.

Integral action in a control system involves continuously summing the error between the reference point (setpoint) and the actual output over time. This cumulative sum, known as the integral of the error (shown in equation 2.60), is then used to adjust the control input to the plant. The integral action ensures that any persistent error is corrected by generating a corrective effort that increases as the error persists.

$$I(t) = K_i \int_0^t e(\tau)\, d\tau \tag{2.60}$$

Where $I(t)$ represents the actual integral action, K_i denotes the integral gain, which determines the influence of the integral term on the overall control output, $e(\tau)$ is the error at time τ and t is the current time.

Integral action offers several benefits, including the elimination of steady-state errors, which is crucial because control systems that do not employ it tend to stabilize with a small but persistent offset from the desired set point. Furthermore, integral action increases the precision of matching the actual output to the reference set point over time, as it allows the system to recover from disturbances and return to the desired trajectory.

On the other hand, a major drawback of integral action is that it requires careful tuning. If the integral gain K_i is set too high, the system can become unstable, leading to oscillations and overshoot. Proper tuning is essential to ensure that the integral action provides precise control without introducing instability.

Integral action is a method of output feedback where the error between the reference signal and the output is addressed. Consequently, this allows for the generation of a new extended plant shown by matrix 2.61 and matrix 2.62.

$$x_{ext} = \begin{bmatrix} x \\ \eta \end{bmatrix} \tag{2.61}$$

$$\dot{x}_{ext} = \begin{bmatrix} \dot{x} \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} A & 0 \\ Ce & 0 \end{bmatrix} \begin{bmatrix} x \\ \eta \end{bmatrix} \tag{2.62}$$

In these matrices, x_{ext} is composed of the original state matrix extended with an additional row representing the integral of the error, which corresponds to the accumulated error over time. This allows for better control by accounting not only for the current error but also for the cumulative effect of any past errors.

The integral action can be defined as shown in matrix 2.63.

$$u_{IA} = K_i \begin{bmatrix} \tilde{x} \\ \eta \end{bmatrix} \tag{2.63}$$

At this point the main challenge is designing the K_i parameter.

We will now reassess the stability of the system with the controller active to determine whether it effectively stabilizes the system.

Eigenvalues are crucial for assessing system stability. A matrix is considered Hurwitz if all its eigenvalues have strictly negative real parts, which guarantees system stability, as any deviations from equilibrium will diminish over time.

The next step involves utilizing the relevant theorem which asserts that, for a fully reachable linear time-invariant (LTI) system described by $\dot{x} = Ax + Bu$, there exists a matrix K such that $A + BK$ not only remains Hurwitz but also has eigenvalues that improve the system's stability and response characteristics.

For the definition of K, appropriate values for Q, associated with the cost of minimizing error, and R, related to the expense of the control input u, were selected. The selection process will be detailed further in the following chapter of the report. Q and R were defined as diagonal matrices, matching the dimensions of the extended matrices A_{ext} and $B_{1_{ext}}$, respectively, as indicated in the code section of Subsection 2.3.2. The values for these matrices were manually specified and are presented in the following table:

qe_x_pos	1
qe_y_pos	15
qe_x_speed	1
qe_y_speed	5
qe_yaw_angle	1
qe_yaw_rate	1
qe_error_steer	10
re_steer	10

It is now possible to calculate the matrix K with MATLAB using the relationship:

$$[K_e, SolAre, poles] = lqr(A, B_1, Q, R) \tag{2.64}$$

Here, K_e represents the optimal state feedback gain matrix, $SolAre$ denotes the solution to the Riccati equation and *poles* includes the eigenvalues of the closed-loop system. The definition of K can then be specified as:

$$K_e = \begin{bmatrix} K & | & K_i \end{bmatrix}$$

This ensures that the closed-loop system matrix $A+BK$ is Hurwitz with improved stability properties.

Checking the *poles* variable now reveals the eigenvalues of the closed-loop system, as shown in equation 2.65.

$$poles = \begin{bmatrix} -0.0816 \\ -0.4842 \\ -0.6894 + 2.2979i \\ -0.6894 - 2.2979i \\ -0.9996 \\ -1.4165 \\ -6.4009 \end{bmatrix} \tag{2.65}$$

In the matrix, it is evident that all the poles have negative real parts. This indicates the stabilizing effect of the controller, successfully achieving the desired system stability. The negative real parts of the poles confirm that the controller effectively drives the system to converge to a stable equilibrium. This

result underscores the efficacy of the control design and provides confidence in its capability to maintain system stability under the conditions tested.

Chapter 3

Modelling on SIMULINK

3.1 Structure

In this chapter, the SIMULINK models will be presented, along with all its sub-components, detailing their functionalities and characteristics. The general structure can be seen in Figure 3.1.

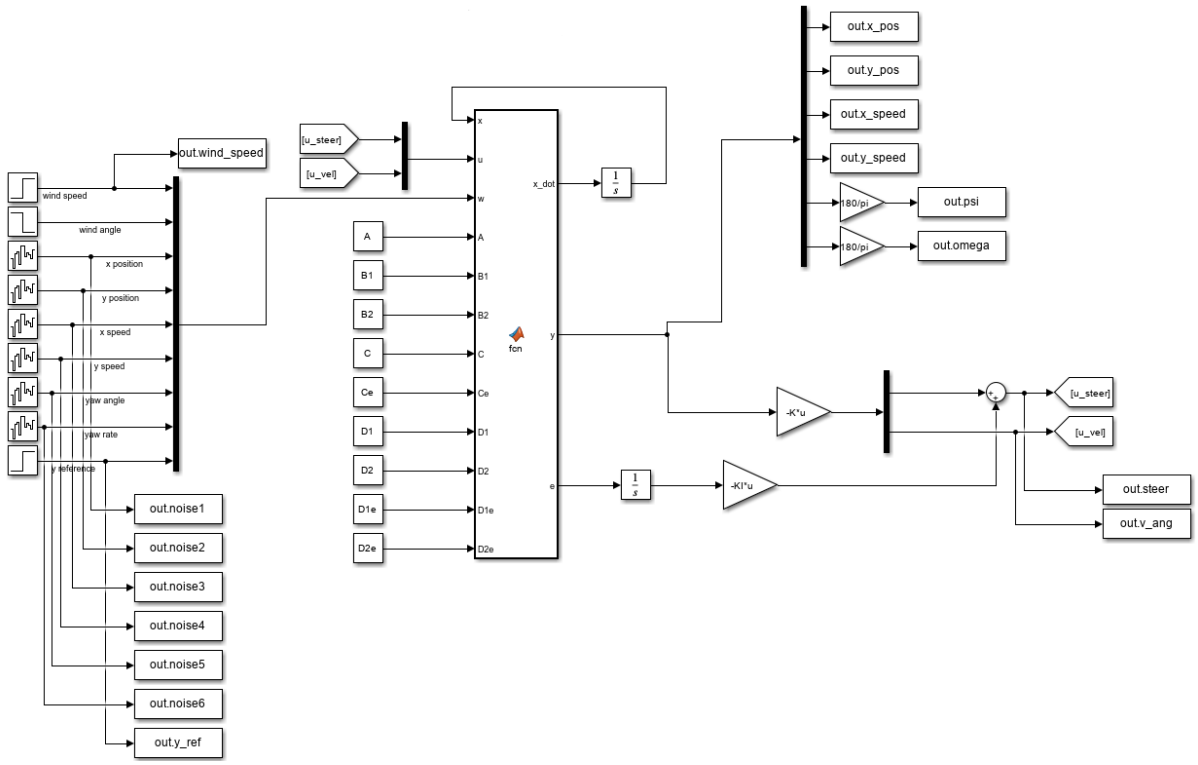


Figure 3.1: SIMULINK block diagram

3.1.1 Structure description

The main block, which is the MATLAB function, outputs the updated states, the sensor outputs and the error values based on the previous states, the control parameters, the exogenous vector and the linearization matrices (A, B1, B2, C, D1, D2, Ce, D1e, D2e) that were previously computed and stored in the MATLAB workspace, as they remain constant throughout the simulation. Inside this block, Equation 2.37 is implemented and the overall structure is illustrated in Figure 3.2.

The output provides the $\dot{x}$ dynamics of the system at the desired time step, which is then integrated and fed back into the MATLAB function to serve as the input states for the next time step.

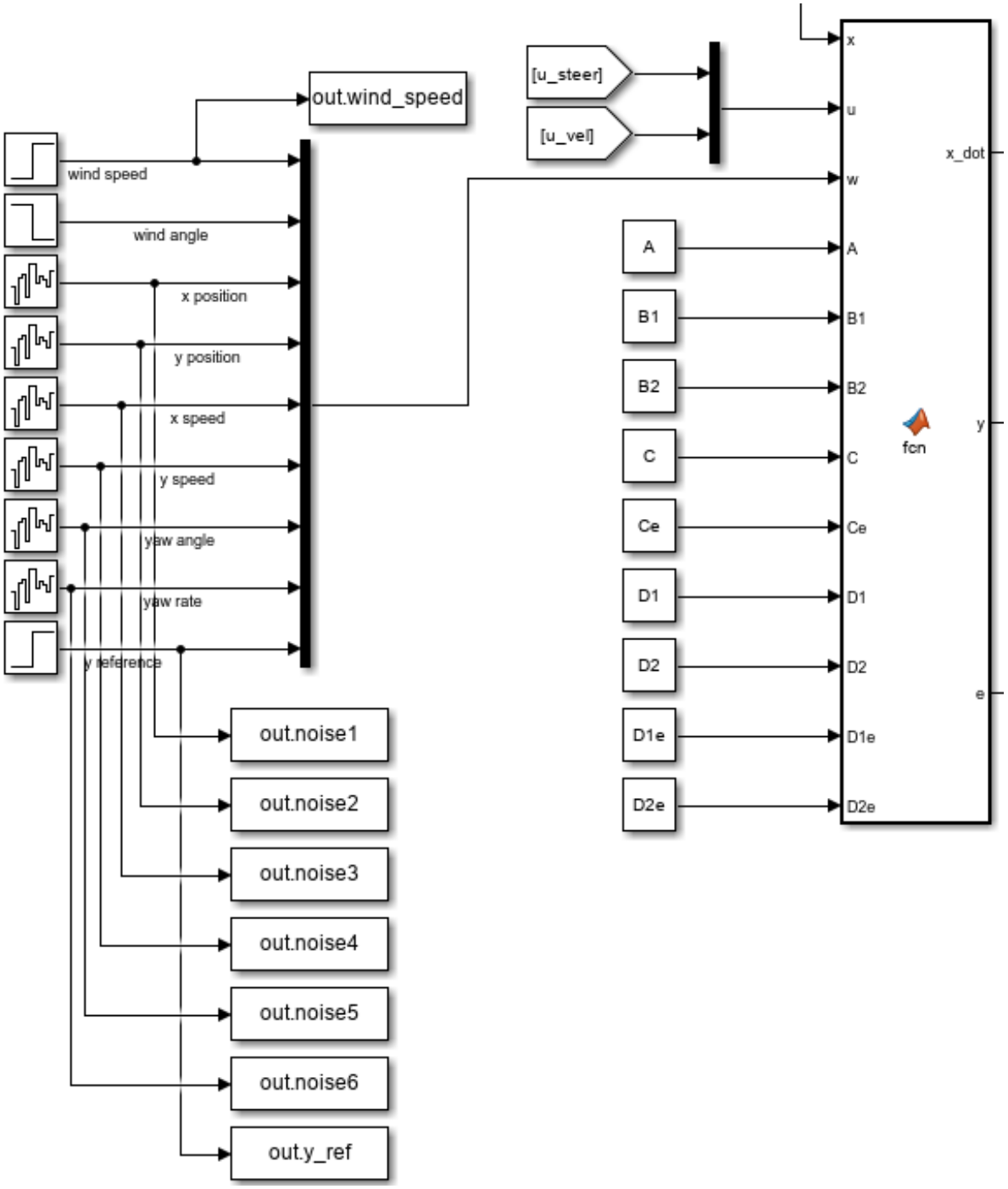


Figure 3.2: SIMULINK linearizaion part

The sensor outputs are multiplied by a negative gain $-K$ (computed in Subsection 2.3.2), which represents the proportional component of the controller. For the integral component, the error output of the MATLAB function is integrated using an integral block $\left(\frac{1}{s}\right)$ and then multiplied by a different gain, K_I. The two signals are then added together and fed back as inputs to the controller, determining the new steering angle and angular velocity of the wheels for the next time step. The structure of this PI part is shown in Figure 3.3.

3.1.2 Noise modelling

As previously mentioned, it is also necessary to address the noise issue related to the sensors. To do so, the sensor suite implemented in the ideal vehicle is recalled:

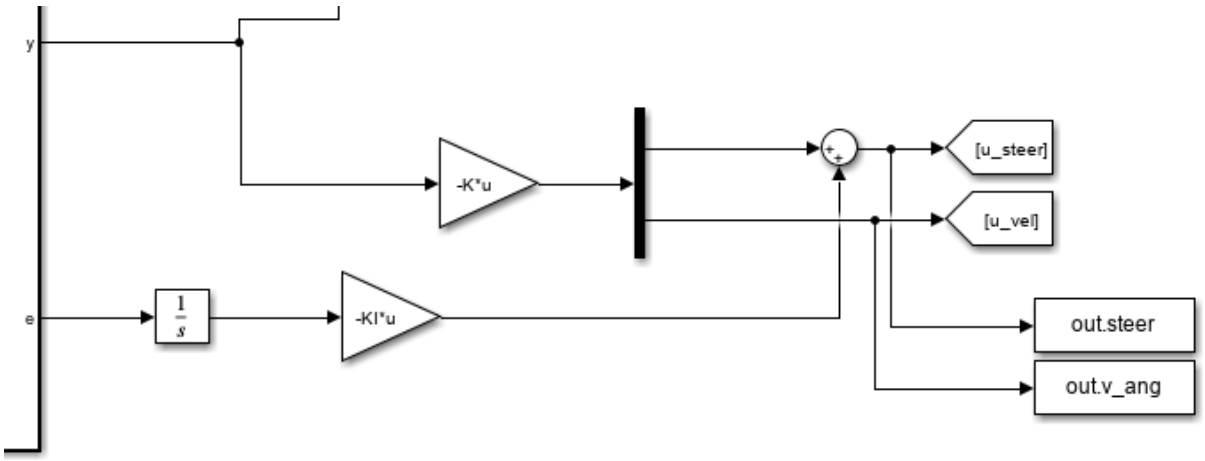


Figure 3.3: SIMULINK PI feedback

- ➢ *Global Navigation Satellite System* (GNSS)
- ➢ Magnetometer
- ➢ Gyroscope
- ➢ Vehicle position detection method

Sensor noise refers to the unwanted variations or disturbances in the data collected by a sensor, which do not represent the actual signal or physical quantity being measured. These noises can arise from various sources, including the sensor's electronic components, environmental conditions or external interference. Sensor noise can degrade the quality of the data, making it more difficult to accurately interpret the measurements. Common types of sensor noise include thermal noise, shot noise and electromagnetic interference. Reducing or filtering out sensor noise is crucial for improving the accuracy and reliability of sensor data.

The noises in this project were defined based on the datasheets of all the sensors. Specifically, the following were used:

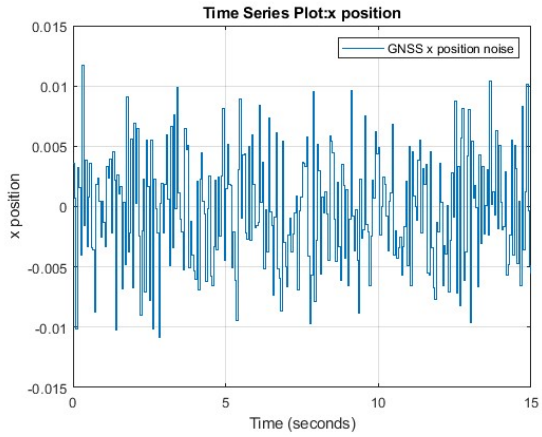
- ➢ SBG Ellipse D GNSS [12]
- ➢ SBG Ellipse D Magnetometer [12]
- ➢ SBG Pulse-40 Gyroscope [13]
- ➢ Vehicle position detection method [10]

To model the noise in SIMULINK, the *band-limited white noise* block was used. By assigning different seeds, along with appropriate sample times and noise power settings, realistic sensor noise was simulated across all measurements. The noise power values were determined based on the specifications provided in the sensor datasheets, including parameters such as absolute or relative uncertainties, ensuring that the simulated noise accurately represented real-world sensor performance.

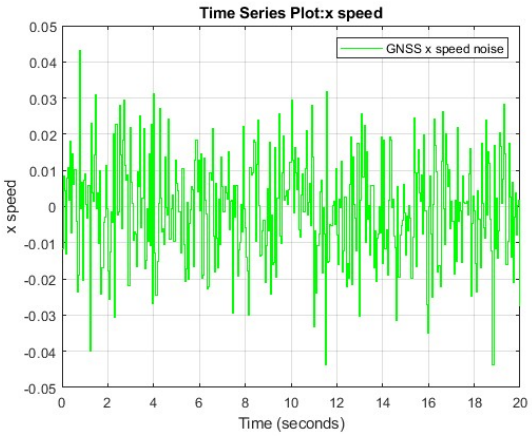
Band-limited white noise is a type of white noise restricted to a specific range of frequencies. White noise, in general, is a random signal with equal intensity across all frequencies, resulting in a constant power spectral density. However, in practical applications, it's often necessary to limit this noise to a specific frequency band. When noise is band-limited, it only contains frequencies within a certain range, while frequencies outside this range are filtered out or have zero power. This makes band-limited white noise more applicable in real-world systems, where infinite frequency ranges are impractical or unnecessary.

The GNSS is used to measure three states, these being the longitudinal position and both the lateral and longitudinal speed of the vehicle; the noise of the chosen GNSS is shown in Figure 3.4 for the x position and for the speeds that are the same for both planes (x and y)

For the Y position as said in Subsection 2.1.2 the method proposed by Ali and Hussein [10] was used and the relative noise was computed based on the data and results obtained by them; the results are shown in Figure 3.5.



(a) X position noise



(b) Speeds noises

Figure 3.4: GNSS noises

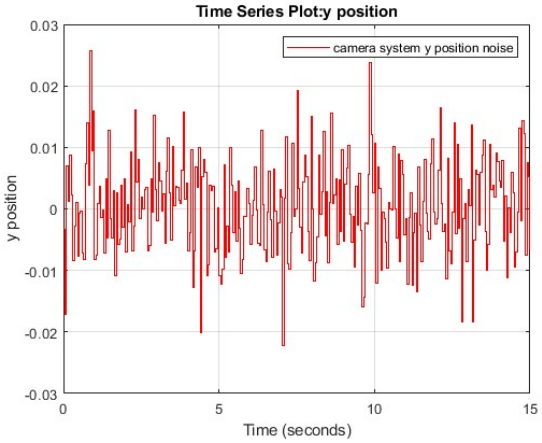


Figure 3.5: Vehicle position detection noise

Regarding the yaw angle the magnetometer employed presents a noise as shown in Figure 3.6.

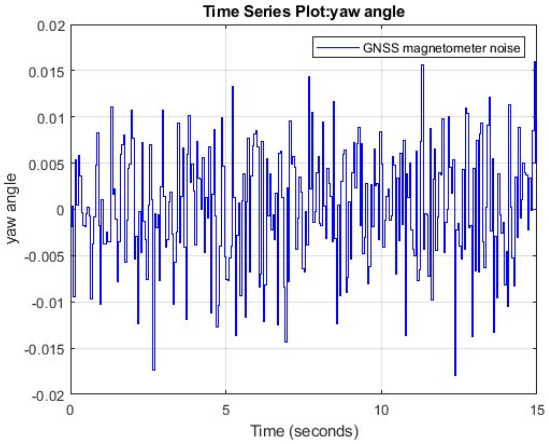


Figure 3.6: Magnetometer noise

Lastly the yaw rate measured by the gyroscope is characterized by a noise presented in Figure 3.7.

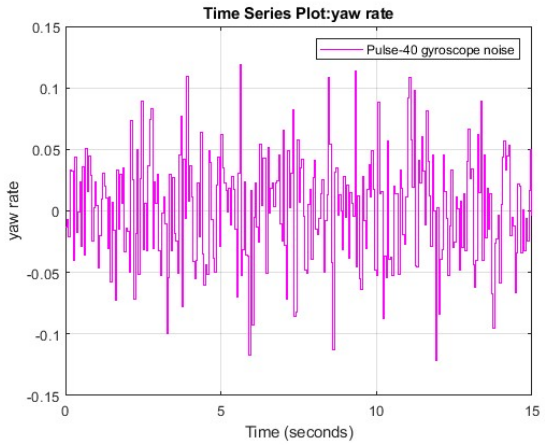


Figure 3.7: Gyroscope noise

3.2 Simulation results

In the following pages, the simulation results will be showcased. Additionally, a tuning procedure will be illustrated and explained ,and finally, the performance of the system will be evaluated.

3.2.1 Results with tuning parameters

Now that everything has been defined, it's time to focus on the results. Simulations were performed to evaluate the system's behavior. These simulations are similar to the previous ones, involving scenarios with wind and lane repositioning. Specifically:

- A 25 $\frac{m}{s}$ wind with an incidence angle of $\frac{\pi}{2}$ is applied at $t = 1\ s$ and continues till the end of the simulation.
- The vehicle reference y position changes moving from 0 to 2.5 m at $t = 5\ s$.

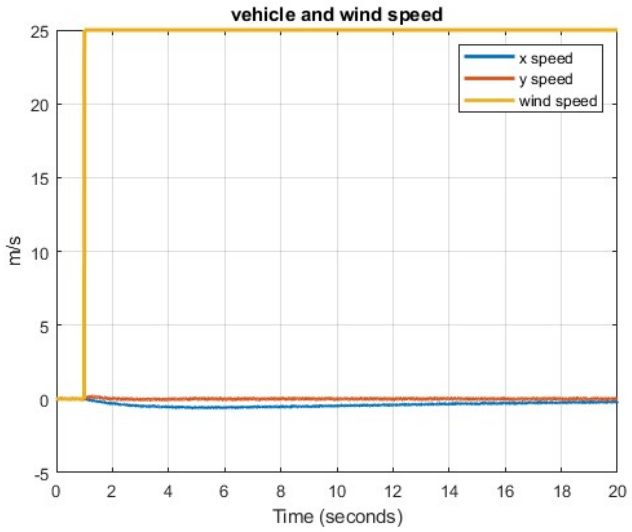
For the first simulation the results can be seen in the Figure 3.8.

The starting point is defined by the initial conditions shown in matrices 3.1.

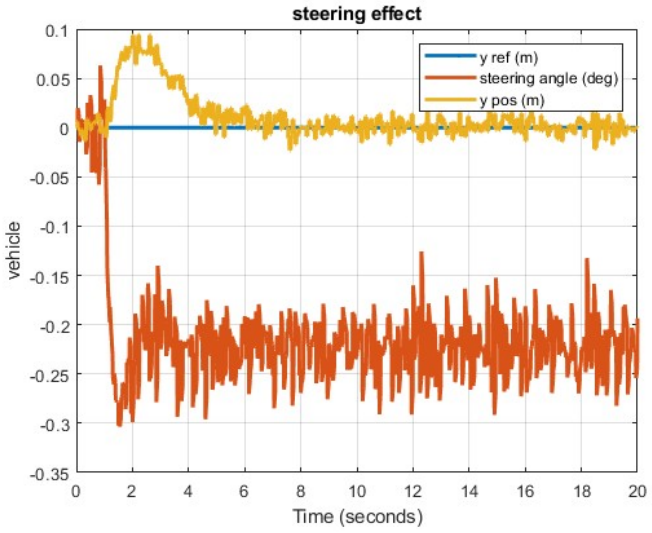
$$\mathbf{x_{init}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{u_{init}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.1}$$

From the graphs, it is evident that the wind was modeled as a step function, chosen for its aggressiveness. A step function is significant because, in signal theory, it encompasses a continuous set of frequencies rather than being limited to a single one. Therefore, if the system can reliably handle this type of signal, it should also perform well with any other signal type.

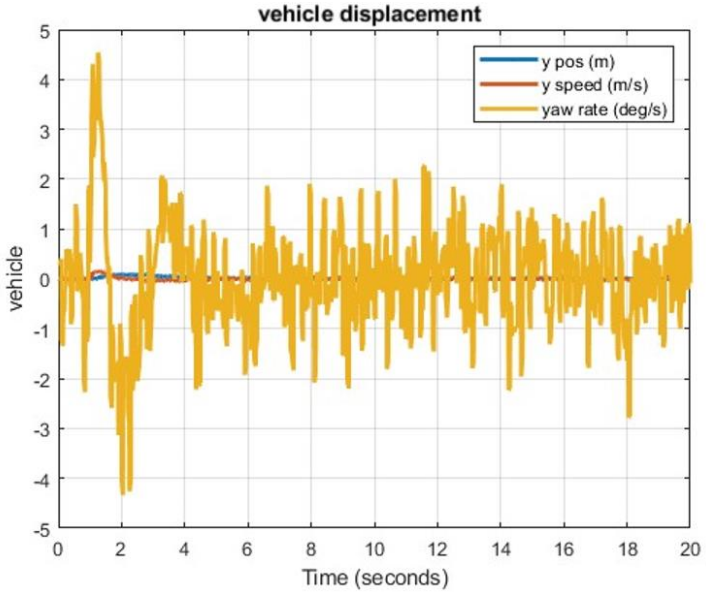
Analyzing the steering effect graph, it's clear that the vehicle successfully returns to the reference position within a few seconds after being impacted by the wind gust, although some oscillation around the reference are present due to noise. Regarding the steering angle, it starts to adjust immediately once the wind begins affecting the system and remains relatively constant since the wind continues to exert force on the vehicle. Although the steering angle appears to be influenced by noise, this is primarily due to the noise in the measured states; in fact, the steering angle itself is not directly affected by noise. In a real-world application, this issue could be easily mitigated with the use of a filter or other noise reduction techniques. Indeed, after applying a simple Gaussian filter with a 150-sample window, the results improved, as shown in Figure 3.9. This filtering was achieved using the MATLAB command `smoothdata`, which effectively reduced the noise in the steering angle response, leading to a smoother and more stable output.



(a) Speeds



(b) Steering angle



(c) General vehicle information

Figure 3.8: Results of the first simulation

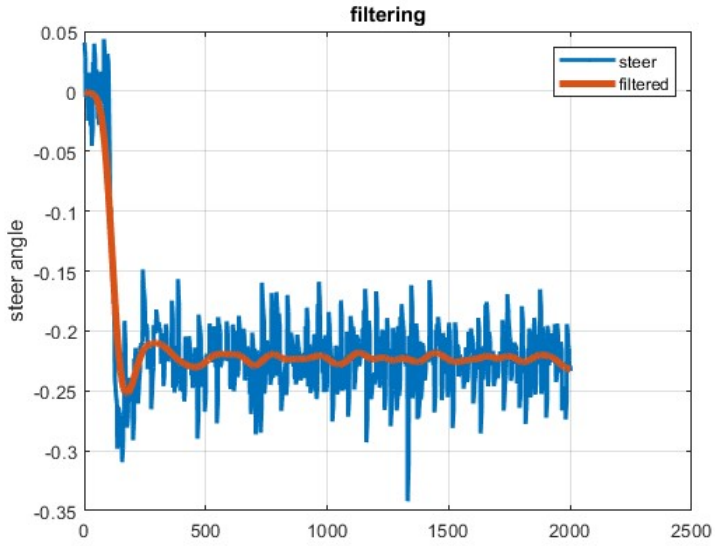


Figure 3.9: Filtered steering angle

It is evident that the yaw rate is significantly more affected by noise compared to the other sensors, as its noise power is an order of magnitude higher due to the sensor's characteristics. This issue was addressed by carefully tuning the LQR parameters, ensuring that the controller's performance remained effective despite the increased noise.

To achieve these results, a fine-tuning process was conducted on the LQR parameters. After several iterations, the values presented in Table 3.1 were determined. As there were no strict rules or principles to guide this process, a rule-of-thumb approach was employed based on expectations of the system's behavior, given the absence of reference guidelines.

qe_x_pos	1
qe_y_pos	15
qe_x_speed	1
qe_y_speed	5
qe_yaw_angle	1
qe_yaw_rate	1
qe_error_steer	10
re_steer	10

Table 3.1: Optimal Q and R values for the first simulation

Modifying these values led to significant changes in the system; in fact:

- qe_x_pos: Adjusting this parameter does not significantly impact the system's performance.
- qe_y_pos: Increasing this value decreases the Y position overshoot, resulting in less lateral displacement. However, the steering angle becomes more aggressive and exhibits increased oscillations. Conversely, decreasing this parameter leads to minor changes, such as slightly higher overshoots.
- qe_x_speed: Changes to this parameter do not result in any meaningful alterations to the system's behavior.
- qe_y_speed: Raising this value reduces fluctuations in Y speed, maintaining it nearly constant or close to zero. This stabilization leads to a smaller change in the Y position but requires more time to return to the reference value, as Y speed becomes more critical than Y position. The steering angle responds similarly to when qe_y_pos is increased. Lowering this value results in minor changes to Y speed, such as a slightly higher overshoot.
- qe_yaw_angle: Modifying this parameter does not have a significant impact on the system's behavior.

➢ qe_yaw_rate: Increasing this value results in a lower overshoot for the Y position but introduces more oscillations at steady-state due to the high noise sensitivity of the yaw rate. Decreasing this parameter, however, does not produce notable changes.

➢ qe_error_steer: Increasing this value causes a larger overshoot in the Y position and increases the steering angle's aggressiveness. Conversely, decreasing this value leads to the Y position either not returning to the reference or taking a longer time to do so.

➢ re_steer: Increasing this parameter causes a significant increase in the Y position overshoot and introduces steady-state oscillations, while the steering angle becomes less aggressive. Decreasing it, on the other hand, reduces the Y position overshoot, keeping it closer to the reference value, with the steering angle becoming more aggressive.

As demonstrated earlier, the controller could have been made significantly more aggressive. However, a more realistic behavior was maintained, allowing for better observation and analysis of the fluctuations in all variables reflecting the controller's operation.

Based on this evaluation, a test was conducted to validate these assumptions. The objective was to design a gentler controller with a higher overshoot that would approach the reference more smoothly. This was achieved by decreasing *qe_error_steer* and increasing *re_steer*, which reduced the emphasis on the position error and limited actuator usage (since *re_steer* is a cost function). The new values are presented in Table 3.2. The results of this test are shown in Figure 3.10, where significant differences in the Y position are evident. However, the effects on the steering angle are less pronounced due to noise overlap and the small differences in values.

qe_x_pos	1
qe_y_pos	15
qe_x_speed	1
qe_y_speed	5
qe_yaw_angle	1
qe_yaw_rate	1
qe_error_steer	1
re_steer	35

Table 3.2: Q and R values for the test

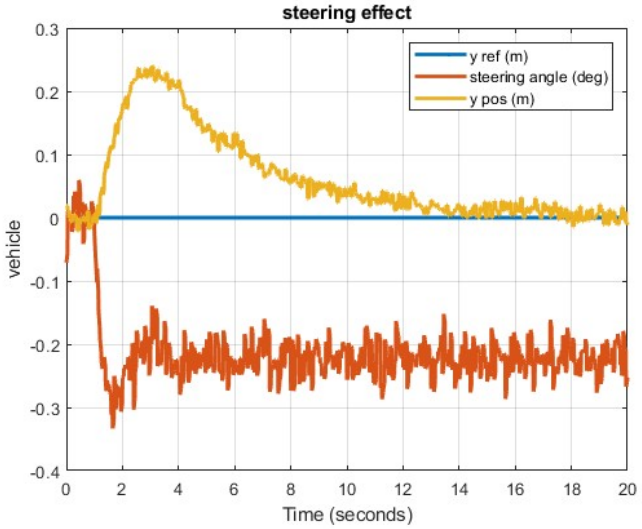
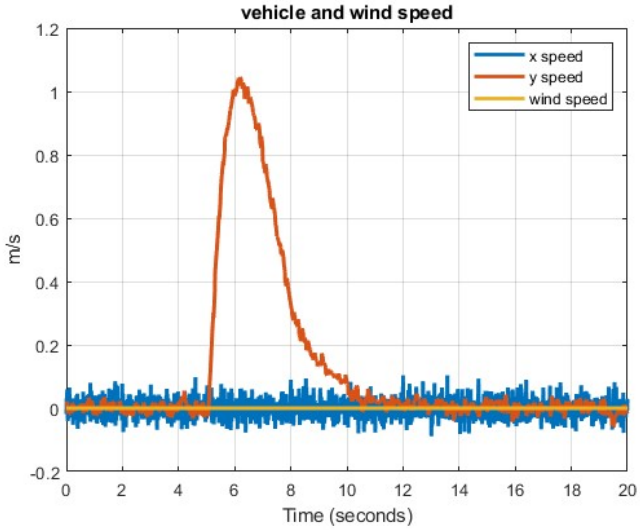
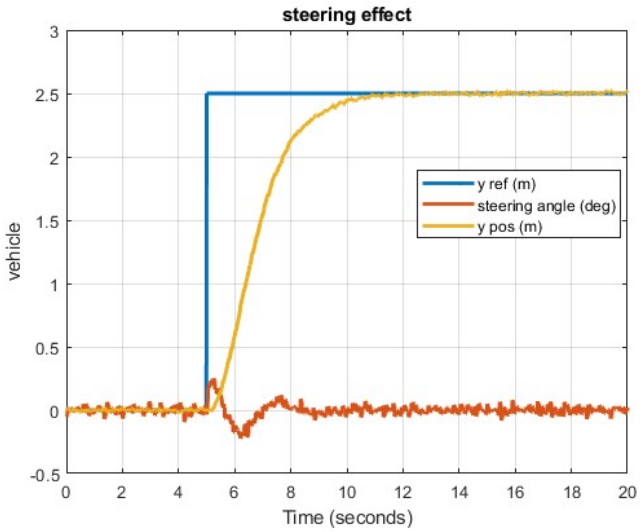


Figure 3.10: Test with different parameters

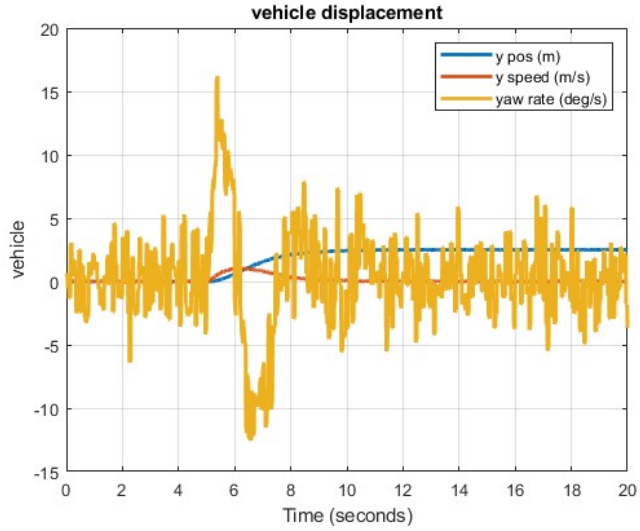
For the second simulation the results can be seen in Figure 3.11, while the starting condition are shown in matrices 3.2



(a) Speeds



(b) Steering angle



(c) General vehicle information

Figure 3.11: Results of the second simulation

$$\mathbf{x_{init}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{u_{init}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.2}$$

In this scenario, the wind is completely absent. The system, initially in an idle state, needs to return to the reference Y position once it is altered. This scenario closely mirrors a lane change maneuver. The primary focus here is to evaluate the system's behavior when only the lateral position error is corrected, without the influence of external disturbances such as wind. This setup provides a clearer view of the system's inherent dynamics and control efficiency, allowing to observe how effectively the system handles the correction of lateral position errors in a controlled environment.

From the speed graph, it can be observed that the vehicle primarily adjusts its lateral speed as it changes orientation during the maneuver. In the steering effect graph, it's noteworthy that the steering angle starts to adjust right when the reference Y position changes, which, in this case, happens at 5 seconds. The Y reference position change is modeled as a step function, for the same reason as the previous case. The steering angle begins with positive values, gradually decreases to negative values and then oscillates slightly before stabilizing near zero in steady-state. Correspondingly, the Y position adjusts as soon as the steering angle changes, and within a few seconds, the vehicle successfully returns to the desired position.

Regarding the yaw rate graph, high oscillations occur at the start of the maneuver. However, after the vehicle returns to the reference Y position, the yaw rate begins to stabilize, oscillating around zero until it nearly stabilizes, with only minor fluctuations caused by sensor noise. If a Gaussian filter is applied, similar to the previous case with the steering angle, the mean value of the yaw rate becomes quite precise and close to zero, as shown in Figure 3.12. This suggests that the system effectively stabilizes after the reference correction, even in the presence of noise.

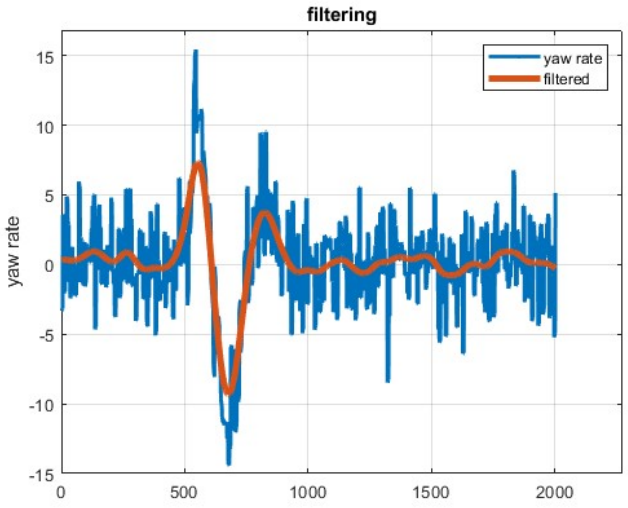


Figure 3.12: Filtered yaw rate

The controller is designed to handle lane change tasks accurately in both directions without needing adjustments to the linearization point or parameters. For brevity, tests under both conditions have not been reported.

In this case, contrary to the expectations, fine-tuning was not necessary. In fact, the LQR parameters used were the same as those shown in Table 3.1, as they delivered satisfactory performance for the lane change task. As before, given the absence of reference values or established procedures, the quality of the results was evaluated based on the authors' expectations.

Similarly to the previous study, a worst-case analysis was performed using the same LQR parameters shown in Table 3.2. As anticipated, this resulted in an increased response time and reduced actuator

usage. These expectations were confirmed, as illustrated in Figure 3.13.

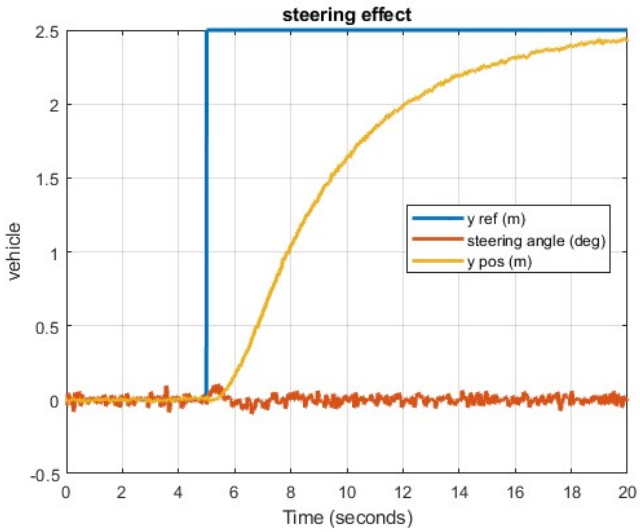


Figure 3.13: Test with different parameters

It can be observed that, due to the reduced use of the steering angle, its values span a smaller range. The Y position demonstrates a smoother behavior, but it is unable to fully reach the reference Y position within the fixed time span. These results support the earlier assumptions about both the reduced actuator utilization and the smoother response of the controller. This indicates that, while the controller is more gentle and less aggressive, it also requires too much time to achieve the desired positional accuracy.

3.2.2 Model scheduling

In the previous section the primary issue emerged: the need to adjust the linearization point during the crosswind stabilization task. This adjustment was required due to specific formulas in the physical model. Ideally, the disturbance linearization point should consist solely of zeros, rather than the values shown in matrices 2.50. The system performs correctly for wind directions within the semi-plane defined by the linearization point, which separates the positive and negative semi-planes with respect to the x-axis. However, for wind directions in the opposite semi-plane relative to the linearization point, the system does not work properly. Notably, this issue does not occur in lane-changing applications, indicating that the problem is specific to the presence of wind. In this section this problem is addressed through the introduction of a new technique called model scheduling or gain scheduling.

Gain scheduling is a control strategy used for systems with nonlinear dynamics by designing different controllers for different operating points, where the system can be approximated as linear. The idea is to linearize the system around various operating conditions and design individual linear controllers for each point. As the system operates and transitions between these conditions, the control strategy adapts by either switching between different controllers or interpolating among them, based on the current operating state, which is represented by a scheduling variable. This allows the control system to adapt to the nonlinear nature of the system while maintaining stability and performance across different linearization regions.

In this task, two controllers were developed, as illustrated in Figure 3.14. Both controllers share a similar structure, shown in Figure 3.15. The only difference between them lies in the point at which they are linearized, corresponding to the two directions of wind arrival ($\frac{\pi}{2}$ and $-\frac{\pi}{2}$). This variation affects the linearization matrices, specifically the A and B_2 matrices. Consequently, in the one with direction of arrival $-\frac{\pi}{2}$, the matrices are updated from the forms presented in matrix 2.51 and matrix 2.53 to those shown in matrix 3.3 and matrix 3.4, while in the other one they remains the same.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.4894 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5694 & 15.9084 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -0.0010 & 0.0270 & 0.0323 & -1.2844 \end{bmatrix} \tag{3.3}$$

$$B_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2582 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0669 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0270 & -0.0261 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

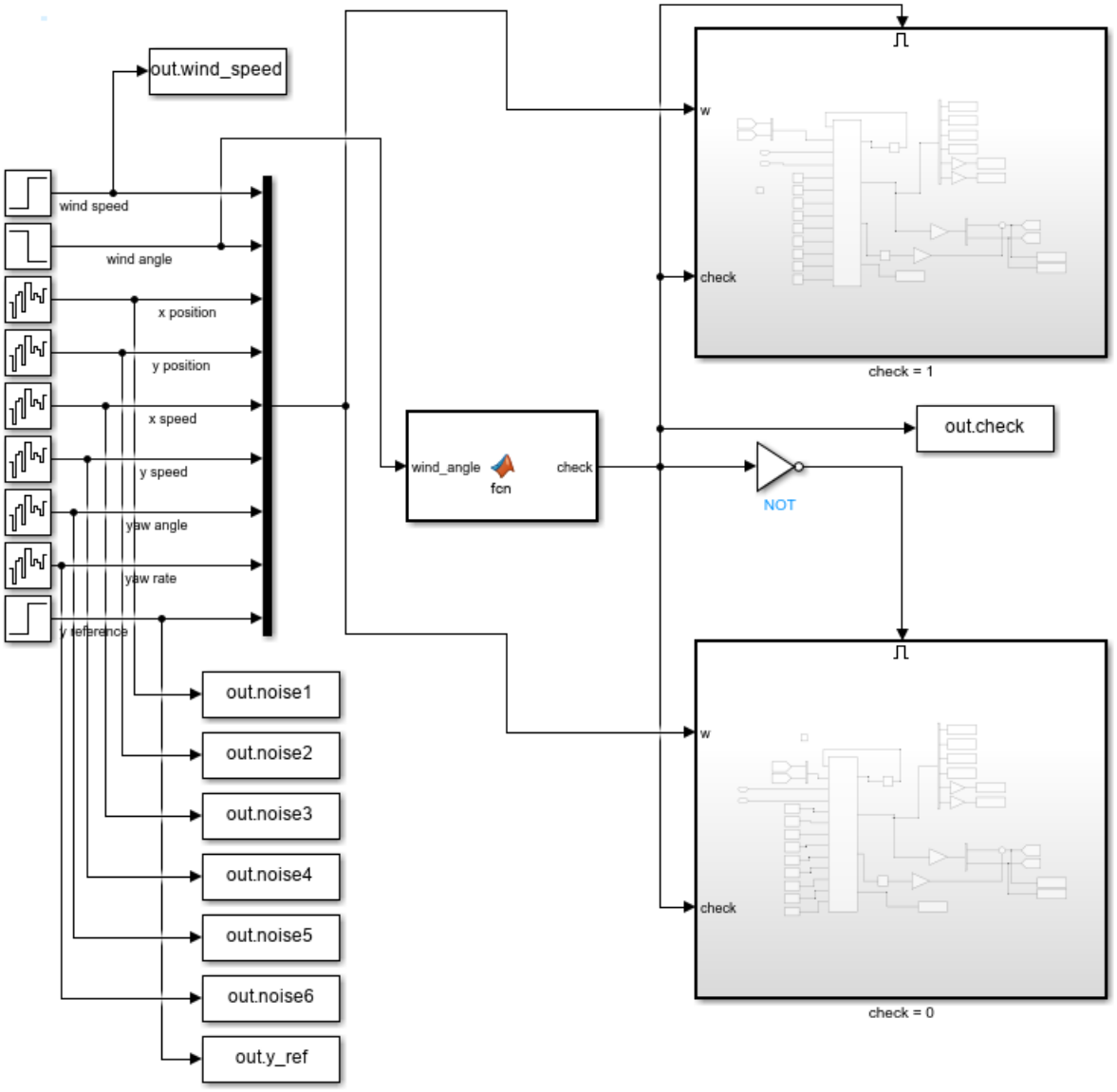


Figure 3.14: Project scheme with model scheduling

To validate this approach two simulation, differing only in the wind direction of arrival, were performed:

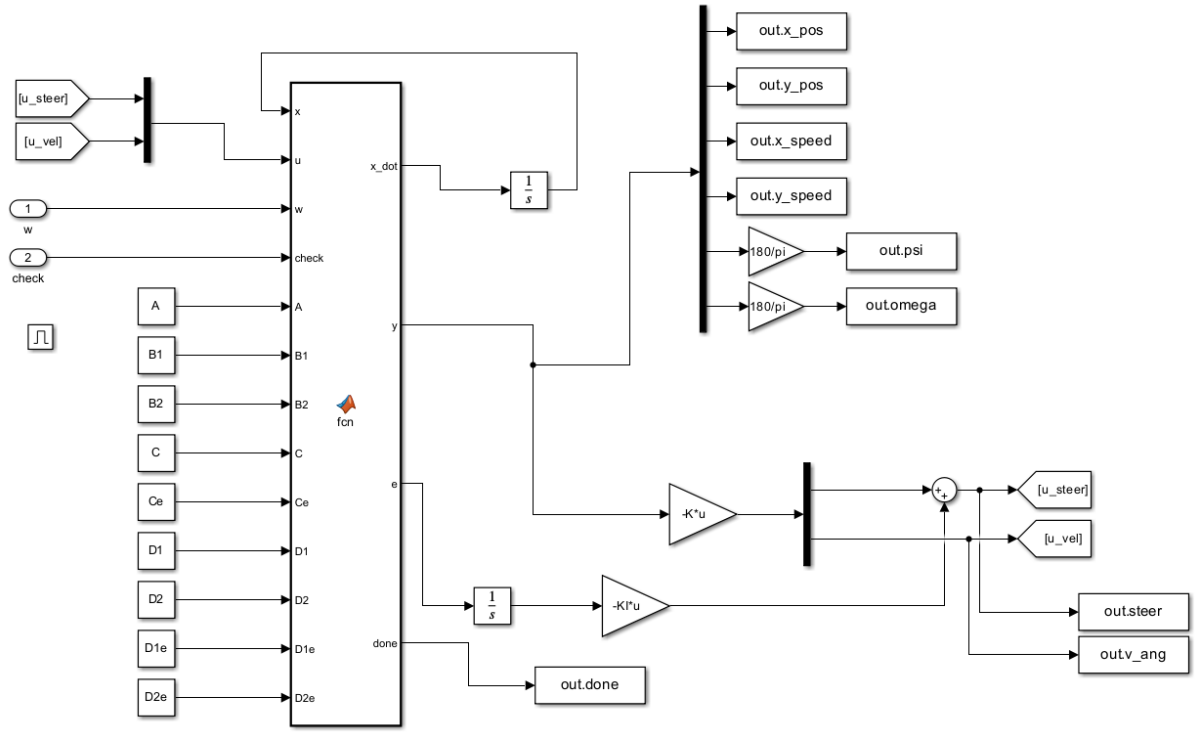


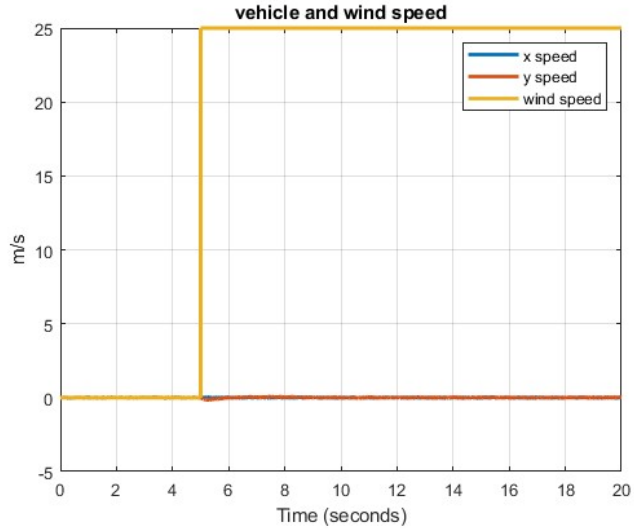
Figure 3.15: Controller architecture in model scheduling

- A 25 $\frac{m}{s}$ wind with an incidence angle of $\frac{\pi}{2}$ is applied at $t = 5$ and continues till the end of the simulation.
- A 25 $\frac{m}{s}$ wind with an incidence angle of $-\frac{\pi}{2}$ is applied at $t = 5$ and continues till the end of the simulation.

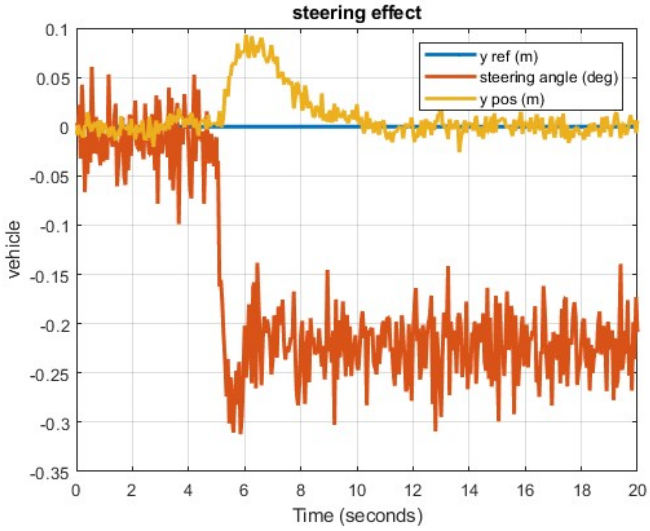
The results of these simulations are shown in Figure 3.16 and Figure 3.17 respectively.

This approach successfully addresses the previous issue concerning wind direction by providing a system that performs effectively in all situations. The responses resemble those in Figure 3.8, as the same LQR parameters and similar simulation settings were used. The two simulations are symmetric with respect to the Y position: in one case, the Y position changes in the positive direction, while in the other, it changes in the negative direction. A similar pattern can be observed with the steering angle, which adjusts oppositely depending on the Y position's direction. This technique allows for differentiating the vehicle's behavior based on wind direction, enabling the design of distinct controllers tailored to react differently depending on the direction of deviation caused by incoming wind.

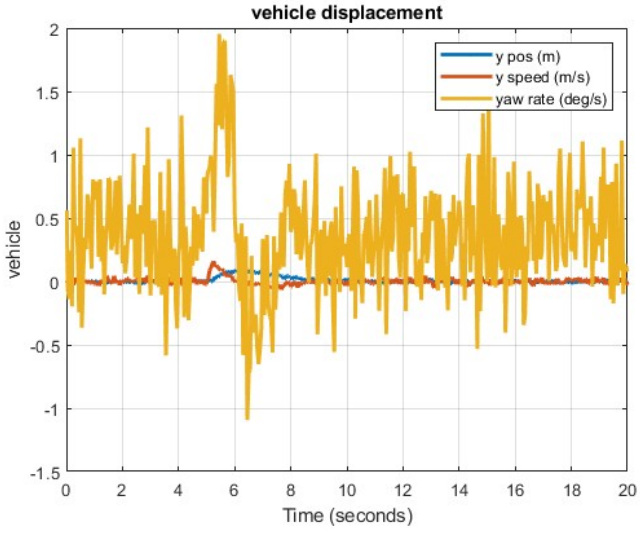
As it could be expected changes in the LQR parameters will affect the system's behaviour as described in Subsection 3.2.1.



(a) Speeds

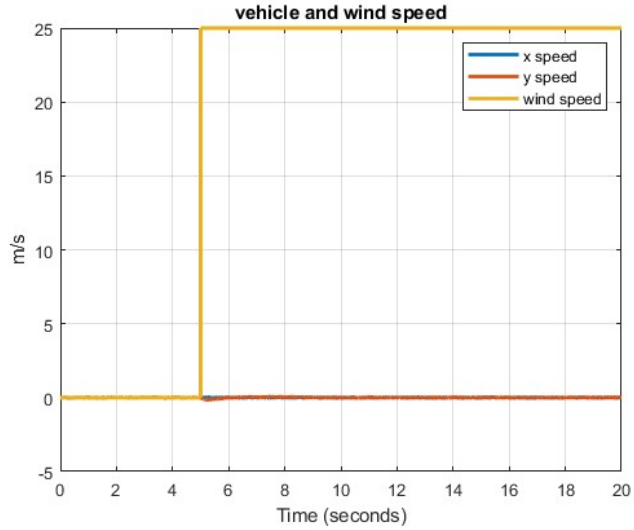


(b) Steering angle

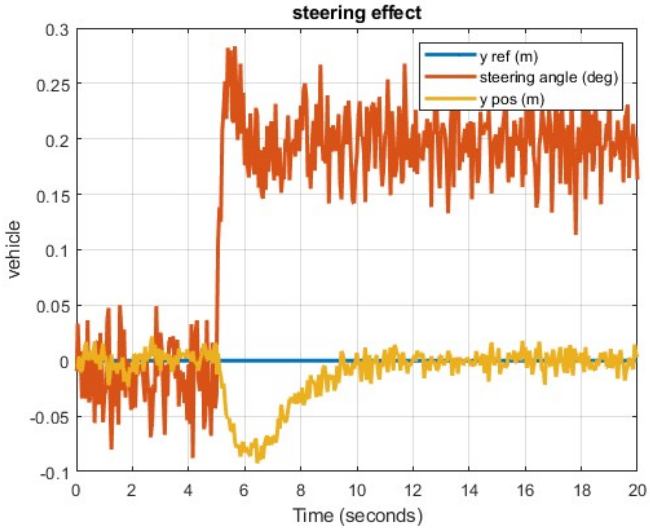


(c) General vehicle information

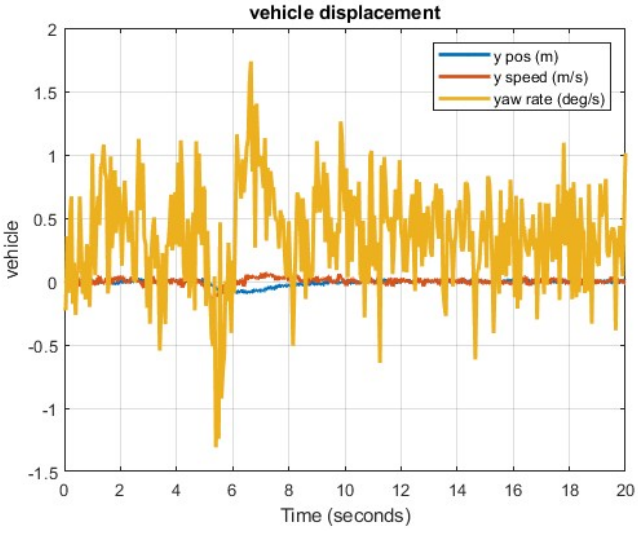
Figure 3.16: Results of the first simulation with model scheduling



(a) Speeds



(b) Steering angle



(c) General vehicle information

Figure 3.17: Results of the second simulation with model scheduling

Chapter 4

Conclusion and further works

4.1 Discussion

Overall, while the control system developed by the group is not perfect, it exhibits solid performance. Nevertheless, there are critical areas that could benefit from further refinement or investigation.

One of the most challenging aspects was defining the physical model. Due to its inherent complexity, several simplifications and assumptions were necessary, such as focusing exclusively on yaw dynamics and dividing the friction forces between the front and rear wheels rather than considering all four wheels individually.

An additional examination could involve applying the LQR optimal control to the non-linear system to assess its effectiveness under more realistic conditions.

Possible improvements to the project could include modeling the steering actuator with a more realistic behavior by incorporating an action delay, so that the steering effect does not act immediately. Additionally, accounting for noise and uncertainties associated with the actuator could provide a more accurate representation of real-world conditions.

Some interesting avenues for further investigation could include examining how changes in model parameters, such as the road friction coefficient listed in Table 2.1, impact the overall system performance. Additionally, exploring variations in weight partitioning, wind friction coefficient or other parameters within the model might offer insights into different scenarios.

4.2 Conclusions

The aim of this project was to develop a crosswind stabilization control system for a heavy vehicle subjected to varying wind speeds and angles of incidence. To achieve this, control theory was applied to create a linearized version of the system around a selected operating point. A PI controller was then designed and fine-tuned to ensure optimal performance under these conditions. Additionally, the controller was adapted to handle lane-changing maneuvers, thereby expanding its practical applicability beyond crosswind stabilization.

The results achieved from the implementation of this PI controller are highly satisfactory. The controller demonstrated effective stabilization of the vehicle under crosswind conditions, showcasing the success of the fine-tuning process. The improvements in vehicle stability during both crosswind exposure and lane changes underscore the controller's robustness and reliability. These enhancements contribute significantly to road safety, benefiting not only the vehicle's driver but also other road users.

Overall, the project successfully met its objectives, delivering a well-functioning control system for the "Automatic Control" exam. The practical applications of the developed system, particularly in improving vehicle stability and safety under challenging conditions, highlight the value and effectiveness of the design and fine-tuning efforts undertaken.

Bibliography

[1] D. Jung and S. Kim, "A Novel Control Strategy of Crosswind Disturbance Compensation for Rack-Type Motor Driven Power Steering (R-MDPS) System", in *IEEE Access*, vol. 10, pp. 125148-125166, 2022, doi: 10.1109/ACCESS.2022.3225359.

[2] Se-Jin Kim, Chul-Hwan Yoo, Ho-Kyung Kim, "Vulnerability assessment for the hazards of crosswinds when vehicles cross a bridge deck", *Journal of Wind Engineering and Industrial Aerodynamics*, Volume 156, 2016, Pages 62-71, ISSN 0167-6105, `https://doi.org/10.1016/j.jweia.2016.07.005`.

[3] N. E. Kahveci, "Adaptive steering control for uncertain vehicle dynamics with crosswind effects and steering angle constraints", *2008 IEEE International Conference on Vehicular Electronics and Safety*, Columbus, OH, USA, 2008, pp. 168-173, doi: 10.1109/ICVES.2008.4640911.

[4] M. Hübner, T. Stork, U. Becker and E. Schnieder, "Lateral stabilization of vehicle-trailer combinations against crosswind disturbances by means of sliding control", *2008 16th Mediterranean Conference on Control and Automation*, Ajaccio, France, 2008, pp. 431-438, doi: 10.1109/MED.2008.4602075.

[5] G. Huang, X. Yuan and Y. Wang, "Predictive Compensation-Based Handling Stability Control Systems for Autonomous Vehicles under Transient Crosswind", *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, Hangzhou, China, 2020, pp. 662-667, doi: 10.1109/CVCI51460.2020.9338642.

[6] J. Zhang and Y. Xie, "Simulation and Control of Cross-Wind Stability of Vehicles Exiting and Entering Tunnels", *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, Nanchang, China, 2020, pp. 66-69, doi: 10.1109/ICECTT50890.2020.00022.

[7] Mercedes-Benz, 25^{th} of June 2014, *A Deep Dive into Mercedes-Benz Crosswind Stabilization Technology*, Mercedes-Benz of Shrewsbury, `https://www.wagnermercedesofshrewsbury.com/blog/2014/june/25/a-deep-dive-into-mercedesbenz-crosswind-stabilization-technology.htm`

[8] Zhao, YQ., Li, HQ., Lin, F. *et al.* 'Estimation of Road Friction Coefficient in Different Road Conditions Based on Vehicle Braking Dynamics'. *Chin. J. Mech. Eng.* 30, 982–990 (2017). `https://doi.org/10.1007/s10033-017-0143-z`

[9] Columbia River SHIP REPORT, (2020, October 12), "The Beaufort Wind Scale, Part 1", `https://shipreport.net/2020/10/12/the-beaufort-wind-scale-part-1/`

[10] A. A. Ali and H. A. Hussein, "Distance estimation and vehicle position detection based on monocular camera", *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, Baghdad, Iraq, 2016, pp. 1-4, doi: 10.1109/AIC-MITCSA.2016.7759904.

[11] Zanasi Roberto, "*Teoria dei Sistemi e del Controllo*", LM Ingegneria Informatica - Elettronica, University of Modena and Reggio-Emilia.

[12] SBG SYSTEMS, "Ellipse Series", 2023, [Online]. Available: `https://www.sbg-systems.com/wp-content/uploads/Ellipse_Series_Leaflet.pdf`

[13] SBG SYSTEMS, "Pulse-40 Inertial Measurement Unit", 2022, [Online]. Available: https://www.sbg-systems.com/wp-content/uploads/Pulse-40_IMU-Leaflet.pdf

[14] Mimmo Nicola, (2024), "Analysis and Design of Control Laws for Advanced Driver-Assistance Systems Theory and Applications", Springer International Publishing.

Codes

Symbolic Model

```
clc
close all
clear
% symbolic definitions
syms mass psi rho Sx Cf W csi g weight_part delta PbyI PbxI lx ly omega
syms Sy Cs VbxI VbyI C1 C2 C3 C4 epsilon v_ang wheel_rad
sign_approx = @(x) tanh(1000*x); % Sostituzione di sign con tanh
abs_approx = @(x) sqrt(x^2 + epsilon);
% plant formulas
Rbi = [cos(psi) sin(psi) 0;     % inertial-body
       -sin(psi) cos(psi) 0;
       0 0 1];
Rwb1 = [cos(delta) sin(delta) 0;  % body-wheel front wheels
       -sin(delta) cos(delta) 0;
       0 0 1];
Rwb3 = eye(3);  % wheel-body rear wheels

% calcolo dei mu
Pw1B = [lx ; ly ; 0];      % position vector of wheel i
Pw2B = [lx ; -ly ; 0];
Pw3B = [-lx ; -ly ; 0];
Pw4B = [-lx ; ly ; 0];
Pw1_2B = [lx; 0; 0];      % bycicle model positions
Pw3_4B = [-lx; 0; 0];
Rbi_der = [-sin(psi) cos(psi) 0;    % first derivative of Rbi
     -cos(psi) -sin(psi) 0;
           0 0 0];
VbI = [VbxI; VbyI; 0];  % intertial body velocity
Vw1I = VbI + Rbi_der' * Pw1_2B * omega; % wheel speed in intertial
Vw3I = VbI + Rbi_der' * Pw3_4B * omega;
Vw1W1 = Rwb1 * Rbi * Vw1I;    % wheel x speed in wheel
Vw3W3 = Rwb3 * Rbi * Vw3I;
% mu calculation

lambda1 = 0;
modulo1 = sqrt(Vw1W1(1)^2 + Vw1W1(2)^2);
beta1 = asin(Vw1W1(2)/modulo1);
mux1 = sign_approx(lambda1) * C1 * (1 - exp(-C2 * abs_approx(lambda1)))
    - C3 * lambda1;
muy1 = sign_approx(-beta1/(pi/2)) * C1 * (1 - exp(-C2 * abs_approx(-
   beta1/(pi/2)))) - C3 * (-beta1/(pi/2));
mux2 = mux1;
muy2 = muy1;
```

```
lambda3 = (v_ang * wheel_rad - Vw3W3(1)) / (epsilon + v_ang * wheel_rad
    );
modulo3 = sqrt(Vw3W3(1)^2 + Vw3W3(2)^2);
beta3 = asin(Vw3W3(2)/modulo3);
mux3 = sign_approx(lambda3) * C1 * (1 - exp(-C2 * abs_approx(lambda3)))
     - C3 * lambda3;
muy3 = sign_approx(-beta3/(pi/2)) * C1 * (1 - exp(-C2 * abs_approx(-
    beta3/(pi/2)))) - C3 * (-beta3/(pi/2));

lambda4 = (v_ang * wheel_rad - Vw3W3(1)) / (epsilon + v_ang * wheel_rad
    );
mux4 = sign_approx(lambda4) * C1 * (1 - exp(-C2 * abs_approx(lambda4)))
     - C3 * lambda4;
muy4 = muy3;
% normal forces
N1 = mass * g * weight_part * 0.5;
N2 = N1;
N3 = mass * g * (1 - weight_part) * 0.5;
N4 = N3;
% force on wheels in wheel coordinates
Fw1xW1 = N1 * mux1;
Fw1yW1 = N1 * muy1;
Fw1W1 = [Fw1xW1; Fw1yW1; 0];
Fw2xW2 = N2 * mux2;
Fw2yW2 = N2 * muy2;
Fw2W2 = [Fw2xW2; Fw2yW2; 0];
Fw3xW3 = N3 * mux3;
Fw3yW3 = N3 * muy3;
Fw3W3 = [Fw3xW3; Fw3yW3; 0];
Fw4xW4 = N4 * mux4;   % wheel x forces
Fw4yW4 = N4 * muy4;   % wheel y forces
Fw4W4 = [Fw4xW4; Fw4yW4; 0];  % matrices of forces on each wheel
% force due to wind disturbance
WxI = abs_approx(W) * cos(csi);
WyI = abs_approx(W) * sin(csi);
WI = [WxI; WyI; 0];  % inertial wind speed
VaI = VbI + WI;  % intertial total velocity
FaxI = 0.5 * rho * Sx * Cf * sign_approx(VaI(1)) * VaI(1)^2; %
    aerodynamic x forces body
FayI = 0.5 * rho * Sy * Cs * sign_approx(VaI(2)) * VaI(2)^2; %
    aerodynamic y forces body
FaI = [FaxI; FayI; 0];  % total forces in inertial coord
FaB = Rbi * FaI;        % total forces in body coord
FwB = Rwb1.' *(Fw1W1) + Rwb1.' *(Fw2W2) + Rwb3.' *(Fw3W3) + Rwb3.' *(
    Fw4W4);  % wheel forces in body coord
FwI = Rbi.' * FwB;  % wheel forces in inertial coord
abI = (FaI + FwI)/mass;     % x and y total acc in inertial coord
% yaw moment
IzB = 275458;       % yaw moment of inertia
laB = [lx; ly; 0];      % vector of body lenghts
MaB = cross(laB, FaB);   % aerodynamic torque on body
Mw1B = cross(Pw1B, (Rwb1.')*Fw1W1);   % torque on wheel i on body
Mw2B = cross(Pw2B, (Rwb1.')*Fw2W2);
Mw3B = cross(Pw3B, (Rwb3.')*Fw3W3);
Mw4B = cross(Pw4B, (Rwb3.')*Fw4W4);
a_psi = (Mw1B + Mw2B + Mw3B + Mw4B + MaB)/ IzB; % acceleration due to
    inertial
% SYMBOLIC MODEL
```

```
y = sym('y', [6 1]);
r = sym('r', [1 1]);
nu = sym('nu', [6 1]);
x = [PbxI;     % x pos
     PbyI;     % y pos
     VbI(1);     % x speed
     VbI(2);       % y speed
     psi;       % yaw angle
     omega];  % yaw rate
u = [delta;    % steering angle
     v_ang];    % wheel angular velocity
d = [W;      % wind speed
     csi];     % wind incident angle
w = [d; nu; r];
f = [x(3);
     x(4);
     abI(1);
     abI(2);
     x(6);
     a_psi(3)];
h = [x(1) + nu(1); % GPS x position
      x(2) + nu(2); % visual system y position
      x(3) + nu(3); % GPS x speed
      x(4) + nu(4); % accelerometer/depth sensor y speed
      x(5) + nu(5); % magnetic angular position sensor yaw angle
      x(6) + nu(6)]; % gyroscope yaw rate
he = PbyI - r(1);

% matrices creation
A = jacobian(f, x);
B1 = jacobian(f, u);
B2 = jacobian(f, w);
C = jacobian(h, x);
D1 = jacobian(h, u);
D2 = jacobian(h, w);
Ce = jacobian(he, x);
D1e = jacobian(he, u);
D2e = jacobian(he, w);
matlabFunction(A, 'File', 'Amatrix')
matlabFunction(B1, 'File', 'B1matrix')
matlabFunction(B2, 'File', 'B2matrix')
matlabFunction(C, 'File', 'Cmatrix')
matlabFunction(D1, 'File', 'D1matrix')
matlabFunction(D2, 'File', 'D2matrix')
matlabFunction(Ce, 'File', 'Cematrix')
matlabFunction(D1e, 'File', 'D1ematrix')
matlabFunction(D2e, 'File', 'D2ematrix')
```

Model Main

```
clc
close all
clear
% RUN symbolic_model_v8 BEFORE THIS ONE
% simulation params
simtime = 20; % [s] simulation time span
DT = 1e-3; % [s] sample time fixed-step integration
SELECTOR = 0; % 1 = non-linearized, 0 = linearized
% vector dimensions
n = 6; % state vector -> x
p = 2; % control vector -> u
q = 6; % measurement vector -> y, nu
m = 1; % regulated output vector -> r
nd = 2; % disturbances vector -> d
r = 9; % exogenous vector -> w
% plant params for bus case
C1 = 1.2801; % [-] -> road fricition coefficients
C2 = 23.990;
C3 = 0.5200;
Cf = 1; % [-] -> aerodynamic drag coefficient front
Cs = 1.35; % [-] -> aerodynamic drag coefficient side
Sx = 7.5; % [m^2] -> longitunal projection area of the bus
Sy = 36; % [m^2] -> lateral projection area of the bus
g = 9.81; % [m/s^2] -> acceleration of gravity
lx = 5; % [m] -> distance from cg to wheel x coord
ly = 1.25; % [m] -> distance from cg to wheel y coord
mass = 22240; % [kg] -> mass of the bus
rho = 1.225; % [kg/m^3] -> air density
weight_part = 0.5; % [-] -> bus weight partition
epsilon = 1e-8; % [-] -> non 0/0 coeff
wheel_rad = 0.25; % [m] -> radius bus wheel
% linearization point - star
x0 = [0;                  % x trajectory
      0;                  % y pos
      25;                % x speed
      0;                  % y speed
      0;                  % yaw angle
      0];                 % yaw rate
u0 = [0; 100]; % control input as steering angle and v_ang
d0 = [25; pi/2]; % Linearization disturbance [wind]
r0 = 0; % reference y position in the middle of the road and wheel
   angular speed
nu0 = [0; 0; 0; 0; 0; 0]; % errors
y0 = [x0(1); x0(2); x0(3); x0(4); x0(5); x0(6)];
e0 = 0;
w0 = [d0; nu0; r0];
% LINEAR PLANT
A = Amatrix(C1, C2, C3, Cf, Cs, Sx, Sy, x0(3), x0(4), w0(1), w0(2), u0
   (1), epsilon, g, lx, ly, mass, x0(6), x0(5), rho, u0(2), weight_part
   , wheel_rad);
B1 = B1matrix(C1, C2, C3, x0(3), x0(4), u0(1), epsilon, g, lx, mass, x0
   (6), x0(5), u0(2), weight_part, wheel_rad);
B2 = B2matrix(Cf, Cs, Sx, Sy, x0(3), x0(4), w0(1), w0(2), epsilon, lx,
   ly, mass, x0(5), rho);
```

```
C = Cmatrix;
D1 = D1matrix;
D2 = D2matrix;
Ce = Cematrix;
D1e = D1ematrix;
D2e = D2ematrix;
B = [B1 B2];
D = [D1 D2];
De = [D1e D2e];
disp('end section 1')
%% observability & reachability
% observability
O = obsv(A, C);
n_o = length(A);
if rank(O) == n_o
disp('(A,C) FULLY OBSERVABLE')
else
disp('(A,C) NOT FULLY OBSERVABLE')
end
% controllability
Reach = ctrb(A, B1); % Reach is the reachabilty matrix
lenA = length(A);
if rank(Reach) == lenA
disp('(A,B1) FULLY REACHABLE')
else
disp('(A,B1) NOT FULLY REACHABLE')
end
disp ('end section 2')
%% design of integral action
%
Aext = [A zeros(6, 1); % extended matrix A, multiplied by [dot_x,
   dot_eta]
        Ce zeros(1 ,1)];
B1ext = [B1;            % extended matrix B1, multiplied by u
         D1e];
Cext = Ce;
Dext = D1e;
qe_x_pos = 1;
qe_y_pos = 15;
qe_x_speed = 1;
qe_y_speed = 5;
qe_yaw_angle = 1;
qe_yaw_rate = 1;
qe_error_steer = 10;
re_steer = 10;
Qe = diag([qe_x_pos qe_y_pos qe_x_speed qe_y_speed qe_yaw_angle
   qe_yaw_rate qe_error_steer]);
Re = re_steer;
% call the LQR for the augmented system
[Ke, ~, poles] = lqr(Aext, B1ext, Qe, Re);
% Split Ke into K and Ki
K = Ke(:, 1:lenA);
KI = Ke(1, lenA+1:end);
disp(poles)
disp('end section 3')
%}
%% simulink and plot
% simulation initial conditions
```

```
x_init = [0;            % x pos
          2.5;            % y pos
          0;           % x speed
          0;            % y speed
          0;            % yaw angle
          0];           % yaw rate
u_init = [0; 0]; % control input as steering angle and v_ang
d_init = [0; 0]; % Linearization disturbance [wind]
r_init = 0; % reference y position in the middle of the road and v_ang
nu_init = [0; 0; 0; 0; 0; 0]; % errors
w_init = [d_init; nu_init; r_init];
y_init = [x_init(1); x_init(2); x_init(3); x_init(4); x_init(5); x_init
   (6)];
e_init = y_init(2) - r_init(1);
%}
%disp("end section 4")
%
close all
%out = sim('test_sim_linear_control_integral.slx', simtime);
out = sim('test_sim_linear_control_integral_noise.slx', simtime);
```

A tribute to all the coffee drank by the authors during the development of this project.