UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Mecànica

MODELLING OF A MULTI-LANE ROAD FOR THE SIMULATION OF AUTONOMOUS VEHICLES AND HETEROGENOUS FLEET OF VEHICLES

Memòria i Annexos

Autor: Francesc Vilaró Antunes Dos Santos

Director: Victor Repecho Del Corral

Co-Director: Arnau Dòria Cerezo

Convocatòria: Gener 2020

Resum

L'objectiu d'aquest projecte és la simulació d'una carretera europea (carretera asimètrica), mitjançant els programes Matlab i Simulink.

Es dissenyarà una secció típica d'autopista formada per corbes de Bézier, formant un conjunt compost per una autopista de diversos carrils junt amb una entrada i una sortida

Per al seguiment de vehicle, s'utilitzarà el model I-ACC ("Ideal-Adaptive Cruise Control"). Per a la política de canvi de carril, s'utilitzarà un model simplificat del MOBIL ("Minimizing Overall Braking Induced by Lane Changes "), que té en compte els guanys i pèrdues generals de velocitat dels vehicles implicats. A més, per garantir un canvi de carril suau i realista, es proposa una funció de canvi de carril original.

Com a resultat, el projecte generarà animacions i diverses gràfiques per facilitar l'observació de la variació de diversos paràmetres, com ara la velocitat i posició a la carretera. L'animació gràfica estarà formada per diverses vistes simultànies de la carretera i dues gràfiques dinàmiques per mostrar variacions de velocitat i posició a la carretera en temps real. També s'analitzaran canvis en variables típiques com el flux de trànsit o la densitat de trànsit.

El programa modular servirà de plataforma per a futures modificacions, com la incorporació d'altres models de comportament, que requereixin canvis en l'estructura original.

Resumen

El objetivo de este proyecto es la simulación de una carretera europea (carretera asimétrica), mediante los programas Matlab y Simulink.

Se diseñará una sección típica de autopista formada por curvas de Bézier, formando un conjunto compuesto por una autopista de varios carriles junto con una entrada y una salida

Para el seguimiento de vehículo, se utilizará el modelo Y-ACC ("Ideal-Adaptive Cruise Control"). Para la política de cambio de carril, se utilizará un modelo simplificado del MOBIL ("Minimizing Overall Braking Induced by Lane Changes"), que tiene en cuenta las ganancias y pérdidas generales de velocidad de los vehículos implicados. Además, para garantizar un cambio de carril suave y realista, se propone una función de cambio de carril original.

Como resultado, el proyecto generará animaciones y varias gráficas para facilitar la observación de la variación de varios parámetros, como por ejemplo la velocidad y posición en la carretera. La animación gráfica estará formada por varias vistas simultáneas de la carretera y dos gráficas dinámicas para mostrar variaciones de velocidad y posición en la carretera en tiempo real. También se analizarán cambios en variables típicas como el flujo de tráfico o la densidad de tráfico.

El programa modular servirá de plataforma para futuras modificaciones, como la incorporación otros modelos de comportamiento, que requieran cambios en la estructura original.

Abstract

The aim of this project is the simulation of a European highway (asymmetric highway), using Matlab and Simulink software.

A typical freeway section consisting of Bézier curves will be designed, forming a set composed by a multi-lane motorway together with an on-ramp and an off-ramp.

For the car following, the I-ACC ("Ideal-Adaptive Cruise Control") model will be used. A simplified model of the MOBIL ("Minimizing Overall Braking Induced by Lane Changes"), which takes into account the general gain and loss of speed of the vehicles involved in it, will be used for the lane-changing policy. Also, to ensure a smooth and realistic lane change, an original lane change function has been proposed.

As a result, the project will generate animations and several plots to facilitate the impact of the variation of several parameters involved. The graphical animation will be formed by various simultaneous views of the highway and two dynamical plots to show variations in speed and position on the highway in real time. Changes in typical variables like traffic flow or traffic density will also be analysed.

The modular program will serve as a platform for future modifications, like the incorporation of other car following models, that require changes on the original structure.

Acknowledgement

To Arnau Doria Cerezo, for the patience and availability.

Also, to my parents and Laura.

Index

Preface

Context and motivation

We currently live in a world of transition in many aspects of life, mainly thanks to the massive technological development of the last century. Science and engineering have redefined some human activities so much, that our recent ancestors would find it hard to believe that we belong to the same species as them.

One of the fields that has undergone the biggest change is the one related to the transportation of goods and people. Specifically, land vehicles like cars, the subject of this project, have become increasingly fast and efficient, being able to travel further distances through the vast worldwide system of roads and highways.

But this has been done always with a person in charge of the motion of the vehicle, the driver. If we want to envision an even more secure, fast and efficient driving system, autonomous vehicles, this is, driverless vehicles, are the next logical step. Being a development upon which many lives would depend on, it must be studied thoroughly, predicting and calculating every possible outcome before its complete integration in normal everyday traffic.

Therefore, this project pretends to provide a computational simulation platform for future studies that pretend to evaluate the feasibility of the coexistence of the different existing autonomous driving models with the normal human traffic.

This program will serve as an alternative for already existing software like SUMO or AIMSUM. These programs, although more complex, have low flexibility and demand a lot of programming for small changes. A simple, highly modifiable software that allows an easy implementation of any kind of behavioural model is then needed, as an alternative.

Original project

This Bachelor´s thesis falls within the framework of a collaboration with the *Institut d'Organització i Control de Sistemes Industrials* (IOC), institution dedicated, among other things, to the mathematical modelling and control of autonomous vehicles.

1. Introduction

Objective and scope of the project

The aim of this project is to provide a simulation tool for future traffic studies that pretend to evaluate the impact of the introduction of autonomous driving in the current human driven traffic.

The program will be done using MATLAB and Simulink software and will be easily modifiable through the use of a modular structure.

Through the simulation of a 1 km multi-lane highway section together with an on-ramp and an off-ramp, vehicles will circulate following several behavioural models that will be proposed for the different types of decisions a driver has to take while driving: adjusting speed to accommodate to the current traffic situation, lane changing, route selection… Nevertheless, these models will be highly expansible and modifiable without the need of changing the model structure to implement them.

A graphical output will also be provided, consisting in some informational live plots and 2D animation.

The following milestones are then defined:

- Creation of the geometry of 1 km long highway section with an on-ramp and off-ramp.
- Implementation of a line following protocol.
- Implementation of a car following protocol.
- Implementation of a lane change model protocol.
- Implementation of a lane change function protocol.
- Graphical output in the form of live plots and 2D animation

Previous projects

To decide how to approach the objective of this project it has been taken as frame as reference other degrees final thesis. Those are:

- *Energy efficiency comparison between human drivers and adaptive cruise control system* (Marc Fernandez, 2017): consists on creating a mathematical model that has been developed with the aim of studying how vehicles behave in different traffic scenarios, not only in fluidity terms but also in energetic consumption.

- *Simulació de la fluència del trànsit de vehicles comparant el comportament humà i l'impacte de vehicles autònoms* (Jaume Cartro,2017): develops a program capable of modelling and simulating the traffic flows in non-urban roads, and study how traffic jams are created.

- *Influence of the Cooperative Adaptive Cruise Control to the traffic flow* (Bernat Bosch, 2017): investigates the influence of the Cooperative Adaptive Cruise Contruol (CACC) to the traffic flow.

- *Modelling and simulation of autonomous vehicles in a multi-lane and heterogeneuous traffic road* (Marta Miranda Elías, 2019): Develops the mathematical modelling and control of a circuit with multiple lanes.

2. Model

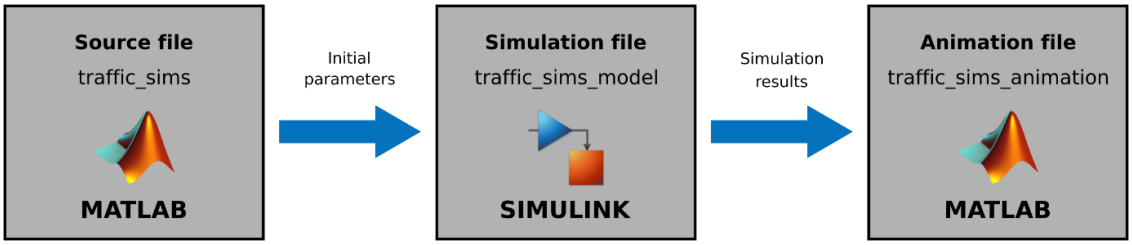


Figure 1 - General structure of the program

The structure of the model is as seen in figure 1. A MATLAB source file (*traffic_sims.m*) sends a set of initial parameters to a Simulink file (*traffic_sims_model.slx*), where the simulation is calculated. Afterwards, the results are sent to a MATLAB animation file (*traffic_sims_animation.m*), where a graphical output is produced.

2.1. Initial data generation

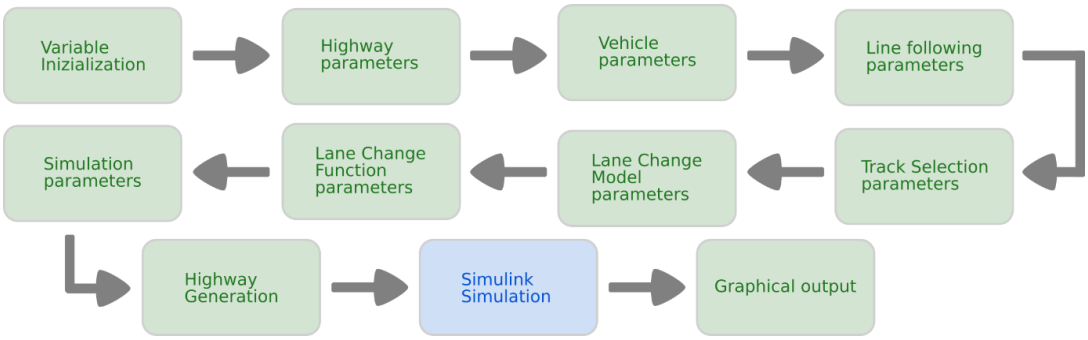


Figure 2 - Flux diagram of the initial data generation

This process takes place in the *traffic_sims.m* file. Its function is to generate and send the initial parameters and track data to the simulation and execute the Simulink model. These initial parameters are the following:

- Highway parameters: number of vehicles, maximum and minimum velocity in the highway and number of lanes.
- Vehicle parameters: Initial position, initial angle, maximum velocity and vehicle length.
- Line Following parameters: Controller parameters, maximum steering angle, maximum visibility distance, bumper-to-bumper minimum distance to the precedent vehicle, desired safety time headway, longitudinal speed time constant.

- Track selection parameters: Percentage of vehicles that take the off-ramp.
- Lane Change Model parameters: Politeness factor, thresholds, maximum visibility distance and forbidden distance.
- Lane Change Function parameters: Lane width, and maximum and minimum lane change maneuver times.
- Simulation parameters: Simulation start and end time.
- Highway geometry parameters: Calculation of the Bézier coefficients and Bézier curves.

After the simulation is concluded, the four following plots are also generated in this file:

1. Vehicle speed vs. time
2. Vehicle position on track A vs. time
3. Traffic density vs. time
4. Traffic flow vs. time

The code for the initial data generation can be found in Annex A.

2.1.1. Geometry Generation

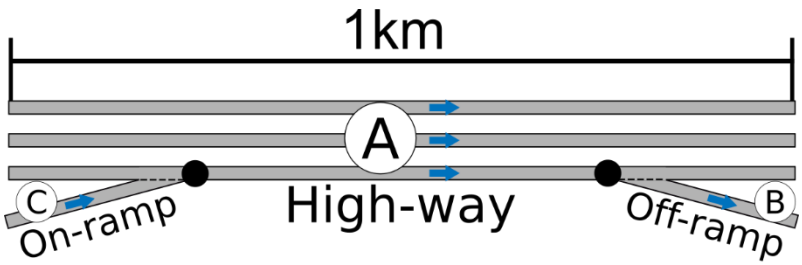


Figure 3 - High-way section

The chosen geometry for the highway section consists of a 1 kilometre long, three-laned straight track (A) together with an on-ramp (C) and an off-ramp (B). Vehicles will come from the left side of the highway section (either from A or from C) and will circulate until they reach the end. Some of them will continue straight and some others will leave the main road using the off-ramp (B).

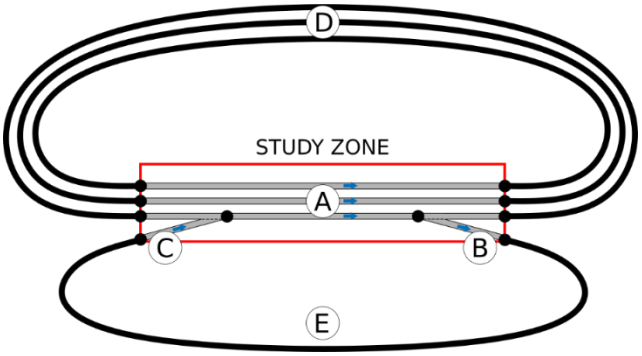


Figure 4 - Complete circuit

For convenience and to simplify the calculations of the position of the vehicles, a loop system has been used (see figure 4). Every car that leaves the study zone using the high-way (A) will follow a track (D) that will take it to the beginning of the high-way section (A). Likewise, a vehicle leaving the study zone by the off-ramp (B), will follow a track that will take it to the on-ramp (C). This loop system will remain hidden during the graphic simulation.

Due to the impossibility of calculating parallel Bézier curves, the control points for the given track were obtained experimentally by trial and error, until the desired geometry was achieved.

The values of the chosen control points can be found in Annex B.

2.1.2. Geometry generation method

All the tracks in the high-way circuit have been generated using Bezier curves. This are parametric curves whose use in computer graphics is very extended due to its possibilities of infinite scaling.

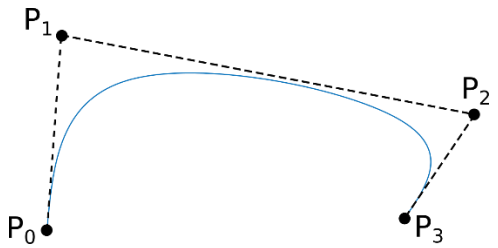


Figure 5 - Control points in an order 3 Bézier curve

They are defined by a set of control points, going from P_0 to P_n, where n is the curve order. These condition the shape of the curvature obtained, as seen in figure 5.

The general explicit definition of a Bezier curve is as it follows:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-q)^{n-i} q^i P_i \qquad (2.1)$$

Being *n* the curve order (n=1 for linear Bezier curves, n=2 for quadratic Bezier curves, …) and *q* being a parameter that varies from 0 to 1, corresponding to the beginning and end of the trajectory described, respectively.

However, the polynomial form of (1) has been used for this project:

$$B(t) = \sum_{j=0}^{n} q^j C_j \qquad (2.2)$$

Where

$$C_j = \frac{n!}{(n-j)!} \sum_{i=0}^{j} \frac{(-1)^{i+j} P_i}{i!\,(j-i)!} = \prod_{m=0}^{j-1} (n-m) \sum_{i=0}^{j} \frac{(-1)^{i+j} P_i}{i!\,(j-i)!} \qquad (2.3)$$

Being C_j the Bézier coefficients. This latter form was chosen over the general explicit form due to the possibility of computing C_j before multiple calculations of *B(t)*.

The code for obtaining the Bezier coefficients used in this project can be found in Annex C.

After obtaining the Bezier coefficients, we obtain xy coordenates for a given value of q between 0 and 1 by multiplying each C_j by its corresponding *q* elevated to *j*, as we can see in Equation 1.2. The code for obtaining the Bézier curve can be found in Annex D.

As a result of (1) we obtain a parametrised curve σ(q), that provides the xy coordenates of the trajectory:

$$\sigma(q) = \begin{pmatrix} \sigma_x(q) \\ \sigma_y(q) \end{pmatrix} \qquad (2.4)$$

Where $\sigma_x(q)$ and $\sigma_y(q)$ are n-th order polynomials that, when solved, output the corresponding xy coordenates for a given value of q between 0 and 1:

$$\sigma_x(q) = c_{n,x}q^n + \cdots + c_{1,x}q + c_{0,x} \quad (2.5)$$

$$\sigma_y(q) = c_{n,y}q^n + \cdots + c_{1,y}q + c_{0,y} \quad (2.6)$$

2.1.3. Initial positioning of the vehicles

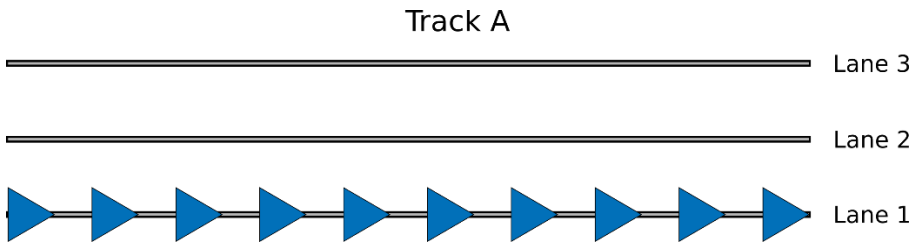


Figure 6 - Initial positioning of the vehicles

A very simple approach has been taken in regards to the initial position of the vehicles, distributing them equally spaced along lane 1 of track A. This causes a limitation in the number of cars of the simulation, since its limited to the length of track A, and would need to be reviewed in future versions of this model.

2.2. Simulation model

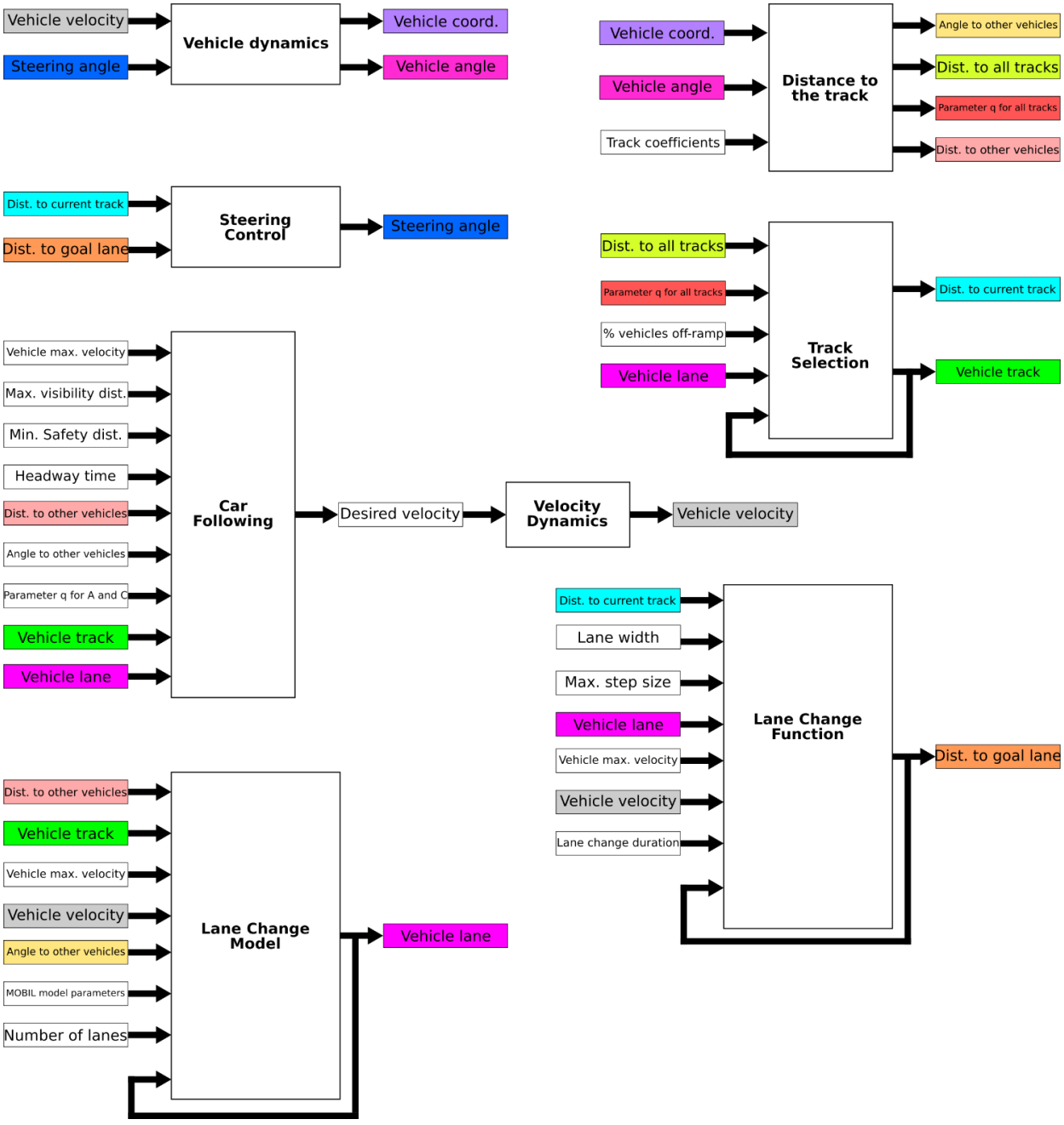


Figure 7 - Simulink model flux diagram. The non-coloured parameters are constants.

After being given a set of initial parameters, a Simulink Model formed by various blocks, will proceed to calculate the traffic results for the amount of time specified in the *traffic_sims* program. This model is comprised of the following functions:

1. Vehicle dynamics
2. Distance to the track and to the other vehicles
3. Steering control
4. Line Following Function
5. Velocity dynamics
6. Lane Change model
7. Lane Change function

An overall view of the Simulink Model and the definition of each one of the parameters used can be found in Annex E.

2.2.1. Vehicle Dynamics

This function has the speed of the vehicles, *vf*, and their steering angle, *δ*, as inputs. It outputs the XY coordinates for every vehicle, as well as the angle with respect to the global frame of reference, ψ.

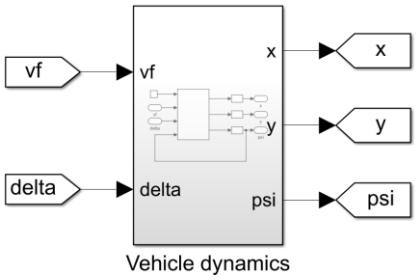


Figure 8 - Vehicle Dynamics Simulink Block

After generating the Bézier curves, it is needed to build a model to ensure that the vehicles follow the desired path accurately. For this purpose, a two-wheel kinematic model of the vehicle (Dòria, 2018) has been adopted, as seen in figure 8. This model is accurate for low

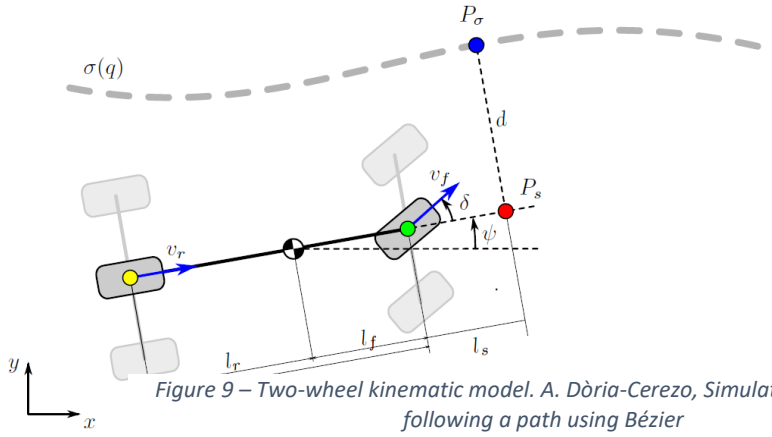


Figure 9 – Two-wheel kinematic model. A. Dòria-Cerezo, Simulations of a vehicle following a path using Bézier

to medium velocities. For high speeds, inertia of the vehicle and tire models should be taken into account.

The differential model that describes the motion of the front wheel is:

- Rate of change of the x coordinate

$$\dot{x}_f = v_f \cos(\psi + \delta) \quad (2.7)$$

- Rate of change of the y coordinate

$$\dot{y}_f = v_f \sin(\psi + \delta) \quad (2.8)$$

- Rate of change of the angle with respect to the horizontal axis (x axis)

$$\dot{\psi} = \frac{v_f}{l} \sin(\delta) \quad (2.9)$$

Integrating the values obtained for the three expressions above, we can locate the vehicle by obtaining x_f, y_f (front wheel coordinates) and ψ (angle with respect to the horizontal axis).

This model has been implemented using the structure seen in figure 9 (Elías, 2019).

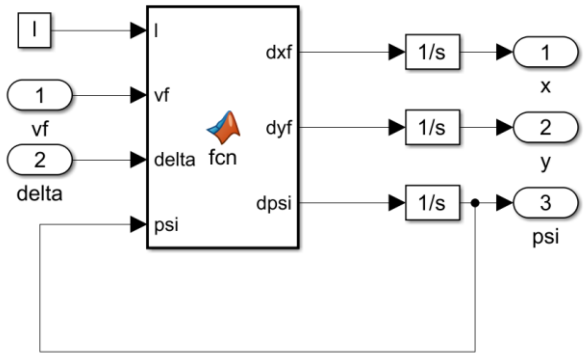


Figure 10 - Implementation of the two-wheel kinematic model in Simulink

The code for the MATLAB function used in this block can be found in Annex F.

2.2.2. Calculation of distances to the track and other vehicles

This block has the XY coordinates, the angle ψ and the track coefficients as inputs. It outputs the distance and the value of the parameter *q* for every track. It also outputs the distance and four-quadrant inverse tangent respect to every other vehicle on the simulation.

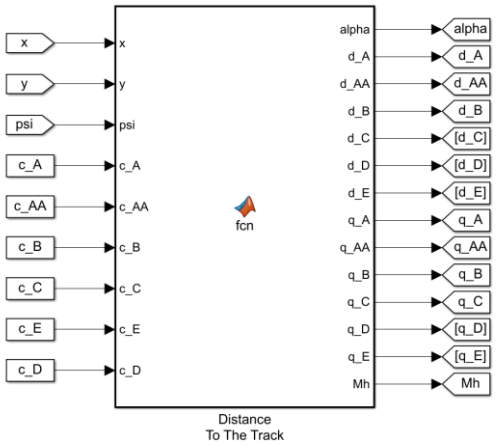


Figure 11 - Distance To The Track Simulink block

For a vehicle to follow a given track, we must know the current distance to the path in order to minimize it. The intersection point of the front-wheel point of coordinates and the given path σ can be obtained as (see figure 8):

$$P_\sigma = \begin{pmatrix} \sigma_x(q^*) \\ \sigma_y(q^*) \end{pmatrix} = P_f + R(\psi) \begin{pmatrix} l_s \\ d \end{pmatrix} \quad (2.10)$$

Where

$$R(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \quad (2.11)$$

Using $P_f = (x_f, y_f)$ and isolating $\begin{pmatrix} l_s \\ d \end{pmatrix}$ on one side of the equation we obtain:

$$\begin{pmatrix} l_s \\ d \end{pmatrix} = R(\psi)^{-1} \begin{pmatrix} \sigma_x(q^*) - x_f \\ \sigma_y(q^*) - y_f \end{pmatrix} \quad (2.12)$$

Developing the first row and asuming l_s=0 for our particular model, we can obtain q^* for a given x_f and y_f

$$\cos\psi\,(\sigma_x(q^*) - x_f) + \sin\psi\,(\sigma_y(q^*) - y_f) = 0 \quad (2.13)$$

Which then allows us to obtain the value for d, developing the second row of (2.6)

$$d = -\sin\psi\left(\sigma_x(q^*) - x_f\right) + \cos\psi\left(\sigma_y(q^*) - y_f\right) \quad (2.14)$$

To calculate the distance to the other vehicles:

$$h = \sqrt{(x_f - x'_f)^2 + (y_f - y'_f)^2} \quad (2.15)$$

Note: The distance to the track is always calculated with respect to lane 1.

Being x'_f and y'_f the cordinates of any other vehicle

To calculate the four-quadrant inverse tangent respect to the other vehicles

$$\alpha = tan^{-1}(y_f - y'_f, \quad x_f - x'_f - \psi) \quad (2.16)$$

This value allows us to know which vehicles are in the back respect to the vehicle (α<0) and those which are in front (α>0).

The code for obtaining the distance to the track *d* and the value of the parameter *q* can be found in Annex G.

2.2.3. Steering Control

This block has the fluctuation with respect to the trajectory, *d*, and the distance to lane 1, *d_lane* as inputs. It outputs the correction angle needed in the trajectory for the vehicle to keep following the given path, *delta*.

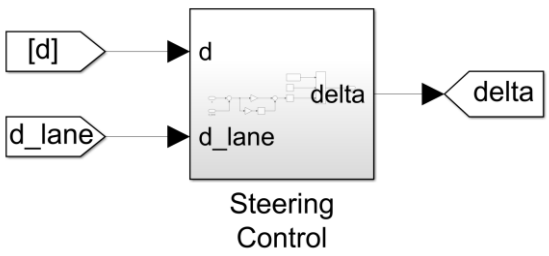


Figure 12 - Steering Control Block

The distance required to calculate the steering angle is the fluctuation from the trajectory, and can be obtained as the sum of d and d_lane (see figure 13)

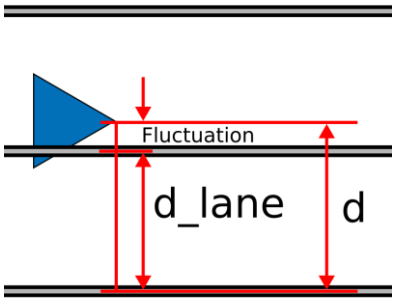


Figure 13 - Fluctuation with respect to the trajectory

For a vehicle to follow a given path, the angle of the trajectory must by corrected in each step. A PI controller (Proportional-Integral controller) was made for the steering control (Elías, 2019). These are the most popular variation of controllers and are composed of a proportional term (P) to remove the gross error, and an integral term (I) to eliminate the residual offset error.

Real vehicles have a maximum turning angle, so a saturation block is needed to set a limit output on the value of delta.

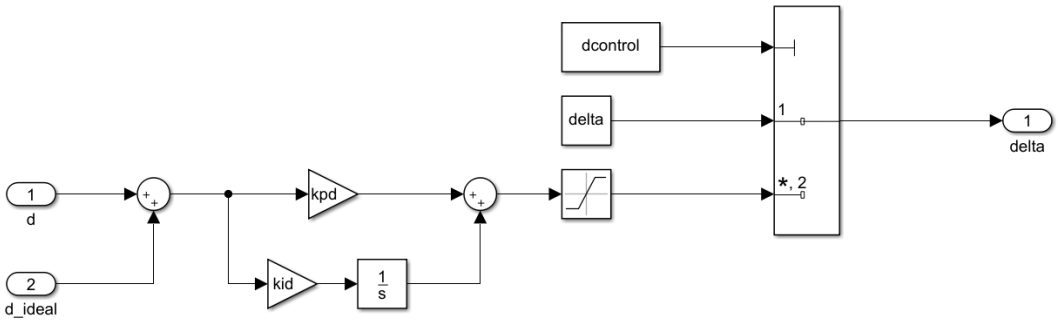


Figure 14 - PI controller for the steering control

2.2.4. Track Selection

The Distance to The Track block calculates, for the current vehicle, the distance and the parameter q for all the tracks, simultaneously. These are the inputs for the Track Selection block. The output of this block is the q and d for the current track, so it only selects a specific set of parameters, depending on which track is the vehicle circulating through. Because the circuit is formed by various tracks, a track selection policy has to be adopted.

This block also outputs the position of the vehicles in track A (*simsPosA*) for a later graphical representation.

A memory block was introduced for the track input to avoid algebraic loops.

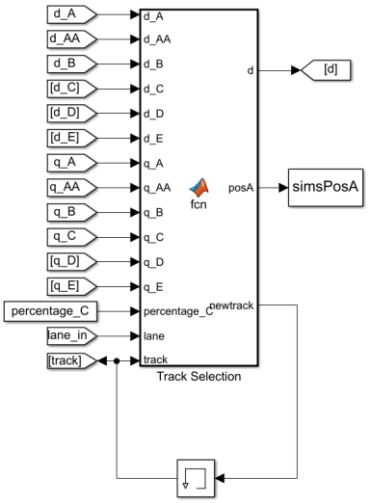


Figure 15 - Track Selection Block

The following assumptions have been taken into the making of the track selection policy:

1. A vehicle reaches the end of a track when the curve parameter q is greater than or equal to 1.
2. If a vehicle reaches the end of a track, it has to enter a new track with q=0
3. The decision of going out on the off ramp or to continue straight through the highway is taken in the beginning of A or in the end of C. This decision is random.
4. Vehicles have to follow a logical succession of tracks: A-D-A or A-B-E-C-A (see figure 4).
5. If a vehicle wants to take the off-ramp but it doesn't find itself in the outside lane in the corresponding moment, decision is changed to stay on the highway.
6. The percentage of vehicles that take the off-ramp can be controlled through a parameter called percentage_C.
7. When a vehicle is travelling through a track, the outputs d and q are the ones of that track
8. If a vehicle changes track, the outputs d and q are those of the new track.

Code for the Track Selection Block can be found in Annex H.

2.2.5. Car following

In this block the car following behaviour is calculated. It outputs the safety distance to the leader vehicle, *hc*, and the desired velocity needed to adopt to the current situation, *vf_desired*.

The v dynamics block that has *vf_desired* as input is explained further on.

The yield behaviour on the on-ramp is also calculated in this block, for convenience.

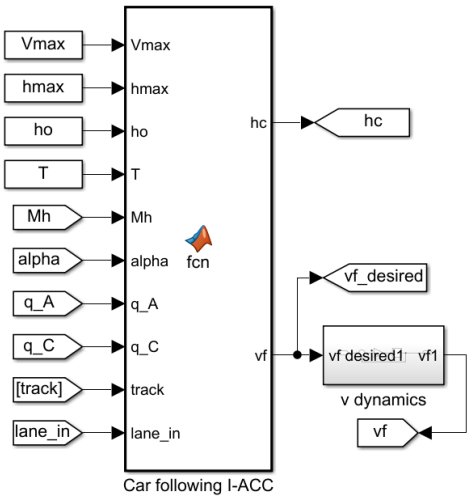
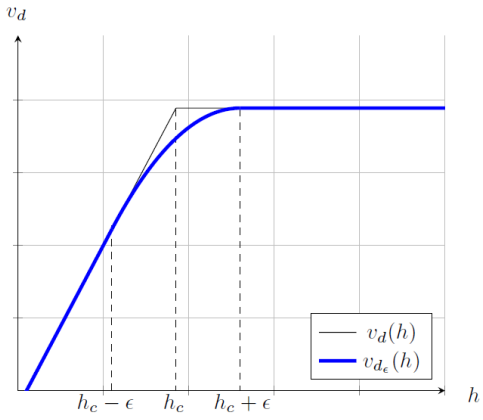


Figure 16 - Car Following Block

For the car following protocol, a first order sliding mode-based adaptive cruise controller was chosen (Dòria et al., 2018). This sets a speed for the current vehicle dependent on the distance to the precedent vehicle while maintaining a smooth movement. The function for the desired speed is as it follows



$$dv_{d_\epsilon}(h) = \begin{cases} v_{max} + \dfrac{h - h_c}{T}, & h - h_c \leq -\epsilon \\ v_{max} - \dfrac{1}{4\epsilon T}(h - h_c - \epsilon)^2, & |h - h_c| < \epsilon \\ v_{max}\,, & h - h_c \geq \epsilon \end{cases} \quad (2.17)$$

Being h_c the safety distance and h the real distance to the precedent vehicle. ϵ is the threshold that determines which velocity policy is chosen.

Figure 17 - Function vd_ϵ and its non-smooth approximation vd. Arnau Dòria-Cerezo. A first order sliding mode-based adaptive cruise controller

h_c can be calculated as

$$h_c = h_0 + T \cdot V_{max} \quad (2.18)$$

Where h_0 is the desired net bumper-to-bumper minimum distance to the precedent vehicle.

To avoid interference with other vehicles, car following is disabled for vehicles on different tracks and lanes.

For the yield behaviour on the on-ramp, a simple protocol was chosen. If a vehicle wants to enter the highway through the on-ramp (track C) and sees another vehicle coming through track A, its speed is automatically set to 0 until the approaching vehicle gets to a safety distance.

The code for the Car Following Block can be found in Annex I.

2.2.6. Velocity Dynamics

This block is placed next to the *vf_desired* output from the Car Following Block and has the desired velocity as input and the real velocity as output.

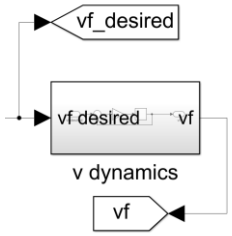


Figure 18 - Velocity Dynamics Block

The velocity obtained in the car following function, *vf_desired*, is a desired speed. This may differ from the real speed, so a controller must be introduced to cause a delay between both. This way, the vehicle takes a realistic amount of time to reduce or increase its speed to accommodate to the current traffic situation. The following block diagram represents the controller, having the desired speed as input and the real speed as output.

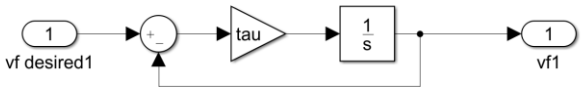


Figure 19 - Velocity Dynamics Controller

2.2.7. Lane Change Model

In this block, the decision of changing or staying in the same lane is taken. It has a set of inputs such as distance to other vehicles, current speed, etc, needed for the lane changing model.

A memory block was introduced for the lane input to prevent algebraic loop error.

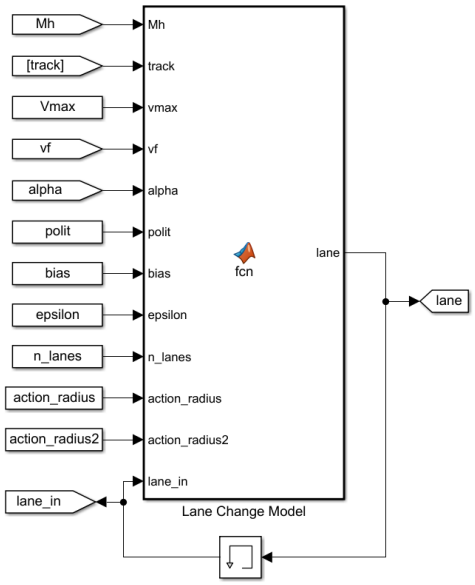


Figure 20 - Lane Change Model Block

The lane change model used in this project is based on the '*Minimizing Overall Braking Induced by Lane Changes'* (MOBIL) model (Kesting, 2006). Although the original protocol is based in accelerations, a speed-based model was used in this project for convenience and ease of implementation on the existing code.

Because this project was done in Europe, an asymmetric lane change criterion was introduced. Specifically, the following rules were assumed:

1. *Passing rule:* Passing on the right-hand lane is forbidden, unless traffic flow is congested. We consider that a vehicle is driving in congested traffic its current speed is lower than v_{crit}. A value of v_{crit}=60 km/h was used for this project.

2. *Lane usage rule:* The right-hand lane is the default lane. Upper lanes should only be used for overtaking.

The proposed condition to meet the first rule is:

$$v = \begin{cases} \min(v, v_{lead}), & if\ v > v_{lead} > v_{crit} \\ v, & otherwise, \end{cases} \quad (2.19)$$

Where v_{lead} is the velocity of the closest front vehicle on the upper lanes.

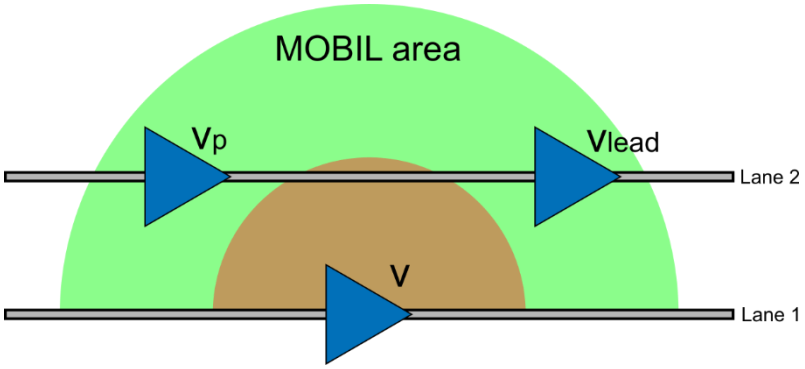


Figure 21 - MOBIL modified model applied in a right to left lane change

The resulting asymmetric incentive criterion applied for lane changes from left (L) to right (R) and right (R) to left (L) are:

$$L \rightarrow R: \qquad \tilde{v} - v + p\left(\widetilde{v_p} - v_p\right) > \Delta a_{th} - \Delta a_{bias} \qquad (2.20)$$

$$R \rightarrow L: \qquad \tilde{v} - v + p\left(\widetilde{v_p} - v_p\right) > \Delta a_{th} + \Delta a_{bias} \qquad (2.21)$$

Where the values with a tilde refer to the new situation in case the lane change occurs and Vp is the speed of the closest back vehicle on the goal lane. A threshold Δa_{th} was introduced to prevent lane changes induced for marginal velocity gains. The second rule of the asymmetric lane change criterion was applied by introducing a constant bias Δa_{bias}, which makes it easier for vehicles to go from a left- to-right lane change than from right-to-left lane change. This later parameter has to be larger than the threshold Δa_{th} to prevent vehicles from staying in the left lane even on an empty road.

The only surrounding vehicles taken into account for the lane change operation are those inside the green area (see figure 19). Any other vehicles will be ignored. This green area is defined by the variable *action_radius* in the MATLAB code function

To prevent accidents, this model only applies if there are no surrounding vehicles inside the red area (see figure 19). Otherwise, the study vehicle stays on the same lane until the red area is free. This red area is defined by the variable *forbidden_radius* in the MATLAB code function.

The condition (2.13) for the first rule was finally not applied due to unnatural behaviour of the vehicle if the closest front car in the target lane would be moving at a lower velocity and inside the forbidden radius. This rule will require a more complex approach on future versions/modifications of this model.

The code for the Lane Changing Model can be found in Annex J.

2.2.8. Lane Change Function

For a smooth lane change, a lane change function was introduced. This function decreases or increases the distance to lane 1 gradually in each step until the vehicle arrives to the intended lane.

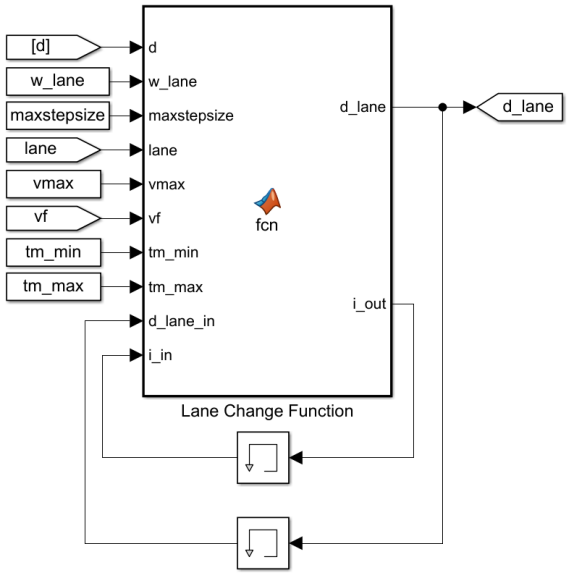


Figure 22 - Lane Change Function Block

For a smooth lane change, a lane change function was introduced. This function decreases or increases the distance to lane 1 gradually in each step until the vehicle arrives to the intended lane.

To calculate the distance variation in each step the following formula was used:

$$\min \Delta d = \frac{Lane\ width}{t_m} \cdot maxstepsize \qquad (2.22)$$

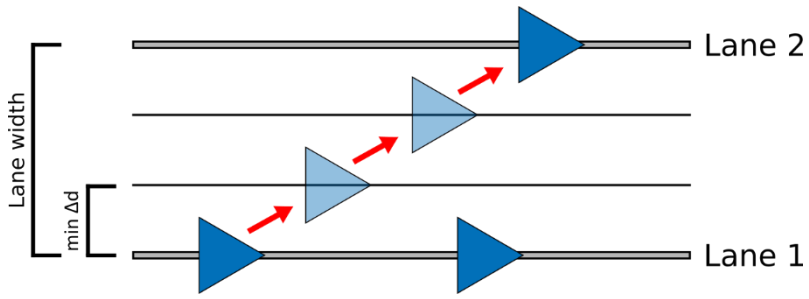


Figure 23 - Example of a right-to-left lane change

Where *Lane width* is the distance between lanes, *tm* is the maneuver time and *maxstepsize* is the fundamental sample time used in the simulation. The value obtained, min Δd, is then the variation in distance to lane 1 that the vehicle has to take in each step to arrive to the intended lane in the given time *tm*.

The meneuver time is not a fixed value and its calculated for every vehicle using

$$t_m = t_{m_{min}} + \frac{v_f}{v_{max}} \cdot \left(t_{m_{max}} - t_{m_{min}}\right) \qquad (2.23)$$

Where $t_{m_{max}}$ and $t_{m_{min}}$ are the minimum and maximum maneuver times for all vehicles, *vf* is the current speed and v_{max} is the maximum possible speed for the given vehicle. This expression considers that the meneuver time increases linearly with the velocity of the vehicle, calculating the current speed ratio with respect to its maximum possible value and then outputting a value for t_m, within $t_{m_{min}}$ and $t_{m_{max}}$, using the same proportion.

This model is possible because the Simulink solver is set to fixed-step. Otherwise, the parameter *maxstepsize* would be variable and we wouldn't be able to calculate the parameter *minΔd* correctly.

The proposed lane change function doesn't avoid crashes during the lane changing process, needing a more complex approach on future versions/modifications of the model.

The code for the Lane Changing Function can be found in Annex K.

2.3. Graphic and animation outputs

The results obtained in the Simulink Model are only numerical. A graphical representation is then needed, for convenience. The data calculated in the *traffic_sims_model* file is collected by the *traffic_sims_animation* file, where the animation occurs. This data consists of:

- XY coordinates of every vehicle for every given time
- Angle of every vehicle with respect to the X axis for every given time
- Speed of every vehicle for every given time
- Position of every vehicle on the highway for every given time

The geometry of the highway is also recalculated in order to represent it graphically.

In order to represent the vehicles, it is first needed to establish a geometrical shape to represent them.

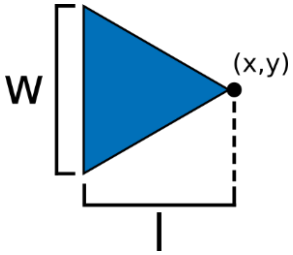


Figure 24 - Chosen geometry for the vehicles in the simulation

The chosen shape for the car geometry was an isosceles triangle with the point of coordinates in the front end, as seen in figure 22. This shape was chosen because it is the simpler geometry that indicates direction, and is widely used in modern GPS systems.

Several views of the high-way were created to give a better understanding of the general state of the traffic. Specifically, four different views were generated: a general view of the highway, a view of the straight section, and a view of the on-ramp and of the off-ramp.

Additionally, two plots were added: velocity vs. time and position on A vs. time.

The animation appearance can be observed in figure 20 and the code can be found in Annex L.

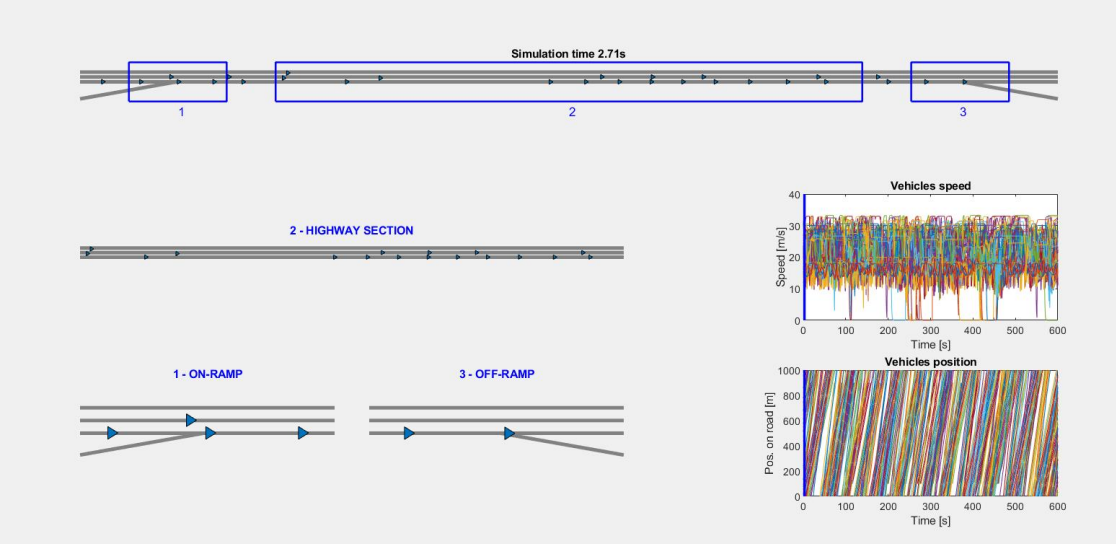


Figure 25 - Graphical output of the animation file

3. Simulation

For the demo simulation, the following values have been introduced:

Variable name	Meaning	Value
n	Number of cars in the simulation	80
vmax (km/h)	Max. speed in the highway	120
vmin (km/h)	Min. speed in the highway	60
n_lanes	Number of lanes	3
ho (m)	Bumper to bumper min. distance	6
T (s)	Headway time	1,8
percentage_C	% vehicles that take the off-ramp	0,1
polit	Politeness factor	0,5
epsilon	MOBIL threshold	10
bias	MOBIL right lane bias	11
tfinal (s)	Time of the simulation	600

And the following plots were obtained:

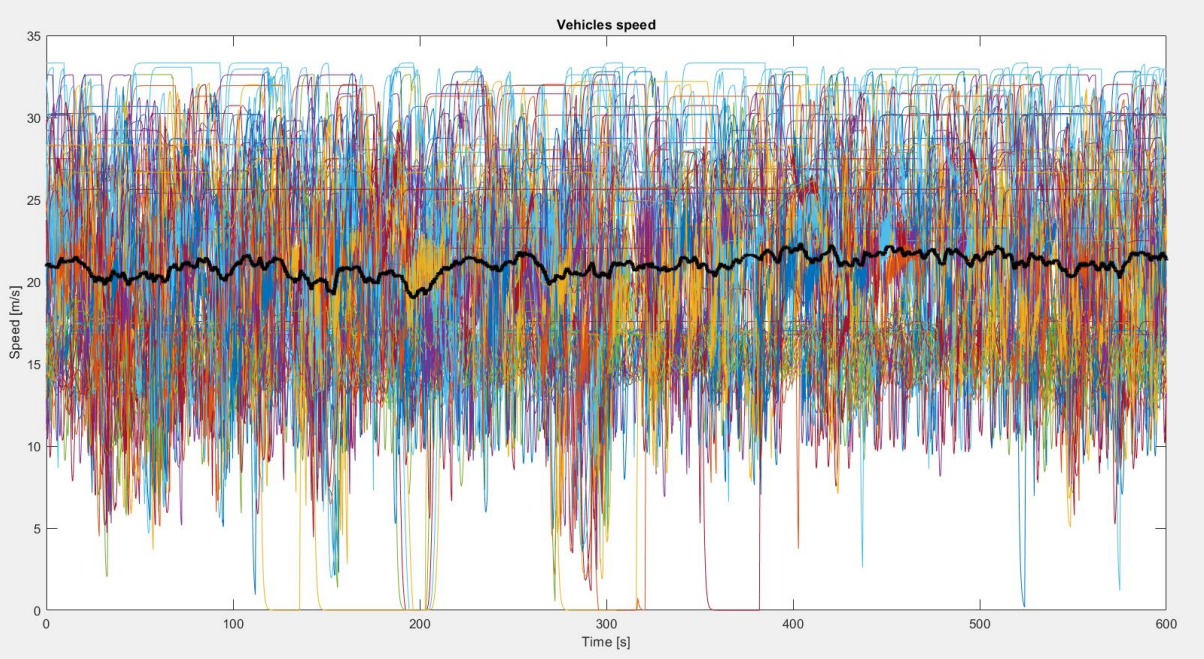


Figure 26 - Speed vs. time plot with mean speed representation

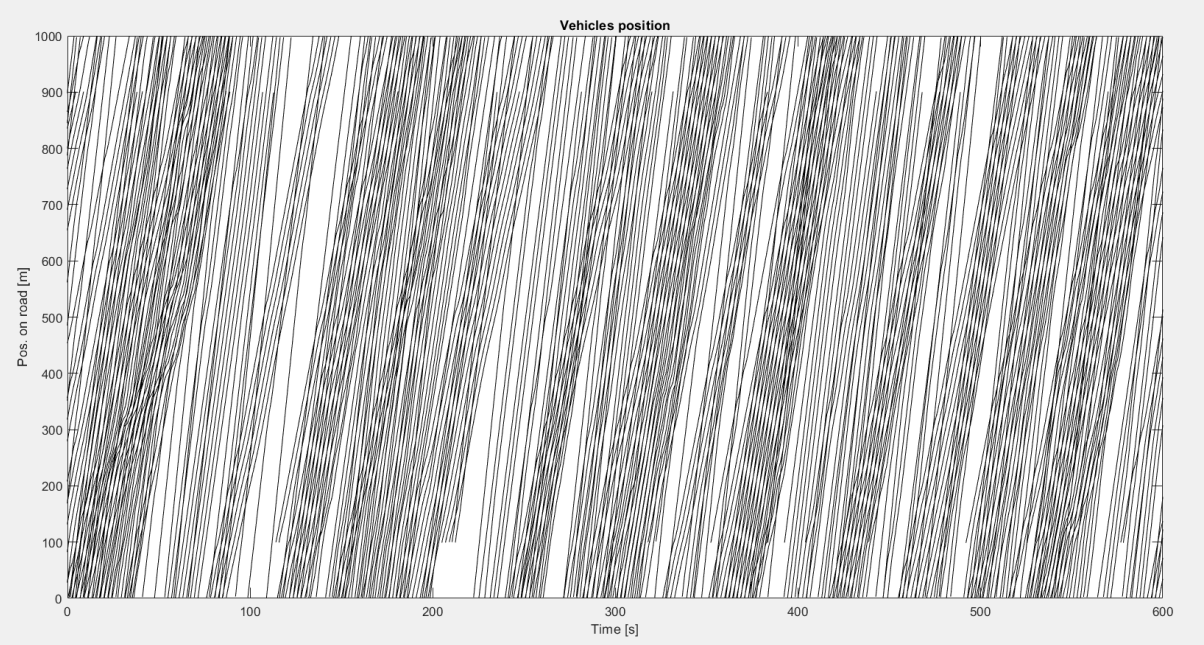


Figure 27 - Vehicles position on track A vs. time plot

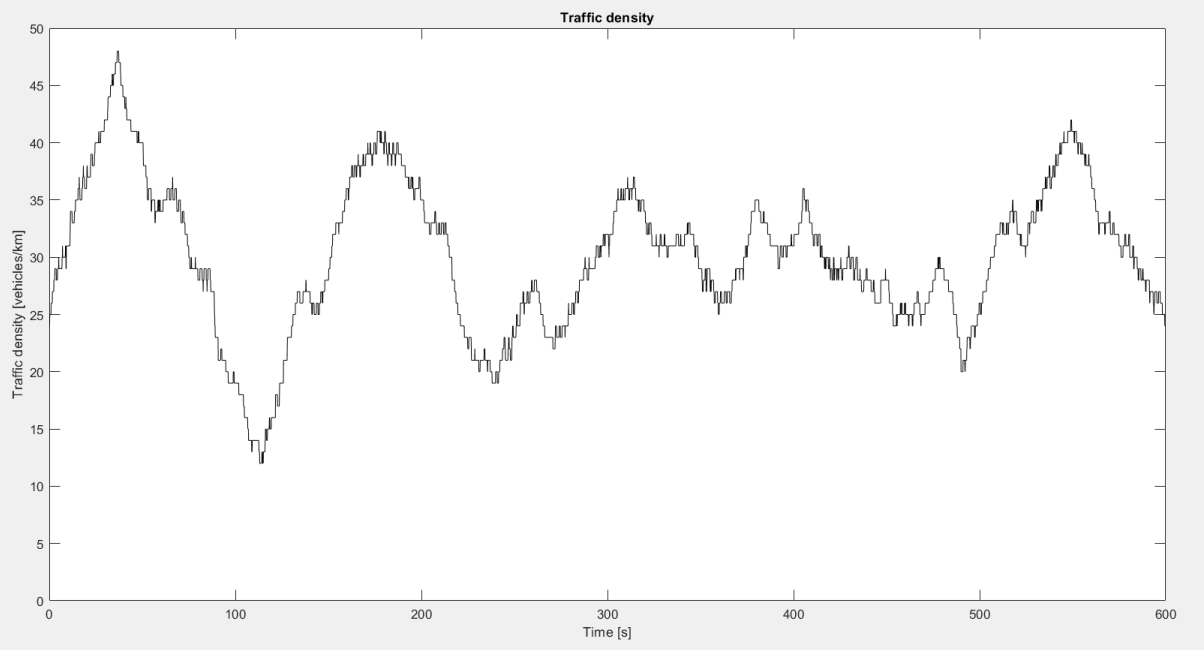


Figure 28 - Traffic density vs. time plot

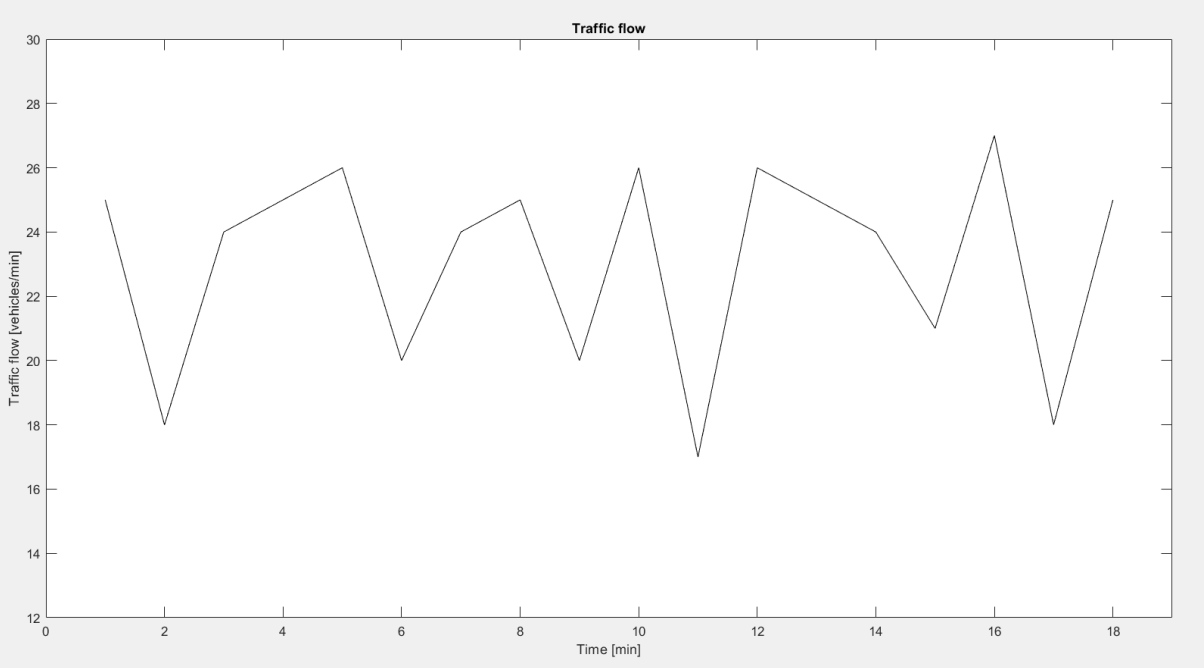


Figure 29 - Traffic flow vs. time plot

As observed in figure 24, the average speed revolves around a value of 20-21 m/s. This is value is coherent since its very close to the expected average speed obtained from the maximum and minimum values of the velocity for the highway.

Some vehicles encounter a speed of zero or almost zero during the simulation. This is due to congested traffic or to the need to stop on the on-ramp to allow priority to the incoming cars.

In figure 25 the vehicle position on track A can be observed. Some line cross each other, meaning that a lane change and a consequent overtaking occurred. Some lines don't start at zero or don't end at 1km. This is due to vehicles entering track A through the on-ramp o exiting through the off-ramp, respectively.

In figure 26 and 27 we can see that magnitudes like traffic flow and traffic density fluctuate considerably. This is due to the construction of the simulation. The loop construction (see Geometry Generation section) of the highway section does not allow a good control of these two variables.

4. Conclusions

As it has been shown in this project, simulating a highway it is much more complex than what it seems. The following conclusions can be drawn:

- The loop system introduced in the Geometry Generation Section is not ideal, because it does not allow to control parameters like traffic density or traffic flow accurately.
- The on-ramp access behaviour presented is very simple and does not match the real behaviour accurately.
- The lane-change function does not avoid crashes between the vehicles involved because it does not take into consideration the velocity of the overtaken vehicle.
- The initial distribution of the vehicles is not ideal, because it only allows approximately 100 vehicles in the simulation. Any number higher than that would cause vehicles to start following wrong tracks or would cause insoluble congested traffic.
- The model seems to work correctly, but an in-depth verification is needed to ensure it emulates real behaviour accurately.
- The values for the different variables (politeness factor, action radius, forbidden radius…) have been obtained experimentally and would need to be refined for a more real simulation.

Nevertheless, this system shows positive signs of coherent functioning.

5. Future works

Future projects could derive from this one. List of suggested possible improvements or implementations:

- Incorporation of human Line Following models, like IDM, to analyse the effect of the progressive incorporation of autonomous vehicles in normal human traffic.
- Redefinition of the loop system used in this project in the Geometry Generation section. A better system would allow to control variables like traffic density or traffic flow.
- Improvement of the on-ramp access behaviour. This would allow a more realistic access to the highway.
- Improvement of the initial position of the vehicles. This would allow more vehicles in the simulation, making possible to analyse congested traffic behaviour.
- Improvement of the Lane Change Function to prevent crashes during the lane change operations.
- Variable value refinement. This would allow the model to operate at its best.

6. Project budget

Concept	Unit cost	Units	Total
Engineering costs			
Research	10€/h	50,00	500,00 €
Programming	10€/h	250,00	2.500,00 €
Preparing and correcting simulations	10€/h	50,00	500,00 €
Redaction of the project report	10€/h	250,00	2.500,00 €
Technological resources			
Computer + Microsoft Office Home	900,00 €	0,10	90,00 €
MATLAB and Simulink Student version	0,00 €	1,00	0,00 €
Transportation costs			
Metro ticket	2,15 €	24,00	51,60 €

TOTAL	**6.141,60 €**

The following considerations have been taken into account in the making of the budget:

- The salary for a junior mechanical engineer has been fixed in 10€/h. This low value is due to the previous lack of experience in the field of knowledge related to the project.
- The cost of the computer is considered to be an amortized quantity. An average lifespan of 5 years was considered for the calculation.
- The price of the metro ticket is for the year 2018, since the majority of this project was done during that year. Prices have changed for 2019.
- The electric consumption has not been taken into account because it was considered negligible.

7. Environmental impact

The environmental impact of this project is negligible. The only energy consumption of this simulation would be that of the computer used, which is almost null. However, future studies done using variations of this software could improve traffic density and traffic flow, helping to increase driving efficiency and reducing, at the same time, greenhouse gas emissions.

Bibliography

- Bosch, B., 2017. *Influence of the Cooperative Adaptive Cruise Control to the traffic Flow*. Bachelor's Thesis, Universitat Politécnica de Catalunya.

- Cartró, J., 2017. *Simulació de la fluència del trànsit de vehicles comparant el comportament humà i l'impacte de vehicles autònoms*. Bachelor's Thesis, Universitat Politécnica de Catalunya.

- Dòria, A, 2018. *Simulations of a vehicle following a path using Bézier curves. Internal report UPC.*

- Dòria, A. Olm, J. Benedito, E. Biel, D. Repecho, V., 2019. *A first order sliding mode-based adaptive cruise controller*. *Internal report UPC.*

- Elías, M. M., 2019. *Modelling and simulation of autonomous vehicles in a multi-lane and heterogeneuous traffic road*. Bachelor's Thesis, Universitat Politécnica de Catalunya.

- Benedito, E., Dòria, 2018. A. *Influence of cooperative-controlled driving in the traffic flow.* Proc. 2018 IEEE International Conference on Industrial Technology (ICIT): Lyon, France: February 19-22, 2018.

- Fernández, M., 2017. *Energy efficiency comparison between human drivers and adaptive cruise control system*. Bachelor's Thesis, Universitat Politécnica de Catalunya.

- Kesting, A., 2007. *MOBIL: General Lane-Changing Model for Car-Following Models.* Transportation Research Record, Vol 1999, Issue 1.

ANNEX A – Code for the initial data generation file

```
%-----------------
% Traffic simulation
%-----------------
clc;close all;

set(gcf,'Renderer','Painters')  % Export eps files without pixels
addpath('Bezier')
addpath('Tracks')
run('track_7')


% Highway parameters
n=80;                   % Number of cars in the simulation
vmax=120;               % Highway maximum speed in km/h
vmin=60;                % Highway minimum speed in km/h
n_lanes=3;              % Number of lanes in the highway

% Variable initialization
track_ini=ones(n,2);
i_ini=ones(n,2);
d_lane_ini=zeros(n,1);
lane_ini=ones(n,1);
x0=zeros(n,1);
y0=zeros(n,1);
psi0=zeros(n,1);
delta=zeros(1,n);

for j=1:n
    track_ini(j,1)=1;
    track_ini(j,2)=randi([2 3]);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Car parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('---------------------------------------')
disp('---- Generating car parameters...')
disp('---------------------------------------')
tic

% VEHICLES POSITIONING
for j=1:n
   x0(j)=400-11.25*j;
   y0(j)=0;
   psi0(j)=0;
   Vmax=(rand(1,n)*(vmax-vmin)+vmin)/3.6;
   l=4;
end

toc

% Line Following parameters
dcontrol=2;
kpd=0.1;
kid=1e-3;
deltamax=15/180*pi;
hmax=100*ones(n,1);     % Maximum distance
ho=6;                   % bumper-to-bumper minimum distance to the precedent
vehicle
```

```
T=1.8;                  % Desired safety time headway
tau=1.2;                % Longitudinal speed time constant

% Track Selection parameters
percentage_C=0.1;       % Percentage of vehicles that take the off-ramp.

% Lane Change Model Parameters
polit=0.5;                  % Politness factor
epsilon=10;                % MOBIL Threshold
bias=11;                   % MOBIL bias
action_radius_up=50;       % Action radius from left to right lane changes
action_radius_down=50;     % Action radius from right to left lane changes
forbidden_radius_up=10;    % If any vehicle is inside this radius, upper lane
change is disabled.
forbidden_radius_down=10;  % If any vehicle is inside this radius, lower lane
change is disabled.

% Lane Change Function parameters
w_lane=5;       % Distance between two lanes
tm_min=20;      % Minimum Lane Change Time
tm_max=100;     % Maximum Lane Change Time

% Simulation parameters
tstart=-500;    % Simulation start time
tfinal=600;      % Simulation end time

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Track
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('----------------------------------------')
disp('---- Generating track coefficients...')
disp('----------------------------------------')
tic

% HIGHWAY GENERATION
% Bezier polynomial coefficients
[nB_A,c_A]=Bcoefficients(P_A);      %For track A
[nB_AA,c_AA]=Bcoefficients(P_AA);   %For track AA
[nB_A2,c_A2]=Bcoefficients(P_A2);   %For track A2
[nB_A3,c_A3]=Bcoefficients(P_A3);   %For track A3

[nB_D,c_D]=Bcoefficients(P_D);      %For track D
[nB_D2,c_D2]=Bcoefficients(P_D2);   %For track D2
[nB_D3,c_D3]=Bcoefficients(P_D3);   %For track D3

[nB_C,c_C]=Bcoefficients(P_C);      %For track C

[nB_B,c_B]=Bcoefficients(P_B);      %For track D

[nB_E,c_E]=Bcoefficients(P_E);      %For track E

% Bezier curve
npts=100;               % num of sampled points
[t_A,B_A]=Bcurve(c_A,npts);     %For track A

[t_AA,B_AA]=Bcurve(c_AA,npts);  %For track AA
[t_A2,B_A2]=Bcurve(c_A2,npts);  %For track A2
[t_A3,B_A3]=Bcurve(c_A3,npts);  %For track A3

[t_D,B_D]=Bcurve(c_D,npts);     %For track D
[t_D2,B_D2]=Bcurve(c_D2,npts);  %For track D2
[t_D3,B_D3]=Bcurve(c_D3,npts);  %For track D3

[t_C,B_C]=Bcurve(c_C,npts);     %For track C
```

```
[t_B,B_B]=Bcurve(c_B,npts);     %For track B

[t_E,B_E]=Bcurve(c_E,npts);     %For track E

toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('---------------------------------------')
disp('---- Simulating ...')
disp('---------------------------------------')

tic
maxstepsize=1e-1;
sim('traffic_sims_v08_02_model')
toc

%-------------------------------------------------------------------------
% GRAPHICAL OUTPUT

% Velocity and Mean Velocity
figure(1)
plot(tout, simsvf);
hold on
plot(tout, simsmean_vf, 'Linewidth', 3, 'Color', 'k');
title('Vehicles speed');
axis manual
axis([0 tfinal 0 35])
xlabel('Time [s]'); ylabel('Speed [m/s]')

% Position on A
figure(2)
plot(tout, simsPosA, 'Color', 'k');
title('Vehicles position');
axis([0 tfinal 0 1000])
xlabel('Time [s]'); ylabel('Pos. on road [m]')

% Traffic density
figure(3)
plot(tout, simscars_A, 'Color', 'k');
title('Traffic density');
axis([0 tfinal 0 50])
xlabel('Time [s]'); ylabel('Traffic density [vehicles/km]')

% Traffic flow
figure(4)
minute=tstart+60;
p=1;
u=1;
traffic_flow_min=zeros(round((-tstart+tfinal)/60)+1,1);
for k=tstart:maxstepsize:tfinal
    if k<minute
        traffic_flow_min(u)=traffic_flow_min(u)+simstraffic_flow_step(p);
        p=p+1;
    elseif k==minute
        u=u+1;
        minute=minute+60;
        traffic_flow_min(u)=traffic_flow_min(u)+simstraffic_flow_step(p);
        p=p+1;
    end

    if k==tfinal
```

```
        traffic_flow_min(u)=NaN;
    end
end
time_min=1:1:length(traffic_flow_min);
plot(time_min, traffic_flow_min, 'Color', 'k');
title('Traffic flow');
axis([0 length(traffic_flow_min) 12 30])
xlabel('Time [min]'); ylabel('Traffic flow [vehicles/min]')
```

Listing 1: traffic_sims.m, Matlab Code Function

ANNEX B – Chosen control points for the Bézier curves

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Chosen control points to obtain the geometry of the highway using Bézier
% curves.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

P_A=[-500 0; 500-100*cos(pi/18) 0];
P_AA=[500-100*cos(pi/18) 0; 500 0];
P_A2=[-500 5;500 5];
P_A3=[-500 10;500 10];

P_D=[500 0; 850 0; 850 300; -850 300; -850,0; -500 0];
P_D2=[500 5; 630 5; 625 170; -625 170; -625 5; -500 5];
P_D3=[500 10; 600 10; 600 140; -600 140; -600 10; -500 10];

P_C=[-500 -100*sin(pi/18); -500+100*cos(pi/18) 0];

P_B=[500-100*cos(pi/18) 0; 500 -100*sin(pi/18)];

P_E=[P_B(2,1) P_B(2,2); P_B(2,1)+700*cos(pi/18) P_B(2,2)-1400*sin(pi/18); -
(P_B(2,1)+700*cos(pi/18)) P_B(2,2)-1400*sin(pi/18); P_C(1,1) P_C(1,2)];
```

Listing 2: track_7.m, Matlab code function

ANNEX C – Code for obtaining the Bézier coefficients

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function to calculate the Bezier polynomial coefficients
% P: Control Points
% n: Curve order
% c: Bezier coefficients
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [n,c] = Bcoefficients(P)
n=length(P);
    for cnt1=1:n
        aux1=[0;0];
        j=n-cnt1;
        for cnt2=1:j+1
            k=cnt2-1;
            aux1=aux1+(-1)^(k+j)/(factorial(k)*factorial(j-k))*P(cnt2,:)';
        end
        aux2=1;
        for m=1:j
            aux2=aux2*(n-m);
        end
        c(:,cnt1)=aux2*aux1;
    end

end
```

Listing 3 - Bcoefficients.m, Matlab code function

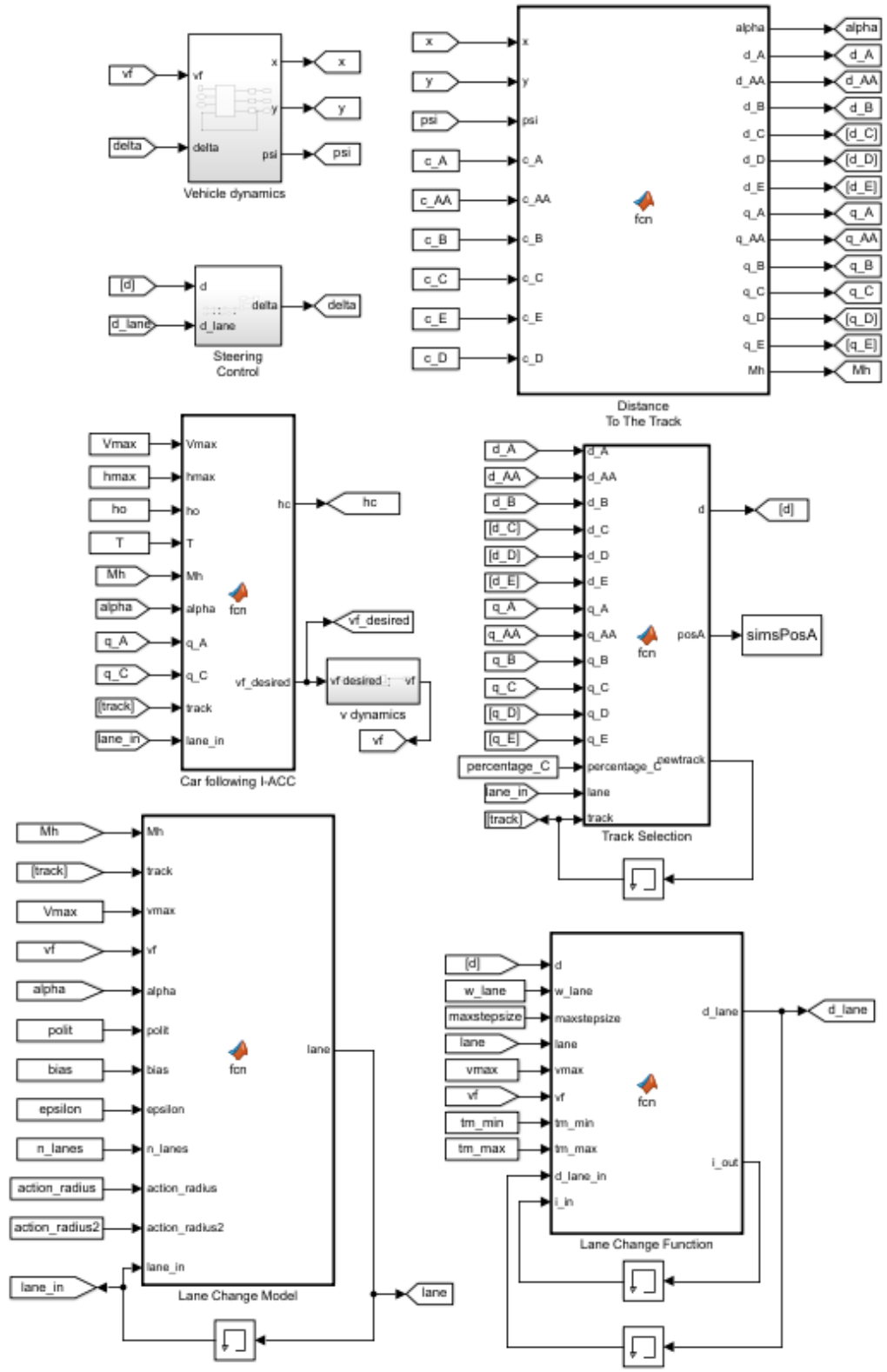
ANNEX D – Code for obtaining the Bézier curve

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Function to calculate a 2D Bezier curve, B,
% parametrized by t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [t,B] = Bcurve(c,npts)
t = linspace(0,1,npts);
n=length(c);
B=zeros(2,npts);        % Bezier curve initial
    for cnt3=1:n
       j=n-cnt3;
       B=B+t.^j.*c(:,cnt3);
    end
end
```

Listing 4 - Bcurve.m, Matlab code function

ANNEX E – General view of the Simulink Model and definition of the inputs and outputs



Parameter	Definition
action_radius	Distance from the vehicle where the MOBIL model is applied from left to right
action_radius2	Distance from the vehicle where the MOBIL model is applied from right to left
alpha	Matrix with the the four-quadrant inverse tangent respect to all other vehicles
bias	Bias for the Lane Change Model (MOBIL)
c_A	Bézier coefficients for track A
c_AA	Bézier coefficients for track AA
c_B	Bézier coefficients for track B
c_C	Bézier coefficients for track C
c_D	Bézier coefficients for track D
c_E	Bézier coefficients for track E
d_A	Distance to track A
d_AA	Distance to track AA
d_C	Distance to track C
d_D	Distance to track D
d_E	Distance to track E
d_lane	Distance to the goal lane in case of lane change
delta	Steering angle
epsilon	Threshold for the MOBIL model
hmax	Maximum visibility for the Car Following model
ho	Desired net bumper-to-bumper minimum distance to the precedent vehicle
lane	Current lane
maxstepsize	Fixed-step size for the simulation
Mh	Matrix with the distance to all vehicles
n_lanes	Number of lanes
percentage_C	Percentage of vehicles that take the off-ramp
polit	Politeness factor for the Lane Change Model (MOBIL)
psi	Angle of the vehicle with respect to the X axis
q_A	Value of the parameter q in track A
q_AA	Value of the parameter q in track AA
q_B	Value of the parameter q in track B
q_C	Value of the parameter q in track C
q_D	Value of the parameter q in track D
q_E	Value of the parameter q in track E
T	Desired safety time headway when following other vehicles
Tm_max	Maximum maneuver time for a lane change
Tm_min	Minimum maneuver time for a lane change
Track	Current track
Vf	Current speed
Vf_desired	Needed speed for the current traffic situation
Vmax	Maximum vehicle speed
W_lane	Distance between two lanes
X	X coordinate of the vehicle
Y	Y coordinate of the vehicle

ANNEX F – Code for the Vehicle Dynamics

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function outputs the derivative of the xy coordinates and the vehicle
% angle using a two-wheel kinematic model.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [dxf,dyf,dpsi] = fcn(l,vf,delta,psi)

dxf=vf.*cos(psi+delta);
dyf=vf.*sin(psi+delta);
dpsi=vf./l.*sin(delta);
```

Listing 5 – traffic_sims/Vehicle dynamics/MATLAB function 1, Matlab code function

ANNEX G – Code for the Distance to the track

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates all the distances between vehicles that are later
% used in the Lane Change  and Car Following Models
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [alpha, d_A, d_AA, d_B, d_C, d_D, d_E, q_A, q_AA, q_B, q_C, q_D, q_E,
Mh]= fcn(x, y, psi, c_A, c_AA, c_B, c_C, c_E, c_D)

n=length(x);        % Number of vehicles

% Order of the Bézier Curves:
nB_A=length(c_A);
nB_AA=length(c_A);
nB_B=length(c_B);
nB_C=length(c_C);
nB_D=length(c_D);
nB_E=length(c_E);

% Distance to the Bézier Curves:
d_A=zeros(n,1);
d_B=zeros(n,1);
d_C=zeros(n,1);
d_D=zeros(n,1);
d_E=zeros(n,1);

% Parameter q for all the Bézier Curves:
q_A=zeros(n,1);
d_AA=zeros(n,1);
q_AA=zeros(n,1);
q_B=zeros(n,1);
q_C=zeros(n,1);
q_D=zeros(n,1);
q_E=zeros(n,1);


alpha=zeros(n,n);   % Matrix with angles among vehicles
Mh=zeros(n,n);      % Matrix with distance among vehicles on the same lane

for j=1:n

    % Distance and parameter q for the Bézier Curve A
    Cx_A=(c_A(1,:)-[zeros(1,nB_A-1) x(j)]);

    Cy_A=(c_A(2,:)-[zeros(1,nB_A-1) y(j)]);

    qall_A=roots(cos(psi(j))*Cx_A+sin(psi(j))*Cy_A);
    aux_A=qall_A>=0 & qall_A<=1.05;

    qsol_A=qall_A(aux_A);

    dsol_A=-sin(psi(j))*polyval(Cx_A,qsol_A)+cos(psi(j))*polyval(Cy_A,qsol_A);

    if isempty(dsol_A)
            q_A(j)=0;
            d_A(j)=0;
    else
        [~,ind_A]=min(abs(dsol_A));
        q_A(j)=real(qsol_A(ind_A));
        d_A(j)=real(dsol_A(ind_A));
```

```
    end

    % Distance and parameter q for the Bézier Curve AA
    Cx_AA=(c_AA(1,:)-[zeros(1,nB_AA-1) x(j)]);

    Cy_AA=(c_AA(2,:)-[zeros(1,nB_AA-1) y(j)]);

    qall_AA=roots(cos(psi(j))*Cx_AA+sin(psi(j))*Cy_AA);
    aux_AA=qall_AA>=0 & qall_AA<=1.05;

    qsol_AA=qall_AA(aux_AA);

    dsol_AA=-
sin(psi(j))*polyval(Cx_AA,qsol_AA)+cos(psi(j))*polyval(Cy_AA,qsol_AA);

    if isempty(dsol_AA)
            q_AA(j)=0;
            d_AA(j)=0;
    else
        [~,ind_AA]=min(abs(dsol_AA));
        q_AA(j)=real(qsol_AA(ind_AA));
        d_AA(j)=real(dsol_AA(ind_AA));
    end

    % Distance and parameter q for the Bézier Curve B
    Cx_B=(c_B(1,:)-[zeros(1,nB_B-1) x(j)]);
    Cy_B=(c_B(2,:)-[zeros(1,nB_B-1) y(j)]);

    qall_B=roots(cos(psi(j))*Cx_B+sin(psi(j))*Cy_B);
    aux_B=qall_B>=0 & qall_B<=1.05;

    qsol_B=qall_B(aux_B);

    dsol_B=-sin(psi(j))*polyval(Cx_B,qsol_B)+cos(psi(j))*polyval(Cy_B,qsol_B);

    if isempty(dsol_B)
            q_B(j)=0;
            d_B(j)=0;
    else
        [~,ind_B]=min(abs(dsol_B));
        q_B(j)=real(qsol_B(ind_B));
        d_B(j)=real(dsol_B(ind_B));
    end

    % Distance and parameter q for the Bézier Curve C
    Cx_C=(c_C(1,:)-[zeros(1,nB_C-1) x(j)]);
    Cy_C=(c_C(2,:)-[zeros(1,nB_C-1) y(j)]);

    qall_C=roots(cos(psi(j))*Cx_C+sin(psi(j))*Cy_C);
    aux_C=qall_C>=0 & qall_C<=1.05;

    qsol_C=qall_C(aux_C);

    dsol_C=-sin(psi(j))*polyval(Cx_C,qsol_C)+cos(psi(j))*polyval(Cy_C,qsol_C);

    if isempty(dsol_C)
            q_C(j)=0;
            d_C(j)=0;
    else
        [~,ind_C]=min(abs(dsol_C));
        q_C(j)=real(qsol_C(ind_C));
        d_C(j)=real(dsol_C(ind_C));
    end
```

```
    % Distance and parameter q for the Bézier Curve D
    Cx_D=(c_D(1,:)-[zeros(1,nB_D-1) x(j)]);
    Cy_D=(c_D(2,:)-[zeros(1,nB_D-1) y(j)]);

    qall_D=roots(cos(psi(j))*Cx_D+sin(psi(j))*Cy_D);
    aux_D=qall_D>=0 & qall_D<=1.05;

    qsol_D=qall_D(aux_D);

    dsol_D=-sin(psi(j))*polyval(Cx_D,qsol_D)+cos(psi(j))*polyval(Cy_D,qsol_D);

    if isempty(dsol_D)
            q_D(j)=0;
            d_D(j)=0;
    else
        [~,ind_D]=min(abs(dsol_D));
        q_D(j)=real(qsol_D(ind_D));
        d_D(j)=real(dsol_D(ind_D));
    end

    % Distance and parameter q for the Bézier Curve E
    Cx_E=(c_E(1,:)-[zeros(1,nB_E-1) x(j)]);
    Cy_E=(c_E(2,:)-[zeros(1,nB_E-1) y(j)]);

    qall_E=roots(cos(psi(j))*Cx_E+sin(psi(j))*Cy_E);
    aux_E=qall_E>=0 & qall_E<=1.05;

    qsol_E=qall_E(aux_E);

    dsol_E=-sin(psi(j))*polyval(Cx_E,qsol_E)+cos(psi(j))*polyval(Cy_E,qsol_E);

    if isempty(dsol_E)
            q_E(j)=0;
            d_E(j)=0;
    else
        [~,ind_E]=min(abs(dsol_E));
        q_E(j)=real(qsol_E(ind_E));
        d_E(j)=real(dsol_E(ind_E));
    end

    % Distance to other vehicles:
    h_j=sqrt((x(j)-x).^2+(y(j)-y).^2);
    Mh(j,:)=h_j;

    % Angle with respect to other vehicles:
    alpha_j=atan2(y-y(j),x-x(j))-psi(j);
    alpha(j,:)=alpha_j;

end

end
```

Listing 6 - traffic_sims/Distance To The Track, Matlab code function

ANNEX H – Code for the Track Selection

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function outputs the vehicle track selection protocol. The decision
% of taking the highway off-ramp is taken randomly.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [d,  posA, newtrack] = fcn(d_A, d_AA, d_B, d_C, d_D, d_E, q_A, q_AA,
q_B, q_C, q_D, q_E, percentage_C, lane, track)

n=length(d_A);          % Number of vehicles.
d=zeros(n,1);           % Distance to the current Bézier Curve.
newtrack=zeros(n,2);    % Current track.
posA=zeros(n,1);        % Position of the vehicles in track A.

for j=1:n

    % Track Selection Protocol:

    %For Track A
    if track(j,1)==1 && q_A(j)<1
        d(j)=d_A(j);
        newtrack(j,1)=1;
        newtrack(j,2)=track(j,2);

    elseif track(j,1)==1 && q_A(j)>=1 && lane(j)==1
        newtrack(j,1)=track(j,2);

    elseif track(j,1)==1 && q_A(j)>=1 && lane(j)~=1
        newtrack(j,1)=2;

    % For Track AA
    elseif track(j,1)==2 && q_AA(j)<1
        d(j)=d_AA(j);
        newtrack(j,1)=2;

    elseif track(j,1)==2 && q_AA(j)>=1
        d(j)=d_D(j);
        newtrack(j,1)=6;

    % For Track B
    elseif track(j,1)==3 && q_B(j)<1
        d(j)=d_B(j);
        newtrack(j,1)=3;

    elseif track(j,1)==3 && q_B(j)>=1
        d(j)=d_E(j);
        newtrack(j,1)=4;

    % For Track E
    elseif track(j,1)==4 && q_E(j)<1
        d(j)=d_E(j);
        newtrack(j,1)=4;

    elseif track(j,1)==4 && q_E(j)>=1
        d(j)=d_C(j);
        newtrack(j,1)=5;

    % For Track C
    elseif track(j,1)==5 && q_C(j)<1
        d(j)=d_C(j);
```

```
        newtrack(j,1)=5;

    elseif track(j,1)==5 && q_C(j)>=1
        d(j)=d_A(j);
        newtrack(j,1)=1;
        aux=rand;

        if aux<=percentage_C
            newtrack(j,2)=3;
        else
            newtrack(j,2)=2;
        end

    % For Track D
    elseif track(j,1)==6 && q_D(j)<1
        d(j)=d_D(j);
        newtrack(j,1)=6;

    elseif track(j,1)==6 && q_D(j)>=1
        d(j)=d_A(j);
        newtrack(j,1)=1;

        aux=rand;

        if aux<=percentage_C
            newtrack(j,2)=3;
        else
            newtrack(j,2)=2;
        end

    end

   % Obtaining of the position of the vehicle in track A
   if newtrack(j,1)==1 && q_A(j)<=1
       posA(j)=q_A(j)*(1000-100*cos(pi/18));
   elseif newtrack(j,1)==2 && q_AA(j)<=1
       posA(j)=1000-100*cos(pi/18)+q_AA(j)*100*cos(pi/18);
   else
       posA(j)=0;
   end


end

end
```

Listing 7 - traffic_sims/Track Selection, Matlab Code function

ANNEX I – Code for the Car Following

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function outputs the velocity of the vehicles taking into
% consideration the distance to the nearest front vehicle using a model
% based on ACC.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [hc, vf_desired]  = fcn(Vmax, hmax, ho, T, Mh, alpha, q_A, q_C, track,
lane_in)

eps=0.2;
n=length(Vmax);        % Number of vehicles
hc=zeros(n,1);         % Safety distance to the leader vehicle
vf_desired=zeros(n,1);         % Velocity output
h=zeros(n,1);          % Current distance to the leader vehicle
hSigned=zeros(n,1);    % Signed distance matrix. Vehicle in the back have a
negative distance.

    for j=1:n

        for k=1:n

            % To disable the use of the car following function for vehicles
            % on different lanes:

            if lane_in(j)~=lane_in(k)
                Mh(j,k)=0;
            end

            % To disable the use of the car following function for vehicles
            % circulating on different tracks:

            if (track(j,1)==5 && track(k,1)==6) || (track(j,1)==6 ...
                    && track(k,1)==5) || (track(j,1)==2 && ...
                    track(k,1)==3) || (track(j,1)==3 && track(k,1)==2) ...
                    || (track(j,1)==5 && track(k,1)==1) || ...
                    (track(j,1)==1 && track(k,1)==5) ...
                    || (track(j,1)==4 && track(k,1)==6) || ...
                    (track(j,1)==6 && track(k,1)==4)
                Mh(j,k)=0;
            end

            % Yield behaviour on the end of track C:

            if track(k,1)==1 && q_A(k)<=0.1 && q_A(k)>0 && ...
                    track(j,1)==5 && q_C(j)>0.6 && q_C(j)<0.80 && ...
                    lane_in(k)==1
                Vmax(j)=0;
            end

        end

        % Calculation of the closest front car

        hSigned(:)=sign(cos(alpha(j,:))).*Mh(j,:);    % The cars behind will have
negative distance with respect to the current vehicle.
        ind=find(hSigned(:)<hmax(j) & hSigned(:)>0); % We find those who are in
front and that have a distance smaller than the maximum distance.
```

```
        if isempty(ind)
            h(j)=hmax(j)+1; % If there isn't any car in front of the current
vehicle, we force a big enough ficticious distance.
        else
            h(j)=min(hSigned(ind)); % The distance of the nearest front car in
respect to the current vehicle will be the minimum value of the distances of the
cars ahead.
        end

        % I-ACC MODEL
        hc(j)=ho+T*Vmax(j);

        vf_desired(j)=Vmax(j).*(h(j)-hc(j)>=eps)+...
            (Vmax(j)+(h(j)-hc(j))/T).*(h(j)-hc(j)<=-eps)+...
            (Vmax(j)-1/4/eps/T.*(h(j)-hc(j)-eps).^2).*(abs(h(j)-hc(j))<eps);

    end

end
```

Listing 8 -traffic_sims/Car Following I-ACC, Matlab code function

ANNEX J – Code for the Lane Changing Model

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates the Lane Change behaviour of the vehicles using a
% modification of the MOBIL model, considering the change in velocity of all
% the affected vehicles in the process of lane changing.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function lane  = fcn(Mh, track, vmax, vf, alpha, polit, bias, epsilon, n_lanes,
action_radius, action_radius2, lane_in)

n=length(vmax);                 % Number of vehicles.
lane=ones(n,1);                % Current lane.
h_signed=zeros(n,1);            % Signed distance matrix. Vehicle in the back
have a negative distance.
car_indexing=1:n;               % Matrix with the index number for each vehicle.

index_Front_Same=zeros(n,1);    % Index of the vehicles in front in the same
lane.
index_Front_Up=zeros(n,1);      % Index of the vehicles in front in the upper
lane.
index_Front_Down=zeros(n,1);    % Index of the vehicles in front in the lower
lane.
index_Back_Up=zeros(n,1);       % Index of the vehicles in the back in the upper
lane.
index_Back_Down=zeros(n,1);     % Index of the vehicles in the back in the lower
lane.

vj_new_up=zeros(n,1);           % Speed of the current vehicle if a lane change
to an upper lane ocurred.
vj_new_down=zeros(n,1);         % Speed of the current vehicle if a lane change
to an lower lane ocurred.

vp_new_up=zeros(n,1);           % Speed of the closest back vehicle in the upper
lane if a lane change to an upper lane ocurred.
vp_new_down=zeros(n,1);         % Speed of the closest back vehicle in the lower
lane if a lane change to an lower lane ocurred.

vp_up=zeros(n,1);               % Speed of the closest back vehicle in the upper
lane.
vp_down=zeros(n,1);             % Speed of the closest back vehicle in the lower
lane.

MOBIL_R_L=zeros(n,1);           % Logical parameter for the lane change condition
from right to left.
MOBIL_L_R=zeros(n,1);           % Logical parameter for the lane change condition
from left to right.

forbidden_radius_up=7.5;        % If a car is inside this radius, upper lane
change is disabled.
forbidden_radius_down=7.5;      % If a car is inside this radius, lower lane
change is disabled.

for j=1:n

    h_signed(:)=sign(cos(alpha(j,:))).*Mh(j,:);

    % Obtaining of the index of the vehicles in front, separated by lanes:
    cars_Front_Up=   car_indexing((h_signed(:)<action_radius & h_signed(:)>0) &
lane_in==lane_in(j)+1);
```

```
    cars_Front_Same= car_indexing((h_signed(:)<action_radius & h_signed(:)>0) &
lane_in==lane_in(j));
    cars_Front_Down= car_indexing((h_signed(:)<action_radius2 & h_signed(:)>0) &
lane_in==lane_in(j)-1);

    % Closest front vehicle in the upper lane:
    if isempty(cars_Front_Up)
        index_Front_Up(j)=0;
        vj_new_up(j)=vmax(j);
        min_Front_Up=500;
    else
        [min_Front_Up, I_1]= min(Mh(j, cars_Front_Up));
        index_Front_Up(j)= cars_Front_Up(I_1);
        vj_new_up(j)= min(vmax(j), vf(index_Front_Up(j)));
    end

    % Closest front vehicle in the same lane:
    if isempty(cars_Front_Same)
        index_Front_Same(j)=0;
    else
        [~, I_2]= min(Mh(j, cars_Front_Same));
        index_Front_Same(j)= cars_Front_Same(I_2);
    end

    % Closest front vehicle in the lower lane:
    if isempty(cars_Front_Down)
        index_Front_Down(j)=0;
        vj_new_down(j)=vmax(j);
        min_Front_Down=500;
    else
        [min_Front_Down, I_3]= min(Mh(j, cars_Front_Down));
        index_Front_Down(j)= cars_Front_Down(I_3);
        vj_new_down(j)= min(vmax(j), vf(index_Front_Down(j)));
    end

    % Obtaining of the index of the vehicles in the back, separated by lanes:
    cars_Back_Up=   car_indexing((h_signed(:)>-action_radius & ...
        h_signed(:)<0) & lane_in==lane_in(j)+1);
    cars_Back_Down= car_indexing((h_signed(:)>-action_radius2 & ...
        h_signed(:)<0) & lane_in==lane_in(j)-1);

    % Closest back vehicle in the upper lane:
    if isempty(cars_Back_Up)
        index_Back_Up(j)=0;
        vp_new_up(j)=0;
        vp_up(j)=0;
        min_Back_Up=500;
    else
        [min_Back_Up, I_4]= min(Mh(j, cars_Back_Up));
        index_Back_Up(j)= cars_Back_Up(I_4);
        vp_new_up(j)= min(vmax(index_Back_Up(j)), vj_new_up(j));
        vp_up(j)=vf(index_Back_Up(j));
    end

    % Closest back vehicle in the lower lane:
    if isempty(cars_Back_Down)
        index_Back_Down(j)=0;
        vp_new_down(j)=0;
        vp_down(j)=0;
        min_Back_Down=500;
    else
        [min_Back_Down, I_5]= min(Mh(j, cars_Back_Down));
        index_Back_Down(j)= cars_Back_Down(I_5);
        vp_new_down(j)=min(vmax(index_Back_Down(j)), vj_new_down(j));
```

```
        vp_down(j)=vf(index_Back_Down(j));
    end

    % Closest vehicle in different lane:
    Mh_min_Up= min(min_Front_Up, min_Back_Up);
    Mh_min_Down=min(min_Front_Down, min_Back_Down);

    % Change in speed if an upper lane change occurs:
    delta_vj_up= vj_new_up(j)-vf(j);
    delta_vp_up= vp_new_up(j)-vp_up(j);

    % Change in speed if an lower lane change occurs:
    delta_vj_down= vj_new_down(j)-vf(j);
    delta_vp_down= vp_new_down(j)-vp_down(j);

    % Obtaining of the logical values for the lane change. If equal to 1,
    % lane change causes an overall positive gain in speed:
    MOBIL_R_L(j)= delta_vj_up+polit*delta_vp_up>epsilon;
    MOBIL_L_R(j)= delta_vj_down+polit*delta_vp_down>epsilon-bias;

    %%%%%%%%%%%%%%%%%%%%%%%% LANE CHANGE MODEL %%%%%%%%%%%%%%%%%%%%%%%%%%

    % Upper lane change
    if (lane_in(j)>=1 && lane_in(j)<n_lanes) && MOBIL_R_L(j)==1 && ...
            Mh_min_Up>forbidden_radius_up
        lane(j)=lane_in(j)+1;

    % Lower lane change
    elseif (lane_in(j)>1 && lane_in(j)<=n_lanes) && MOBIL_L_R(j)==1 && ...
            Mh_min_Down>forbidden_radius_down
        lane(j)=lane_in(j)-1;

    else
        lane(j)=lane_in(j);
    end

    % If vehicle is circulating through a one-laned track, lane change is
    % disabled:
    if track(j,1)==3 || track(j,1)==4 || track(j,1)==5
        lane(j)=1;
    end

end

end
```

Listing 9 - traffic_sims/Lane Changing Model, Matlab code function

ANNEX K – Code for the Lane Changing Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function calculates the path taken by the vehicle when a lane change
% occurs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [d_lane, i_out] = fcn(d, w_lane, maxstepsize, lane, vmax, vf, tm_min,
tm_max, d_lane_in, i_in)

n=length(d);         % Number of vehicles.
d_lane=zeros(n,1);   % Distance between the current lane and lane 1.
i_out=ones(n,2);     % Just a counter.
d_goal=zeros(n,1);   % Distance between the goal lane and lane 1.

for j=1:n

    % Manouver time (lane change time):
    tm=tm_min+(vf(j)/(vmax/3.6))*(tm_max-tm_min);

    % Change in distance in every step during the lane change
    min_delta_d= (w_lane)/tm*maxstepsize;

    % Goal distance to lane 1. If lane change is to lane 2, then d_goal is
    % equal to w_width. If lane change is to lane 3, then d_goal is
    % equal to 2*w_width.

    d_goal(j)=w_lane*(lane(j)-1);

    % Lane Change Function for an upper lane change:
    if d_lane_in(j)<d_goal(j)-min_delta_d
        d_lane(j)=d_lane_in(j)+min_delta_d*i_in(j,1);
        i_out(j,1)=i_in(j,1)+1;
        i_out(j,2)=0;

    % Lane Change Function for an lower lane change:
    elseif d_lane_in(j)>d_goal(j)+min_delta_d
        d_lane(j)=d_lane_in(j)-min_delta_d*i_in(j,2);
        i_out(j,1)=0;
        i_out(j,2)=i_in(j,2)+1;

    else
        d_lane(j)= d_goal(j);
    end

end
```

Listing 10 - traffic_sims/Lane Change Function, Matlab code function

ANNEX L – Code for the animation file

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is the animation file for the traffic simulator.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all
clc

grey1=[1 1 1]*0.85;
grey2=[1 1 1]*0.7;
blue=[0.0 0.4470 0.7410];
orange=[0.9856 0.7372 0.2537];
l=4;
w=2.5;
lw=1;
lw_squares=1.5;
c_squares='b';
vid = VideoWriter('sims_traffic25','MPEG-4');
vid.FrameRate = 15;
myVideo.Quality = 20;

set(figure(1), 'Position', [0 0 1400 650]); hold on;

%%%%%%%%%%%%%%%%%%%%%%%% FINAL ANIMATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GENERAL VIEW
subplot(3,7,[1 7],'Color','w')

plot(B_A(1,:),B_A(2,:),'k','Linewidth',3, 'color', '#828282')
hold on
plot(B_AA(1,:),B_AA(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A2(1,:),B_A2(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A3(1,:),B_A3(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_C(1,:),B_C(2,:),'k','Linewidth', 3, 'color', '#828282')
plot(B_B(1,:),B_B(2,:),'k','Linewidth', 3, 'color', '#828282')

square_1_x= [-450 -450 -350 -350 -450];
square_1_y= [-20 20 20 -20 -20];
txt1='1';
text(-400, -30, txt1, 'color', c_squares);

plot(square_1_x, square_1_y, 'color', c_squares, 'Linewidth', lw_squares);

square_2_x= [-300 -300 300 300 -300];
square_2_y= [-20 20 20 -20 -20];
txt1='2';
text(0, -30, txt1, 'color', c_squares);

plot(square_2_x, square_2_y, 'color', c_squares, 'Linewidth', lw_squares);

square_3_x= [350 350 450 450 350];
square_3_y= [-20 20 20 -20 -20];
txt1='3';
text(400, -30, txt1, 'color', c_squares);

plot(square_3_x, square_3_y, 'color', c_squares, 'Linewidth', lw_squares);

axis equal
axis off
axis([-500 500 -20 20]);
```

```
% HIGHWAY VIEW
subplot(3,7,[8 11],'Color','w')

plot(B_A(1,:),B_A(2,:),'k','Linewidth',3, 'color', '#828282')
title('2 - HIGHWAY SECTION', 'color', c_squares);
hold on
plot(B_A2(1,:),B_A2(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A3(1,:),B_A3(2,:),'k','Linewidth',3, 'color', '#828282')

axis equal
axis off
axis([-300 300 -20 20]);

subplot(3,7,[15 16],'Color','w')

plot(B_A(1,:),B_A(2,:),'k','Linewidth',3, 'color', '#828282')
title('1 - ON-RAMP', 'color', c_squares);
hold on
plot(B_A2(1,:),B_A2(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A3(1,:),B_A3(2,:),'k','Linewidth',3, 'color', '#828282')

plot(B_C(1,:),B_C(2,:),'k','Linewidth', 3, 'color', '#828282')

axis equal
axis off
axis([-450 -350 -20 20]);

subplot(3,7,[17 18],'Color','w')

plot(B_A(1,:),B_A(2,:),'k','Linewidth',3, 'color', '#828282')
title('3 - OFF-RAMP', 'color', c_squares);
hold on
plot(B_AA(1,:),B_AA(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A2(1,:),B_A2(2,:),'k','Linewidth',3, 'color', '#828282')
plot(B_A3(1,:),B_A3(2,:),'k','Linewidth',3, 'color', '#828282')

plot(B_B(1,:),B_B(2,:),'k','Linewidth', 3, 'color', '#828282')

axis equal
axis off
axis([350 450 -20 20]);

% Speed graphic
subplot(3,7,[13 14],'Color','w')
plot(tout, simsvf);
title('Vehicles speed');
axis manual
axis([0 tfinal 0 40])
xlabel('Time [s]');ylabel('Speed [m/s]')
t1 = line([0 0], ylim, 'Linewidth', 2, 'Color', c_squares);

% Position graphic
subplot(3,7,[20 21],'Color','w')
plot(tout, simsPosA);
title('Vehicles position');
axis([0 tfinal 0 1000])
xlabel('Time [s]');ylabel('Pos. on road [m]')
t2 = line([0 0], ylim, 'Linewidth', 2, 'Color', c_squares);

ind=1;
carsX=[];
carsY=[];
for i=1:n
```

```
    xtail=[simsx(ind,i)-l*cos(simspsi(ind,i))-w*sin(simspsi(ind,i)) simsx(ind,i)-
l*cos(simspsi(ind,i))+w*sin(simspsi(ind,i))];
    ytail=[simsy(ind,i)-l*sin(simspsi(ind,i))+w*cos(simspsi(ind,i)) simsy(ind,i)-
l*sin(simspsi(ind,i))-w*cos(simspsi(ind,i))];
    carsX=[carsX; simsx(ind,i)  xtail];
    carsY=[carsY; simsy(ind,i)  ytail];
end

subplot(3,7,[1 7],'Color','w')
    text_title = title(['Simulation time ' num2str(round(tout(1),2)) 's']);
    cars1=patch(carsX',carsY',blue);

subplot(3,7,[8 11],'Color','w')
    cars2=patch(carsX',carsY',blue);

subplot(3,7,[15 16],'Color','w')
    cars3=patch(carsX',carsY',blue);

subplot(3,7,[17 18],'Color','w')
    cars4=patch(carsX',carsY',blue);

j=30;
fps=15;
for i=0:1/fps:tout(end)
    if i==tout(end)
        ind=length(tout);
    else
        ind=find(tout>i,1,'first');
    end
    carsX=[];
    carsY=[];
    for i=1:n
        xtail=[simsx(ind,i)-l*cos(simspsi(ind,i))-w*sin(simspsi(ind,i)) ...
            simsx(ind,i)-l*cos(simspsi(ind,i))+w*sin(simspsi(ind,i))];
        ytail=[simsy(ind,i)-l*sin(simspsi(ind,i))+w*cos(simspsi(ind,i)) ...
            simsy(ind,i)-l*sin(simspsi(ind,i))-w*cos(simspsi(ind,i))];
        carsX=[carsX; simsx(ind,i)  xtail];
        carsY=[carsY; simsy(ind,i)  ytail];
    end

    subplot(3,7,[1 7],'Color','w')
        set(cars1,'XData',carsX','YData',carsY')
    subplot(3,7,[8 11],'Color','w')
        set(cars2,'XData',carsX','YData',carsY')
    subplot(3,7,[15 16],'Color','w')
        set(cars3,'XData',carsX','YData',carsY')
    subplot(3,7,[17 18],'Color','w')
        set(cars4,'XData',carsX','YData',carsY')

    subplot(3,7,[13 14],'Color','w')
        set(t1,'XData', [tout(ind) tout(ind)]);

    subplot(3,7,[20 21],'Color','w')
        set(t2,'XData', [tout(ind) tout(ind)]);

    if i==tout(end)
        set(text_title,'String',['Simulation time ' num2str(round(tout(ind),2))
's']);
    else
        set(text_title,'String',['Simulation time '
num2str(round(tout(ind),2)+0.01) 's']);
    end
    open(vid);
    f = getframe(gcf);
```

```
        writeVideo(vid,f);
end
close(vid);
```

Listing 11 – traffic_sims_animation.m Matlab file