Technische Universität Braunschweig

INSTITUT FÜR FAHRZEUGTECHNIK
PROF. DR.-ING. F. KÜÇÜKAY

Redesign of a statistical driver model for the 3D-method

MASTER THESIS

Miguel Pradini

Braunschweig (Germany) in June 2013

Technische Universität Braunschweig
Institut für Fahrzeugtechnik
Direktor: Prof. Dr.-Ing. Ferit Küçükay
Betreuer: Dipl.-Ing. Benedikt Weiler

DECLARATION

Declaration in Lieu of Oath / Eidesstattliche Erklärung

I hereby declare in lieu of oath that I composed the following thesis independently and with no additional help other than the literature and means referred to.

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Literatur und Hilfsmittel angefertigt habe.

12.06.2013	
Date/Datum	Signature/Unterschrift

ABSTRACT

In the field of customer use of the development of vehicles, the so called 3D method allows the representation of the customer behavior through the parameter space. Building a statistical driver to work with this method is a challenging problem given the limitations of available data and the high statistical accuracy requirements in the simulation of strongly correlated variables. Previous students have failed to group the information in a way that preserves the information available.

Through this document, the different simulation strategies that were built will be described with focus on the one that proved to be the most successful. This technique is based in the creation of variable length blocks of information to be resampled en reorganized during the simulation. The statistical resemblance among the real and the simulated roads leaves us one step closer to the creation our statistical driver that will be used for simulation and optimization purposes.

INDEX

GLOSSARY

IAE - Institute of automotive engineering at the Technical University of Braunschweig

3D Method - Method developed in the Institut of automotive engineering (IAE) that takes into account the driver, the driven vehicle and the driving environment

Parameter space - All the possible combinations of factor of the 3D method

3V - The three main variables considered in this Project: slope (%), lateral acceleration (m/s^2) and speed (km/h)

Oriented profiles - profiles with the values of the next objective class that the driver is going to reach for the corresponding variable

3-coordinates - The classes of the 3V for a given point- As an example, a set of coordinates of (10, 22, 7) is equivalent to a slope class of 10, a lateral acceleration class of 22 and a speed class of 7

Classification vector - Vector that contains the limit values to decide the class which corresponds to a variable value. The classification is the process that assigns such classes and the declassification its inverse.

Peck, rhoeck & veck vectors - vectors that contain the oriented profiles of the slope, lateral acceleration and speed respectively

Neutral class - This a class that contains the zero and divides the classes of a symmetrical variable (slope and lateral acceleration). For the lateral acceleration, I also name it the no-curve class.

1 INTRODUCTION

1.1 MOTIVATION AND GOALS OF THE PROJECT

The final objective of this Project is to generate an infinite road which characteristics match the available data and that will be used to statistically predict in real time how the road ahead is likely to be and which will be the corresponding next maneuver of the driver.

This will be achieved through the prediction of 3 variables, the slope and lateral acceleration (curves) that provide information about the physical features of the road; and the speed, that includes also information about the driven vehicle, legal limitation and the driver's behavior.

In the future, this program will be part of a bigger simulation model that will be used to simulate a whole car and its interactions with the driving environment and the driver and aid in the optimization and the early design of many vehicle elements.

1.2 STATE OF THE ART

Although all the ideas exposed in this project were of my own, this project is the continuation of the work of two previous students that tried unsuccessfully to finish it. Although they did not reach the goals that had been defined for their projects, the ideas they implemented and their conclusions were of vital importance for the later success of this project helping me to find the right direction.

The reasons of that failure were the strong limitations in terms of computational memory, a very limited set of available real data and, at the same time, high requirements of computational speed and statistical accuracy with heavily correlated variables. Even if almost all their code had to be discarded, I would like to thank them for their efforts without which this project would have probably failed too. The main difficult is with the existent work at the IAE is that they were not considering the 3 variables together before, incrementing substantially the complexity of the problem.

1.3 3D METHOD

The increasing consumer demands make fundamental the development of more efficient strategies of operation of all the components of a car. Simulation has turned out to be essential for the constant improvement and tuning of the car manufacturing parameters. In this regard, the 3D method developed in the IAE considers not only the internal factors of an automotive vehicle but also the road characteristics and the driver behavior, allowing more sophisticated control strategies.

The 3D method is based on the 3D parameter space composed by the **D**river, the **D**riving environs and the **D**riven vehicle **[1]**. This technique takes into account more information than the common approaches and it has been successfully applied to intelligent control strategies to reduce the CO2 emissions up to 23.6% **[2]** among others optimization examples.

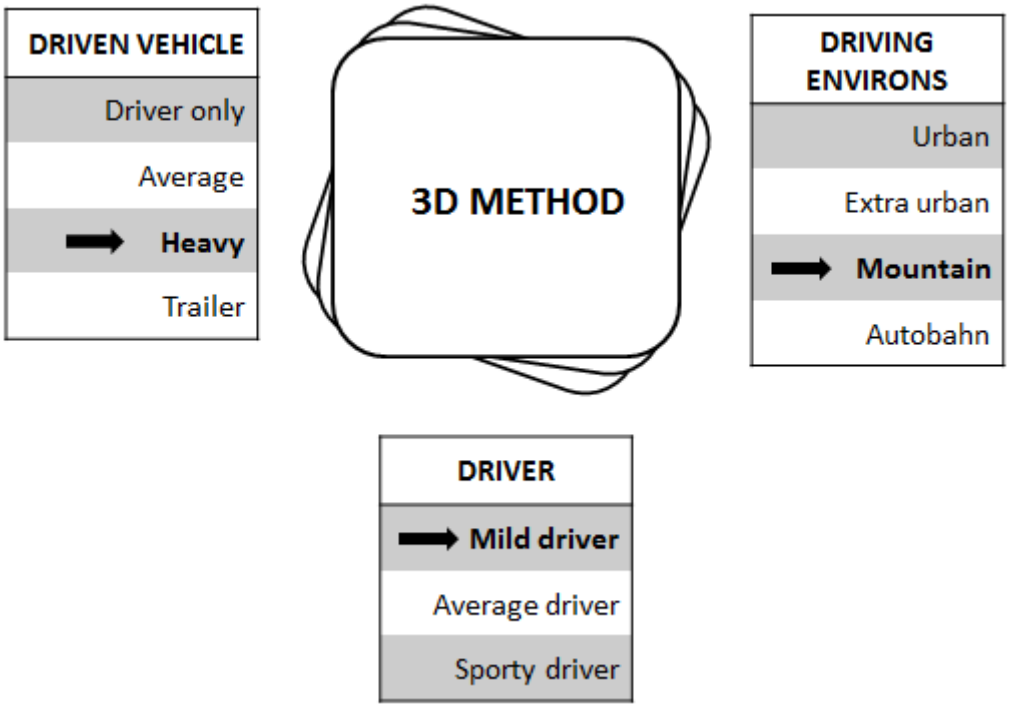


Figure 1.1 - 3D Method example

In the example, a mild driver is driving a heavy vehicle through a mountain road and its driving will be aided by the statistical data that matches such combination of factors.

As shown in figure 1.1, 3 different kinds of driver, 4 kinds of road and 4 kinds of loads are considered in this example model, producing a total output of 48 possible combinations. Each real case will be fit into a combination of the factors of the parameter space. Therefore, this implies that our real driver may not behave always as a sporty driver but it can be stated that he drives in a sporty way a certain percentage (%) of the time and in an average way the rest of the time, as an example. So, any given case can be interpolated using the 48 different cases that define the space parameter.

In this context, the development and validation of a road simulation program relates directly to one of the three parts of the 3D method, the driving environment or driving environs. Being able to predict statistically road events, such as curves or slopes, and driver reactions is useful for developing, improving and testing intelligent control systems. It must be pointed out that gathering real road data is an expensive and long procedure, forcing to use simulation alternatives as often as possible. Another advantage is the fact that the statistical simulation can run forever, simulations are possible for any road or length without taking time into account. To some extent, the goal is to be able to predict which are going to be the next maneuvers before they happen, so a more efficient operation is possible even in traffic conditions.

2 DATA PROCESSING

2.1 CLASSIFICATION VECTORS

2.1.1 SLOPE

Slope vector

Number of classes: 22 classes

Min value: -0.03 m/m (-30%)

Delta interval: 0.03

Max value: 0.03 m/m (+30%)

Middle class: [-0.03, 0.03]

Table 2.1 - Slope classification vector

Class	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Range	<-0.3	-0.27 -0.3	-0.24 -0.27	-0.21 -0.24	-0.18 -0.21	-0.15 -0.18	-0.12 -0.15	-0.09 -0.12	-0.06 -0.09	-0.03 -0.06	**-0.03** **0.03**
Class	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
Range	0.03 0.06	0.06 0.09	0.09 0.12	0.12 0.15	0.15 0.18	0.18 0.21	0.21 0.24	0.24 0.27	0.27 0.3	>0.3	**fictional**

Justification

It is important to have an odd number of classes so that there is a central class that contains the zero slope state. That middle class is used for the beginning of the simulation.

In addition, slope changes lengths are usually much longer than curvature or speed changes, reaching in extreme cases files that could contain no relevant slope change at all Like in the file: *"VW_Touareg_3LTDI_6AT_V_SC_B_00532.mat"* This fact makes more important to deal with the beginnings and the endings of the files in a way that no information is lost in order to preserve the statistics accuracy. Being more specific, had you a long even road which no slope change, without making any adjustment no information could be retrieved from such a file since there is no previous slope class to address to the current slope. Therefore, the frequency of the slope changes in the simulated road would be higher as a result of deleting low frequency information.

For this purpose, an additional slope class (20) has been added to the classification that allows us to keep track of the endings of the files and linking them to the beginnings of other files. This has been built with the idea on mind that only one class of data files ('land', 'city' or 'mountain') will be loaded into the program at each simulation.

2.2 SENSORS DATA PROCESSING AND FILTERING

2.2.1 SLOPE

The slope is calculated from derived the height data obtained from barometric pressure sensors according to the following formula **[3]**:

$$p = p_0 + \left(1 - \frac{L \cdot h}{T0}\right)^{\frac{g \cdot M}{R \cdot L}} \approx p_0 \cdot exp\left(-\frac{g \cdot M \cdot h}{R \cdot T0}\right) \quad \textbf{(2.1)}$$

Where the constant parameters are described below:

Equation 2.1 - Pressure dependence on the height

Parameter	Description	Value
P_0	sea level standard atmospheric pressure	101325 Pa
L	temperature lapse rate	0-0065 K/m
T_0	sea level standard temperature	288.15 K
g	earth-surface gravitational acceleration	9.80665 m/s2
M	molar mass of dry air	0.0289644 kg/mol
R	universal gas constant	8.31447 J/(mol·K)

There are alternatives to obtain the slope based on the relationship among the different accelerations in the car, using at the end the angle between the vertical and horizontal acceleration discounting previously the part caused by the engine of the car itself. However these alternatives lead to extra steps of filtering and less accuracy at the reconstruction of the height, since this has to be calculated later from the slope.

Instead of that, the formula shown previously reaches is more exact, however it still requires to be filtered before being able to obtain the slope:

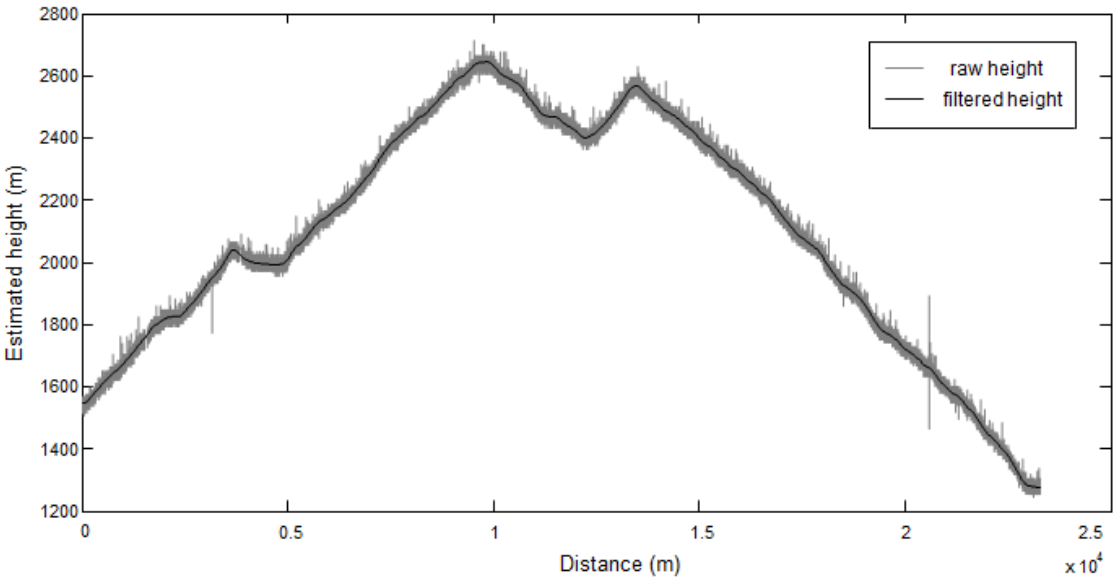


Figure 2.1 - Influence of the filtering on the height estimation

After that step, the slope is simply calculated by deriving the height using a distance base of 1 meter per point and interpolating the obtained slope with it.

These frequency filtering algorithms, in combination with the limitation of the sensors, cause abnormal measurements that must be filtered. There is also another phenomenon to take care of, because in the way these pressure sensors work, whenever a door of the car is open it produces wrong measurements. This takes place usually at the beginning and the end of the files and it is easily observed in the figure 2.2.

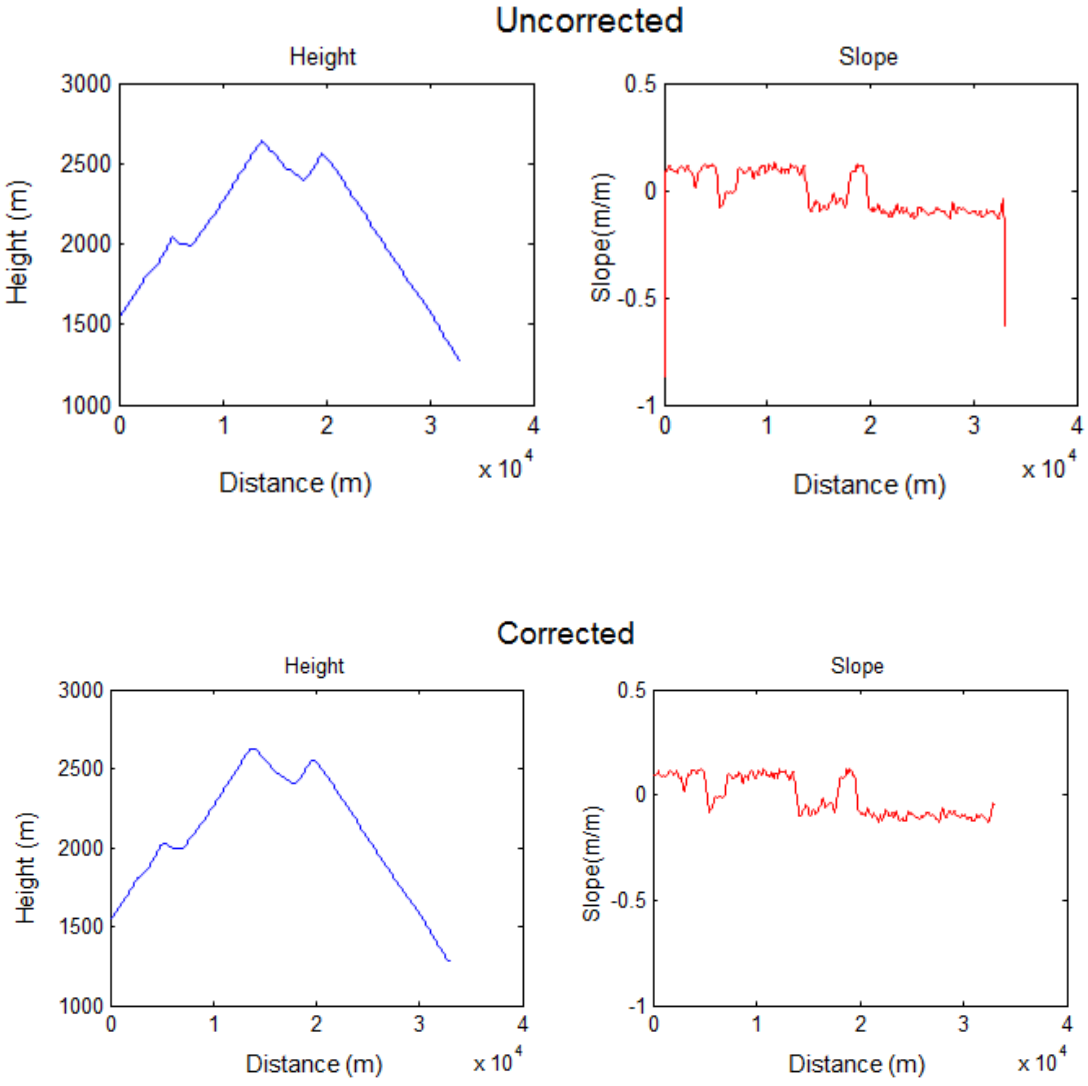


Figure 2.2 - Filtering of the slope

However, it happens also in the middle of the files to a much smaller scale due to the filtering algorithms. Regarding this issue, extremely fast incoherent variations of the slope are easily detected at the extremes of the files. The way I have chosen to deal with them is to calculate the derivative of the slope and substitute all the slope points which a high derivative value (the threshold value was set at 0.0001 after plotting the results with a large number of files) with the appropriate values.

For the beginnings of the files, all those points receive the value of the first space point with a low derivative, in other words, the first point which is not changing its slope function value too fast. The same strategy is applied for the ending of the files where the values deleted are replaced by the value of the last valid point. The strategy for the in between points is a bit different, since those points are substituted by a linear transition of the values at the extremes of the suppressed interval. The following graphs display the result after the filtering.

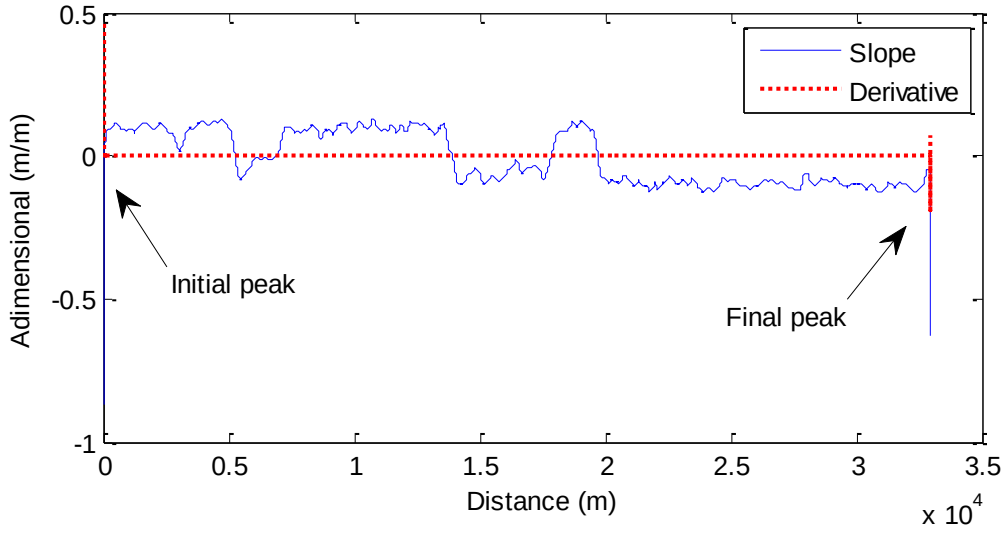


Figure 2.3 - Slope filtering: derivative peaks

It should be noticed that the height is also being altered at those points where the slope is being altered to keep a reasonable correlation between both. Compared to the length of the files, only a small fraction of the total points is altered through this process. Had not this approach been taken, it would result in obtaining unreal slope changes at the beginnings and ends of the files which do not correspond to the real tracks.

The strategy followed in the code that I received was limited to changing all the extreme values to the maximal or minimal class which did not solve the problem and would cause these unreal changes to be picked as real and fitted to an existent class.

2.3 CLASSIFICATION

2.3.1 SPEED

This section was made as part of a team task with another two students -Aitor Diaz de Cerio and José Ángel Lacalzada-. The classification of the speed was the first task we received before we started with our differentiated assignments. The main difficult of this task is that, unlike the curves, there is not a certain value (namely, straight tracks between curves) that repeats periodically and allows us to divide the problem into smaller ones. Besides there are two additional problems: the interaction of external variables that do not correspond to the desires of the driver (such as the oscillations caused by gear shifts) and the need of identifying not only the global changes of speed but also the changes of intensity with which the driver is speeding up.

Detecting the relevant points

The first step of the classification is detecting the local maximum and minimum values because all the interesting points belong to that group, with the only exception of the changes of intensity of the acceleration.

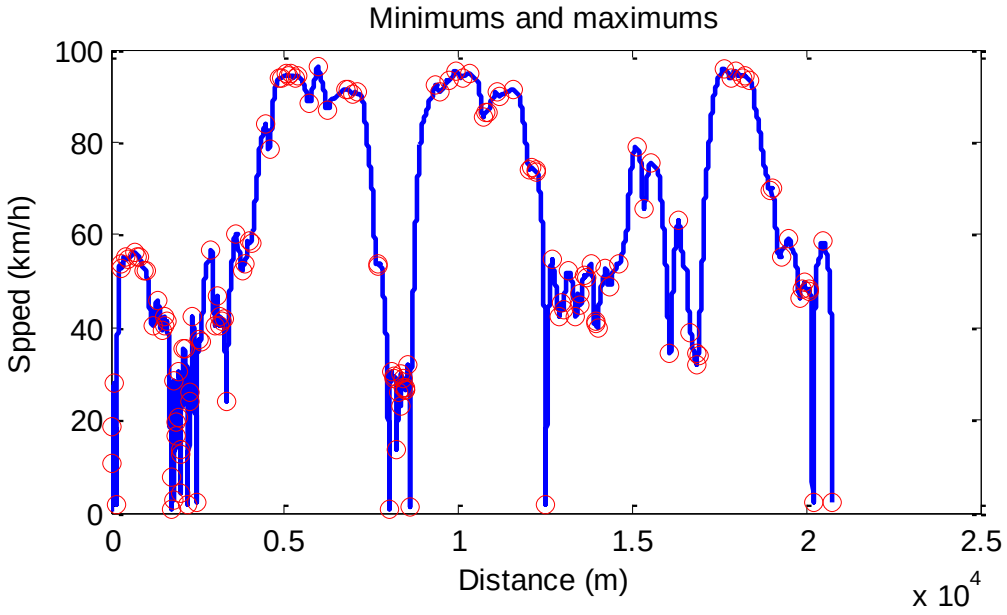


Figure 2.4 - Identifying the extreme points

However, not all those points are of interest, so to avoid picking two points that are too close in value they are picked only when there is a “meaningful” change of speed. This delta or gap is set to be 5km/h, although there is one additional complication: if a certain section starts for example at 60km/h and the objective is to reach 120km/h before slowing down again, it is not desired to observe a change in the oriented speed profile (veck vector) each 5 km/h. Instead, only one single change from 60 to 120 km/h would be considered. However, while going over the vector with the speed if a change of tendency is encountered, it I not possible to know yet looking only at past values if the speed that is going to be reached is a relevant change of speed (in which case the previous point should be picked) or just a small variation that will soon end.

To some extent, the problem is asking us to store a “candidate value” and only once it has been confirmed that there is effectively a change of tendency with a speed gap greater than 5km/h it is allowed to pick that previous value. This idea is the approach that was followed in the initial group task, by constantly updating the value of the potential value along the vector and only keeping the latest values before relevant changes. So each time a change was identified, the point of reference for the previous value was processed. At this point, the selected points already look close to our desired result.

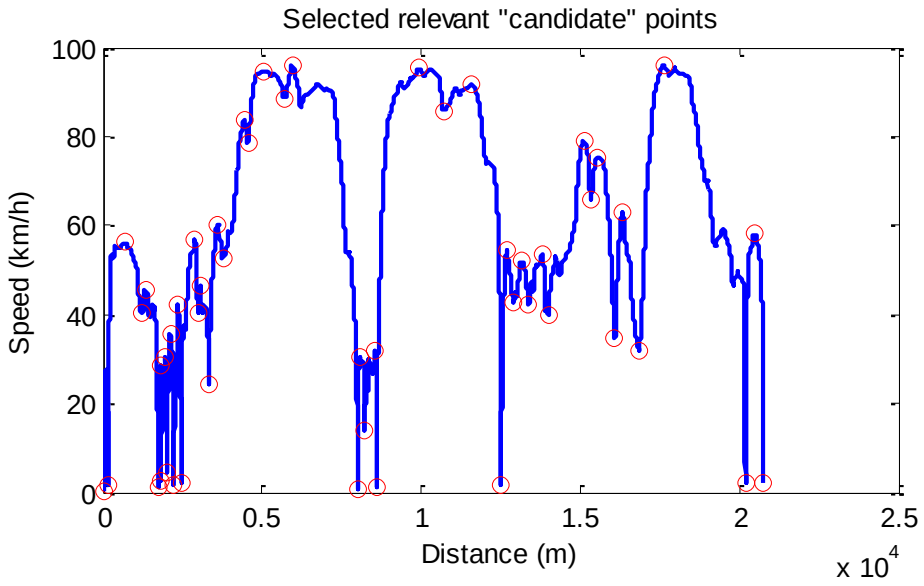


Figure 2.5 - Selecting the relevant points

Identifying the changes of intensity of acceleration

Some driver actions such as the change in the intensity of speeding up and breaking maneuvers should be recognized as different actions, because even if the average derivative of those parts has the same sign they receive quite different values.

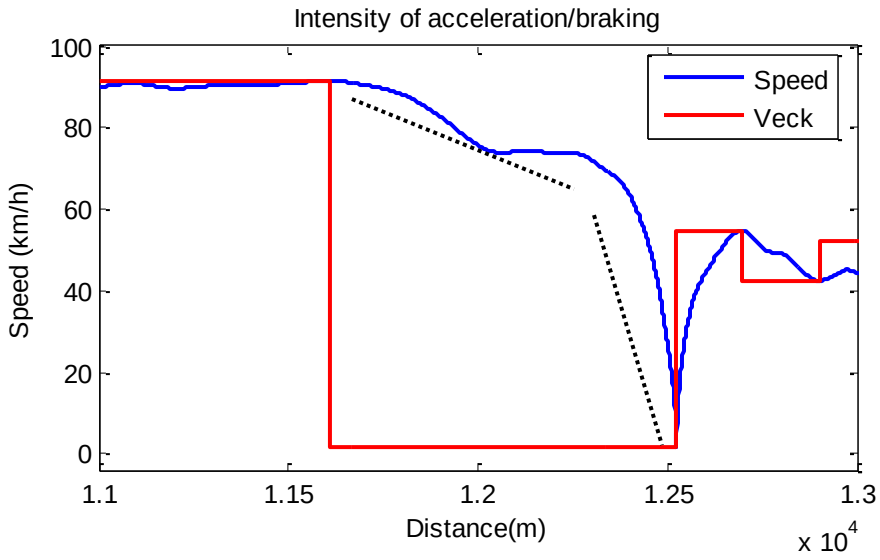


Figure 2.6 - Identifying acceleration and braking maneuvers

To identify these situations that do not always correspond to maximum values the algorithm must go over the speed vector a second time once it has already identified the candidate values that rule the major changes of speed according to the behavior of the driver. Then, a line connecting those points is built, representing the average speed change in that section, so distances of consecutive points to that line can be compared. Maximums and minimums of that subtraction are potentially the desired points, given that the total distance exceeds a threshold value to identify two (or more) different regions.

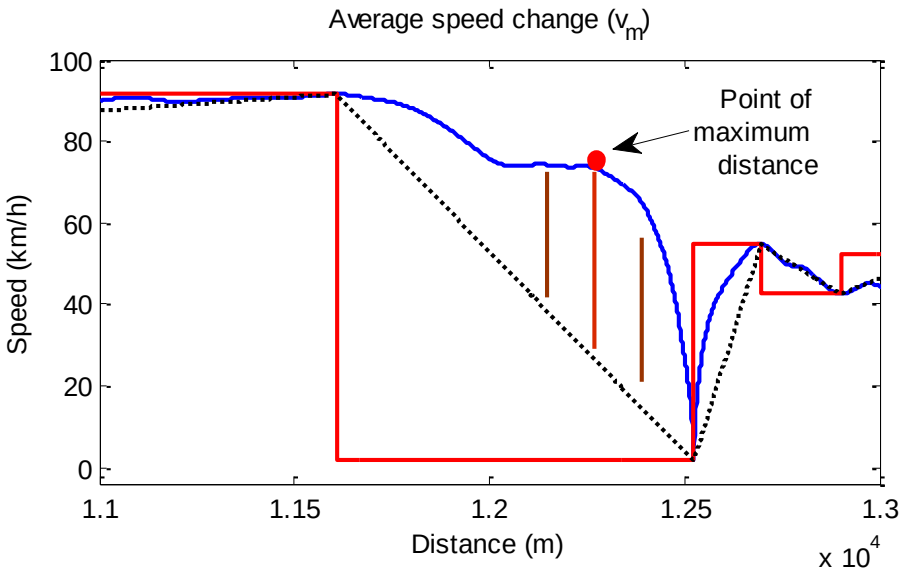


Figure 2.7 - Locating relevant acceleration change points

The following step is scanning the result between the limits of that section by smaller steps trying to identify the values with the maximal and minimal differences to the average line. However this will result in the obtainment of more points than just the right ones.

Filtering

Consequently the total number of points must be filtered. From this point and on, no new points will be added and only subtracted from the selection.

There are three stages of filtering:

Stage 1: the detected points which origin was a gear shift are eliminated

Stage 2: the points obtained at the previous stage which do not represent a sufficiently big change of acceleration are discarded by comparing the variation of the average derivative with the neighbor selected points.

Stage 3: as a last step, delete the points that are too close to each other (below the 5km/h delta) are deleted. The strategy to achieve this is to assign a likelihood value to each point according to the values of the surrounding points and the distances to them. This is the most complex of the three filters.

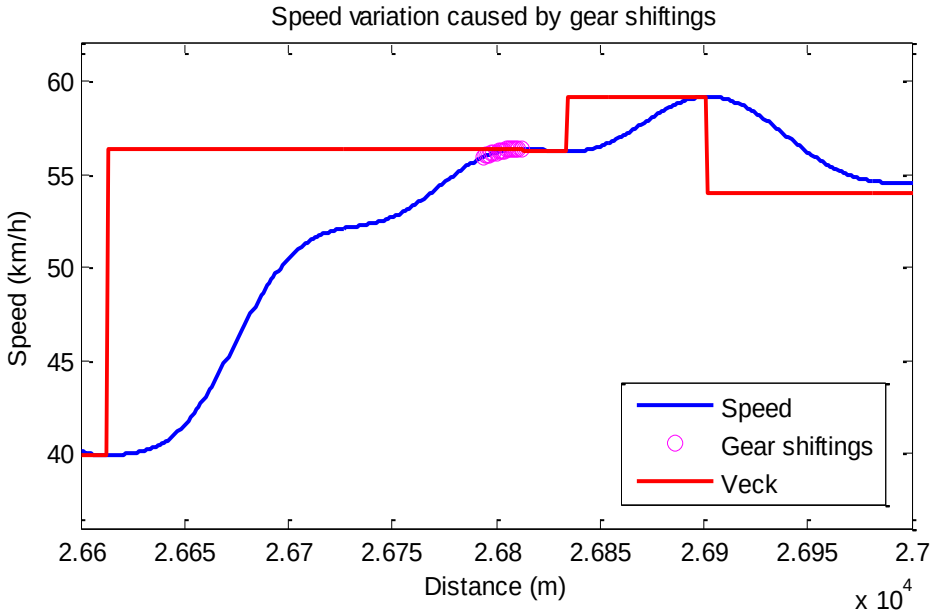


Figure 2.8 – Undesirable point caused by a gear shifting

Overall, I am satisfied with the result obtained here and it seems to be accurate enough for the purposes of this project.

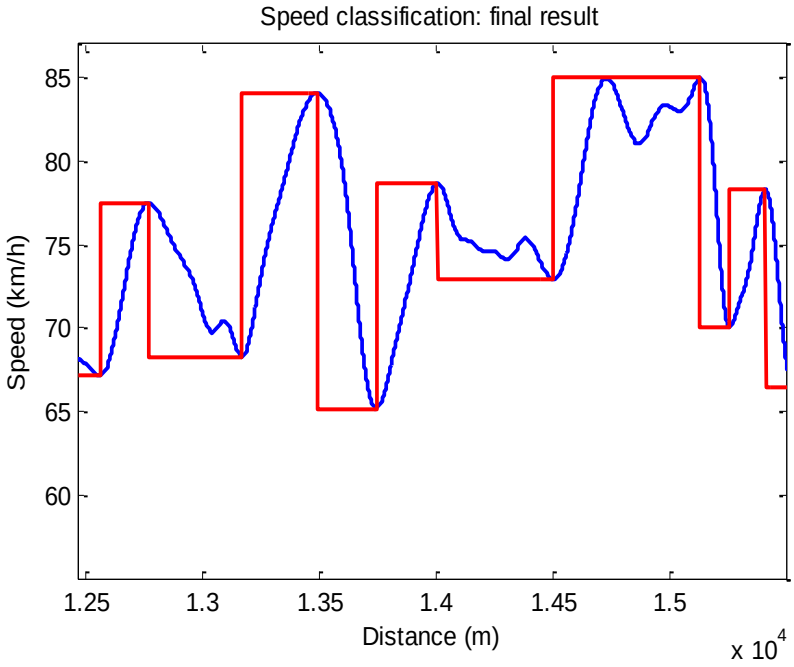


Figure 2.9 - Final result of the speed classification

Discarded strategies (acceleration)

Along the way, other strategies were followed to try to identify the changes of the acceleration intensity. However they turned out to be worse for several reasons. The first and more obvious strategy to be considered is directly identifying in the throttle position and brake pedal position profiles the significant variations.

On the one hand, since the brake pedal is usually only used in braking maneuvers a good correspondence between the significant points of both can be achieved

On the other hand, the throttle position cannot be dealt the same way because even when the driver is trying to drive at a constant speed, he still has to speed up to compensate friction, air resistance… This effect can hide the effect caused by other variables such as slopes so it adds difficult to the problem -*overtaking maneuvers must however be identified as desired speed changes when they cause such changes*-. Besides, the acceleration required also a lot of additional processing and filtering with the new problems they introduce, and more importantly, it depended of the car model.

It is intuitive to think that a car with a powerful engine requires the throttle to be pushed softer to produce the same amount of acceleration and consequently the same change in the speed. For this reason it is not possible to set a universal threshold value that covers any possible car, because it would need to use the relation power/ratio variable, unfortunately not included in the information of the received Matlab files, an additional variable that was not stored in the recorded data and that could not be used. This caused the whole strategy to be discarded even if it could seem more intuitive at the beginning.

2.3.2 SLOPE

The usual order of slope - lateral acceleration - speed has been altered here because the classification of the slope is just a particular case of the classification of the speed, so it was clearer to explain the speed before. The slope is in fact easier to deal with given that it is not influenced by external factors such as gear shifts, wind, traffic… That saves us several filtering steps.

No further explanations on this topic are necessary; everything else described for the speed applies here.

2.3.3 LATERAL ACCELERATION

The lateral acceleration, however, makes use of a completely different approach. The reason for it is that unlike the slope or the speed, there is a "neutral class" corresponding to the no-curve state that is reached periodically. Therefore, any curve can be seen as a process that takes place between two straight sections of the road. This is an advantage that can be used to easily identify the ending of a curve and the beginning of the following one based on the transitions from and to this neutral state.

This algorithm is based on the work of a previous student **[4],** although it had to be revised to improve performance and fix some problems.

The limits are defined by an interval of ± 0.2m/s^2, so the points picked are the last ones that fall inside the neutral region before the profile leaves it. Nevertheless, a problem may appear whenever there is a sudden change of acceleration. Sharp profiles after the interpolation of the distance base (1 m between two consecutive points) could cross the neutral region without having any point inside. These transitions must be detected anyway, so a bit more sophisticated algorithm was implemented to keep track of these transitions too. In those special cases, the points picked are the first ones outside the borderlines of the region, even if they exceed the threshold value of 0.2m/s^2.

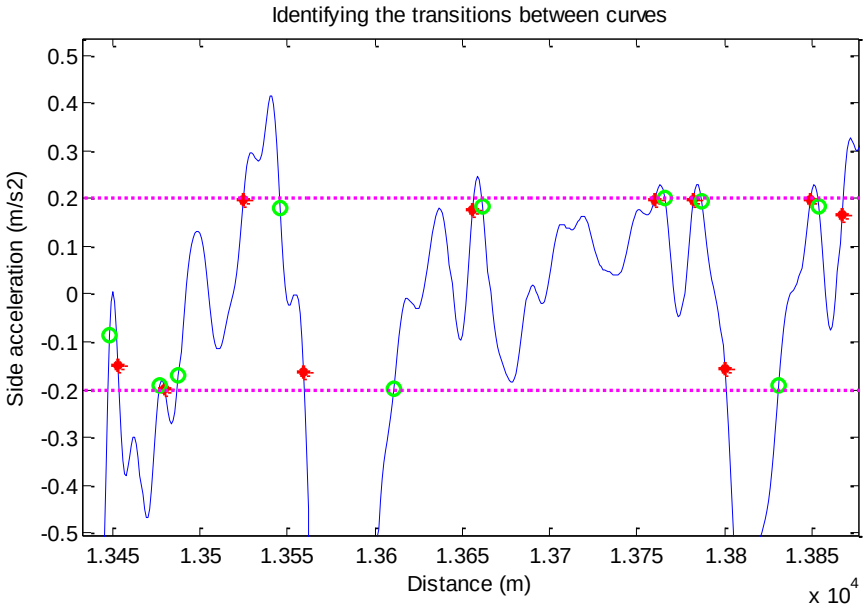


Figure 2.10 - Identification of the transitions between curves

The points and the circles represent the beginning and the ending of curves respectively

Only once all the curves have been identified, only relevant curves are considered for their classification. The rule followed is that a curve is not relevant enough if the average value of that section doesn't exceed an absolute value of twice the size threshold value, 0.4 m/s^2.

After the limits have been satisfactorily identified, the profile within has to be approximated by a trapezium. The slope going up represents the approximation curve that connects more smoothly the straight track with the peak of the curve. In analogy, another line of negative slope can be identified as the output approximation curve that decreases the lateral acceleration from the peak value to zero. The track that lies between holds a lateral acceleration equal or higher than the 90% of the maximum peak, while the class of the curve is defined based on that peak. This 90% represents the 90% of the value of the peak, previously discounted the neutral threshold values to avoid mismatches with low peak curves.

It should be highlighted that only two values are necessary in order to store the information regarding the approximation curves. Once the full length of the curve is known, if the length of the two approximation slopes is stored, the length of the stretch of constant curvature at the pea of the curve can be calculated as the difference between the sum of those values and the whole length of the curve. On a further step, it can be easily reconstructed knowing that the peak stretch is placed at the 90% of the peak value.

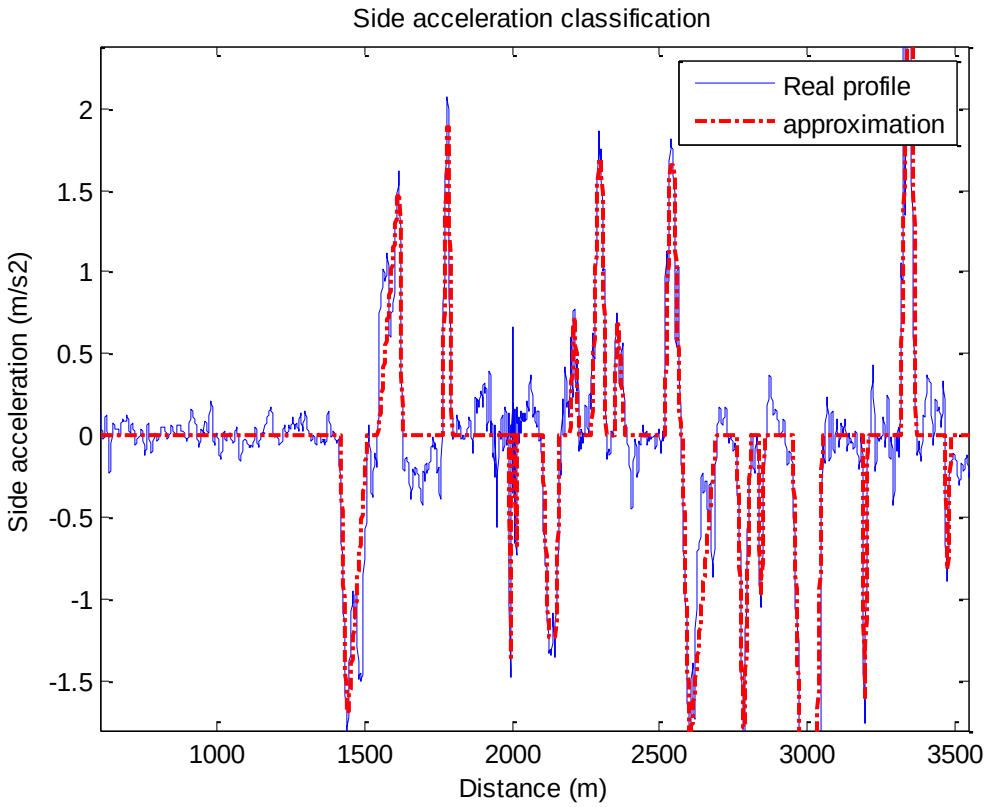


Figure 2.11 - Lateral acceleration classification

Initially the curves had been classified using the data from the curvature and after some months it was replaced by the lateral acceleration. The reason behind this is that the curvature required more classes to give an equivalent accuracy, approximately three times more. More classes imply not only more memory usage, but also the impossibility of using those extra classes or memory to improve other areas or capture the relationship among the other variables.

2.3.4 TYPES OF FILES (COMPARISON)

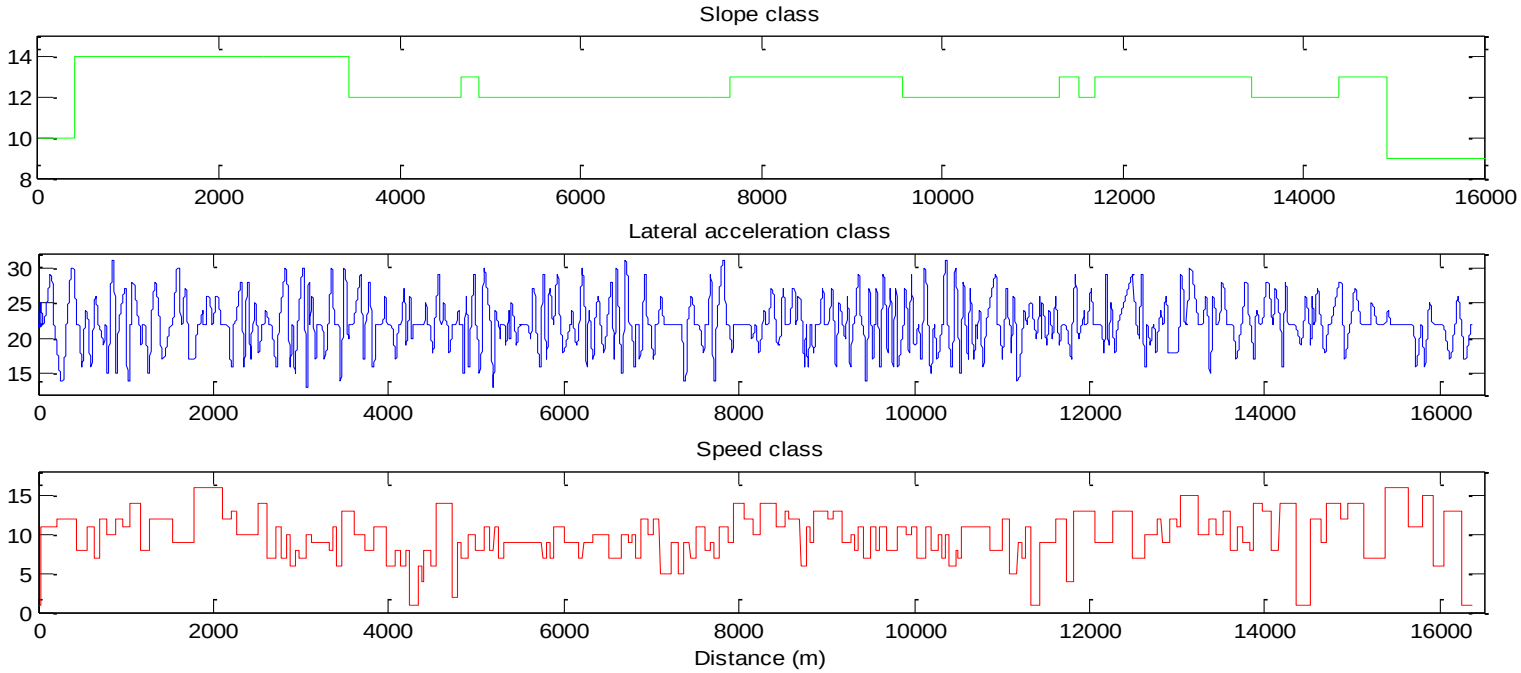


Figure 2.12 - Mountain file

As it is shown above, mountain tracks have more slope variations and these reach higher classes. The speeds achieved are also significantly lower to those reached in the files corresponding to land tracks. Long turns of high curvatures are also more frequent than in other kinds of files.

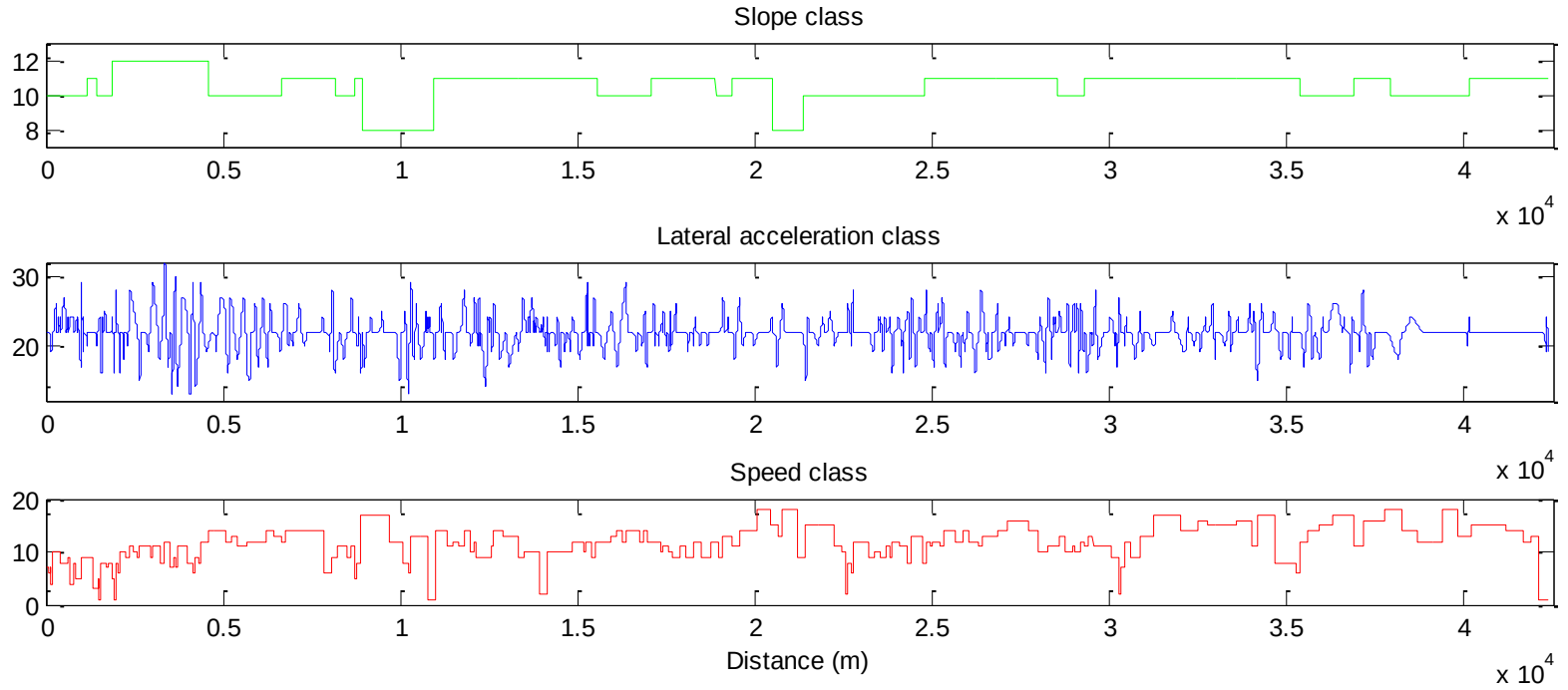


Figure 2.13 - Land file

The land and city roads have less slope changes than the mountain tracks, while the speeds reached are slightly superior

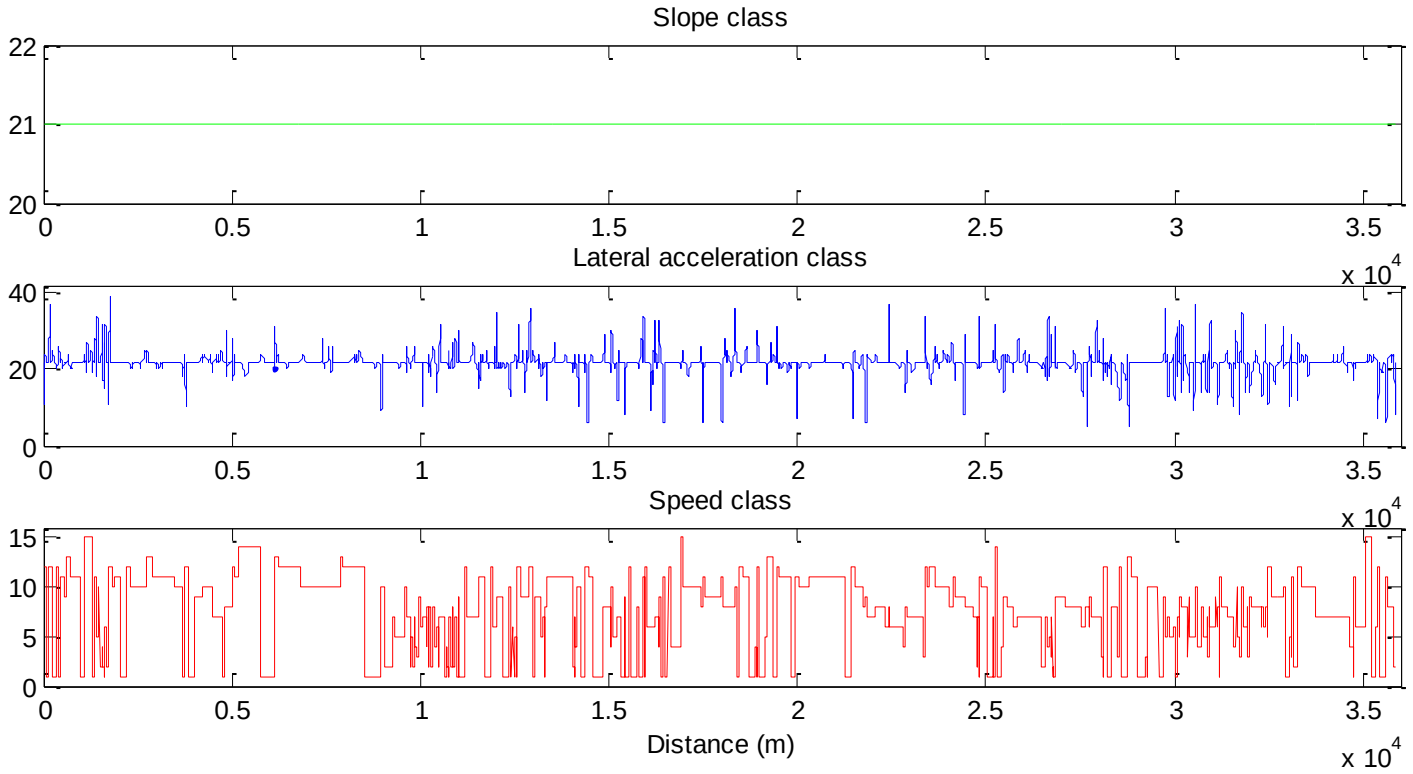


Figure 2.14 - City file

City roads are characterized by sharp but really brief turns and lower speeds with predominating speeds under 50 km/h. Also, there are files like this with no slope class change at all.

Note – I was not provided any Highway track, the fourth kind of road usually considered and therefore they were not analyzed

3 SIMULATION METHODOLOGY AND STATISTICS

3.1 STRUCTURE

In this section, the different statistical tools that will be used to validate the model and compare the results at the end will be exposed, along with the steps of the process that I will follow.

3.1.1 GENERAL CYCLE OF A SIMULATION PROCESS

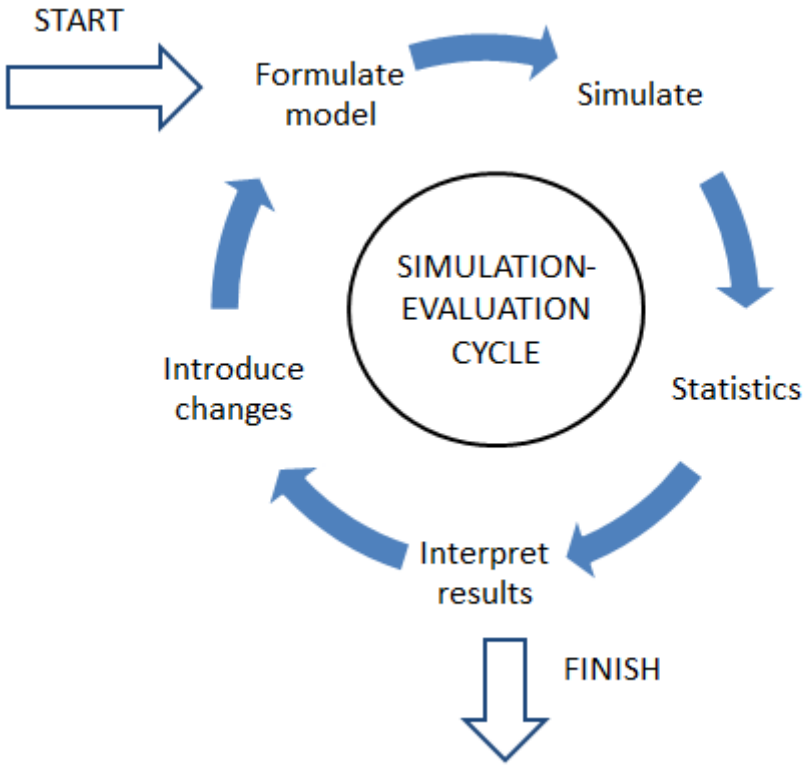


Figure 3.1 Simulation-evaluation cycle

The objective is the generation of 3 vectors containing for each time the appropriate values of slope, lateral acceleration (as a measure of the curves) and speed for a desired track length (from now and on, “3V”, as noted in the glossary).

Steps:

1. Create the tables and initialize the statistics
2. Read the file information and fill the tables with it
3. Generate the track using Matlab or Simulink
4. Compare the statistics
5. If necessary introduce changes and restart the cycle until the statistics match and the simulation resembles the original data

3.1.2 LOGICAL INDEXING

Before being written into any table, the data must be beforehand classified. The objective is double: on one side it allows to reduce a potentially infinite number of values (depending on the decimal precision) to a finite, and relatively small, number of positions that can be fitted to a table; on the other side, it concentrates the data, finding more matches for each input.

After the data is classified, and aiming to speed up the algorithm and improve performance, all the possible combinations of the input 3V need a reserved space in each lookup table, so the table has always a fixed row size with independence of the data it contains. However, the column size cannot be predicted as it depends on the most repeated event in the data which is different for each combination of files loaded. For this reason, I considered the automatic indexing built-in Matlab not satisfying and defined mine with a simple recursive formula:

$$(\text{Speed class} - 1) * n * m + (\text{Lateral acceleration class} - 1) * n + \text{Slope class} \quad \textbf{(3.1)}$$

Where,

n = number of slope classes

m = number of lateral acceleration classes

Following the example the table will be formed from 46053 possible combinations of classes: slope (21 classes), lateral acceleration (43) and speed (51 classes). As an example of this logical indexing, if the input to be classified were: slope (10), lateral acceleration (22) and speed (15); the resulting input position would be:

$$\text{Position in table} = (10-1)*43*51 + (22-1)*51 + 15 = 20.283$$

3.1.3 INITIAL CONDITIONS

The initial conditions for all the files of the data* are set to be: no slope (11), no curve (22) and a stopped car (1), corresponding to the position 20.809 of the table according to the logical indexing exposed above. These will be the initial conditions for our simulation as well.

*To be accurate, there are a few files that do not satisfy this, as shown below, they are however very infrequent. The following example shows it:

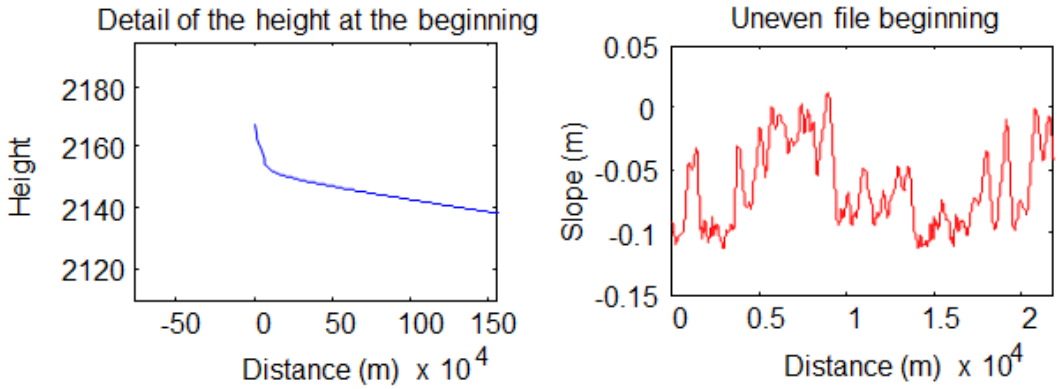


Figure 3.2 - Initial conditions

These profiles belong to the file: ***"VW_Touareg_3LTDI_6AT_L_DU_B_00522.mat"***, and show particular initial conditions for which the slope of the first meters does not fit the neutral class. Above on the right, the height profile confirms that this issue has not been caused by the filtering of the slope, leaving inaccuracies and limitations of the pressure sensor as the most probable reason for the mismatch. It could be that the driver had made a mistake during the recording of the data sets, although it seems unlikely. In any case, the initial conditions that will be used for our simulated track are the ones here described.

3.2 STATISTICAL TOOLS

All the values exposed in the following sections have been estimated from a portion of the data the IAE possesses for confidentiality reasons. I was never provided the rest of the files so it is as accurate as it can be, assuming they are representative of the whole data set.

3.2.1 FREQUENCY AND VALUE STATS

The most basic statistical tool to compare the results of the simulation with the real data is the average frequency of class change for each of the 3 variables and type of track. It is calculating by counting the total number of changes and dividing by the length of the track in meters. Then, the difference of frequencies is divided by the average frequency of the real data, obtaining a percentage error without sign.

Average frequency per variable and type of track in the real data:

Table 3.1 - Frequency statistics

AVERAGE FREQUENCY	Mountain (37 files)	Land (19 files)	City (23 files)
Slope	7.1673 e-04	2.6957 e-04	8.3482 e-05
Lateral acceleration	1.9769 e-02	7.6731 e-03	1.3110 e-02
Speed	6.0517 e-03	3.4707 e-03	8.4772 e-03

A correction must be applied when comparing simulations in some strategies, as the information of the end of the files is not read until the end, instead to the point where the last change takes place. Nevertheless, the whole length of the file is considered for the calculation of the average frequency. This effect is minor at variables that evolve fast (lateral acceleration, speed) but it is noticeable at the slopes.

The frequency is not enough to assure a good correlation and that is why the average class value is also studied and compared:

Table 3.2 - Average class value statistics

AVERAGE CLASS	Mountain (37 files)	Land (19 files)	City (23 files)
Slope	10.0515	10.0410	10.2576
Lateral acceleration	21.3114	21.5165	21.6049
Speed	12.3477	14.1676	8.7975

While the values of the average slope and lateral acceleration are trivial and predictable (nearly the neutral classes), the average speed is interesting.

3.2.2 DISTRIBUTION GRAPHS

There is one additional test that has to be done: a match will only be considered good if it fits in both, average values and distribution. Otherwise certain loops and deficiencies could remain unnoticed.

This color graph displays the frequency with which each change takes place, depicted proportionally to the frequency of the most repeated event. Such event receives a 100% (colored) and an event that never takes place is granted a 0% (white).

A color code has been used in the program to easily distinguish the variables

- Green - slope
- Blue - lateral acceleration (curves)
- Red - speed

This will be used later to visually compare the distribution of the real and the simulated data. If the frequency and average values are close and the distribution graphs look similar, it will not be risky to assure that there is a good correlation between the real and the simulated roads.

I show one distribution graph of each kind as an example:

Slope

In all the graphs, the symmetry will be remarkable, as it is expected. The absence of such symmetry could indicate an insufficient number of kilometers observed or the presence of errors in the code.

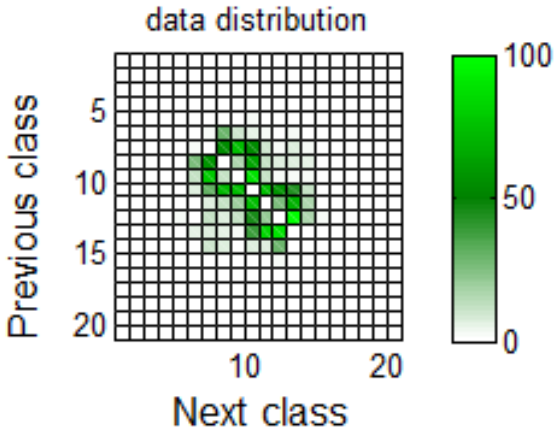


Figure 3.3 - Slope distribution graph

This distribution belongs to a specific mountain track from a road placed in The Alps. It is very uncommon to find roads with slopes that match the lowest and highest classes

Lateral acceleration

Another special consideration must be done regarding the lateral acceleration. It is not as interesting to compare the new class to the class from the immediate previous class as it is to compare it versus the previous non-neutral class.

On the left, the new class is compared to the class of the previous event (either the new or the previous value is the class 22). On the right, only non-neutral classes are involved. This second view offers more relevant information in terms of statistical accuracy.

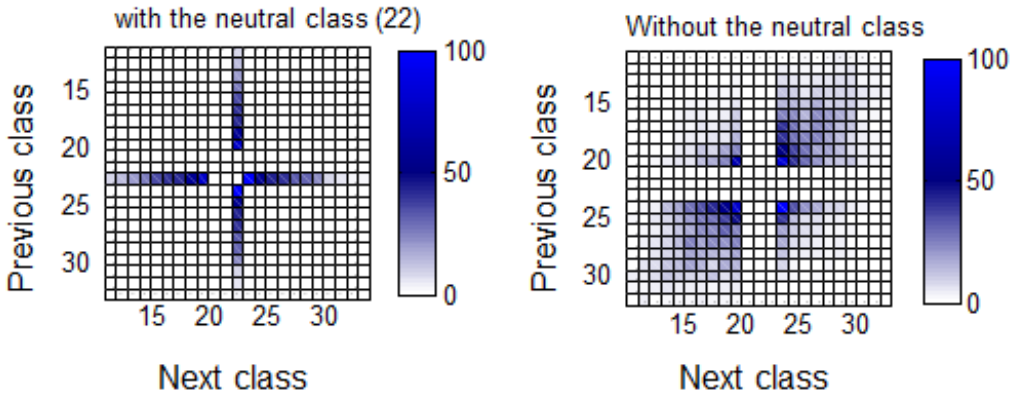


Figure 3.4 - Lateral acceleration distribution graphs

Speed

The speed distribution is beyond a shadow of doubt essentially different for the three different road types. The city tracks show a high number of stops and generally lower speeds, whereas the mountain ones follow a very symmetric pattern with a low dispersion. At last, the land files present the highest average speed and also a much higher dispersion.

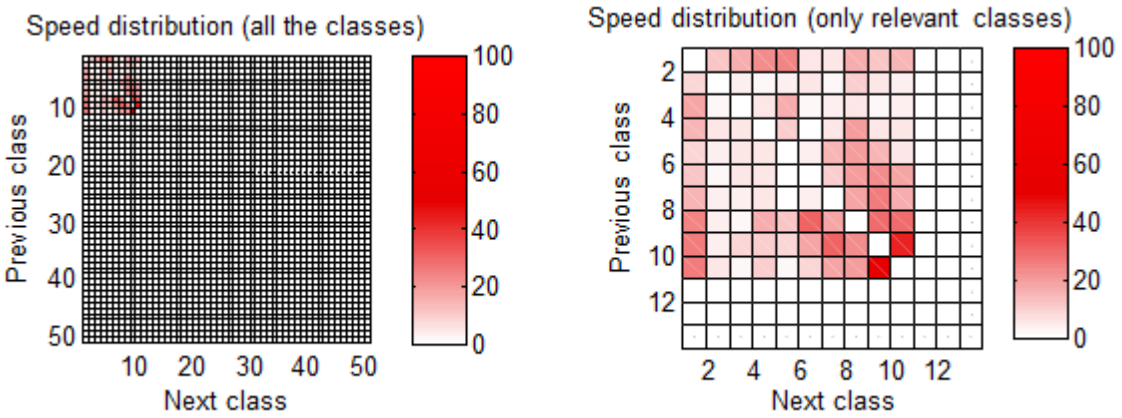


Figure 3.5 - Speed distribution graphs

It is indeed very remarkable that only the lowest classes are filled in all the files used. That distribution corresponds to an urban road, but speeds over 150 km/h (class 30 and above) were not observed. It would be very natural to wonder then if it is necessary to maintain such a high number of classes. It must be understood that the program that is being build must be flexible to adapt to all kinds driven vehicles including sporty cars on highway tacks potentially with no legal speed limitations. I did not have those files among the data I was given and that is why 250km/h was set as the top speed. Therefore the classes remain untouched even if they do not make sense with the current data. Instead I will zoom in the graphs appropriately.

Regarding the speed, when all the tracks of all kinds were combined, the frequent stops of the city files dominate, and the resulting graph has a greater dispersion than any of the file types involved.

3.3 ROAD REPRESENTATION

Before being able to represent the road, whenever the program works with classified simulated values, it is first required to declassify the information. After that, the space coordinates x and y of the points of the road are calculated applying basic trigonometry as it follows:

$$\theta(i) = \delta s * \text{curvature(i)}) \quad \textbf{(3.2)}$$

$$Y(i) = \delta s * \cos(\text{cumsum}(\theta(i))) \quad \textbf{(3.3)}$$

$$X(i) = \delta s * \sin(\text{cumsum}(-\theta(i))) \quad \textbf{(3.4)}$$

Note – the curvature and slope here are not oriented profile values, they are real values at each time. The curvature can be estimated from the lateral acceleration.

The remaining variable (speed) will be used to define the color of the line, but this one can be displayed in both classified and declassified versions. The following examples show the classified version, in other words, the speed that the driver is willing to reach at any point and not the real speed of the class.

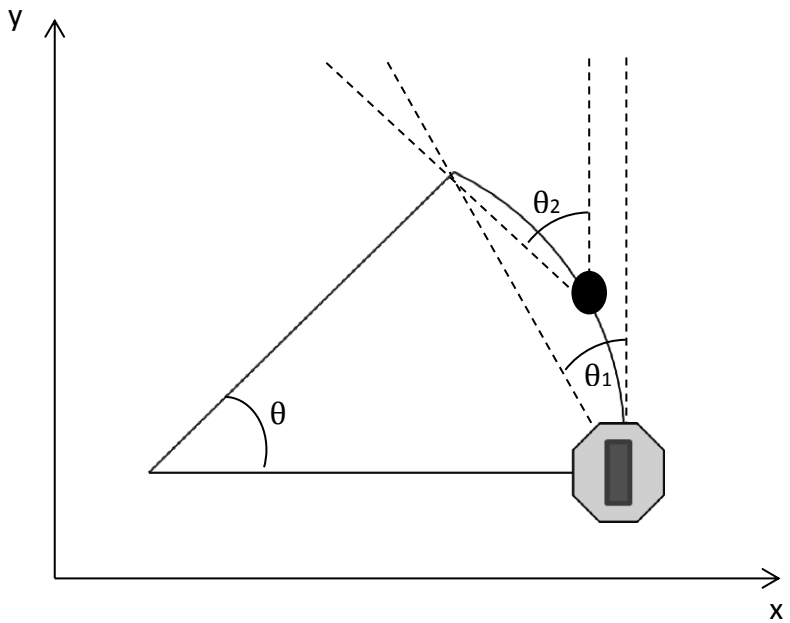


Figure 3.6 - Trigonometry at the declassification

3.3.1 2D ROAD GRAPHS

The 2D graphs are displayed using the Matlab function scatter that allows us to use a vector depending color unlike the basic plot function.

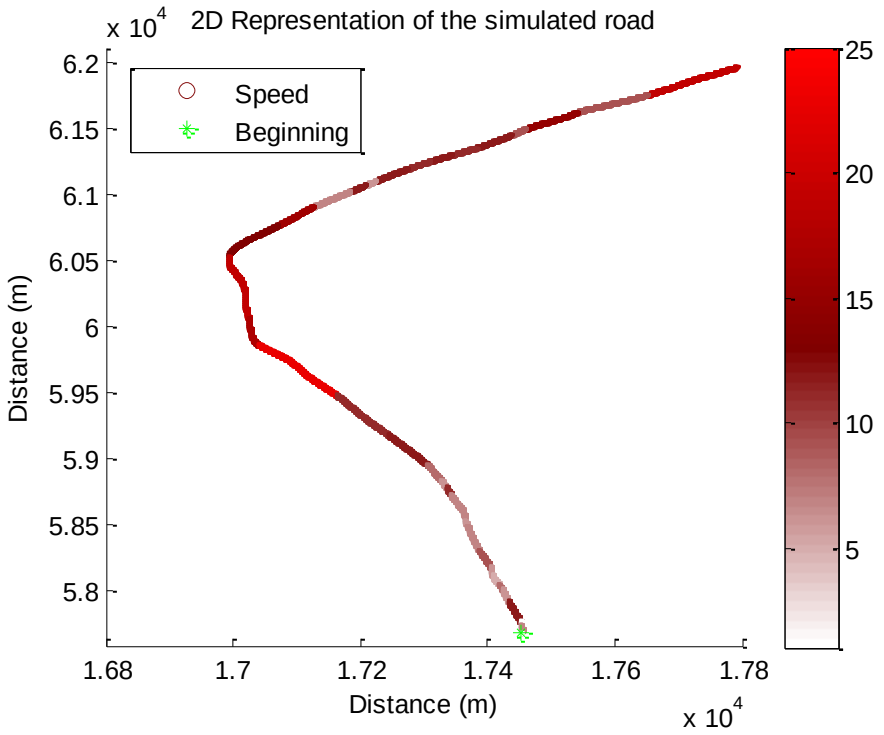


Figure 3.7 - 2D Representation of a section of a road

This is an appropriated tool for roads for which the influence of the slope is almost even and not relevant, allowing us to simplify the model into something easier to plot and visualize.

3.3.2 3D ROAD GRAPHS

The height is calculated incrementally from the slope

$$h(i) = cumsum(\delta s * \text{slope(i)}) \qquad \textbf{(3.5)}$$

Unfortunately, there is no direct Matlab function (to my knowledge) to create a 3D plot with the color depending of a vector. To be able to represent the height in addition to the previous information, a Matlab script was specifically written and the results are shown below:

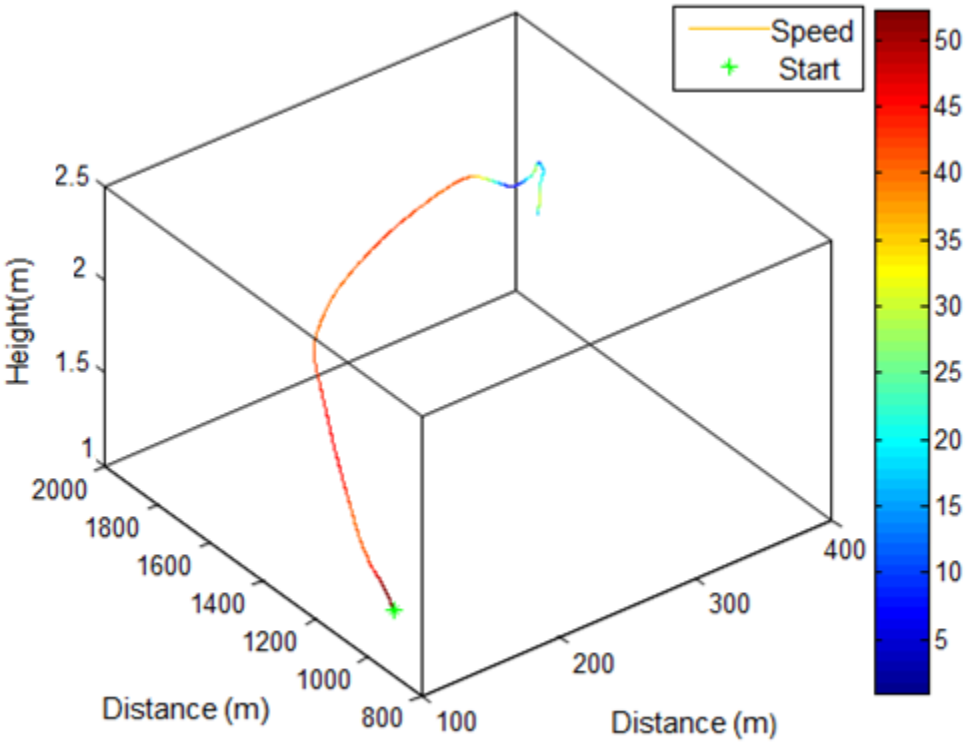


Figure 3.8 - 3D representation of a section of a road

The number of points that are generated are excessive for most computers to manage in 3D so only a small part of the road can be plotted at the same time. This representation is useful to detect bugs and incoherent behaviors.

4 SIMULATION STRATEGIES

4.1 SIMULATION STRATEGY Nº1: ALL IN ONE

The strategy that is described here was the first suggested approach and continues the attempts of the previous student. Although it was clear that it was not going to work, it was important to understand the problems in their full scope and be able to propose alternatives.

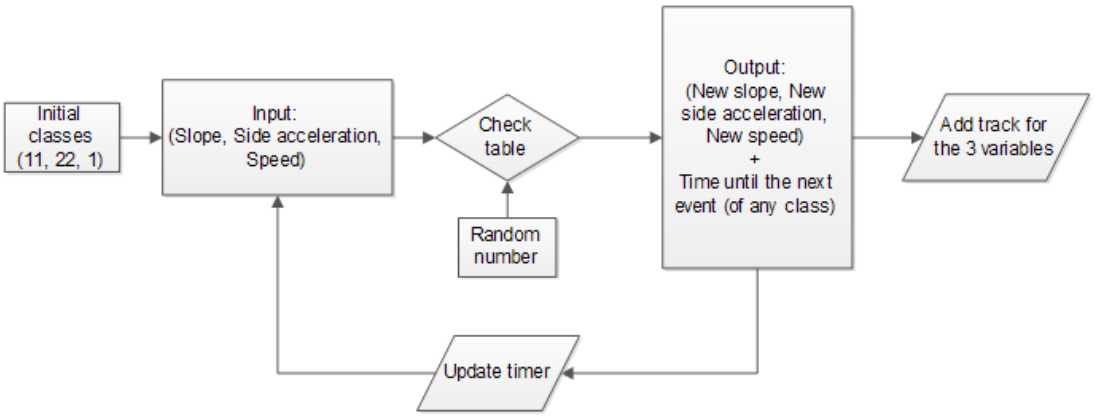


Figure 4.1 - Strategy 1 overview

As shown on the graph above, the statistic simulation model faces several problems that will be described in the following subsections.

4.1.1 LOSS OF ANY CONTROL OF THE TRACK LENGTH

The main problem with this strategy is that it goes along the real data storing for each change in any of the 3V, the real value of the 3V and the distance to the next change of any of them. That is not the change of the same variable again, the change of any of the 3 variables. It is plain to see with the help of the graph shown below. The classes and time steps are fictional and the 3V have been reduced to 2 to make the example easier to understand, corresponding to straight track without curves.

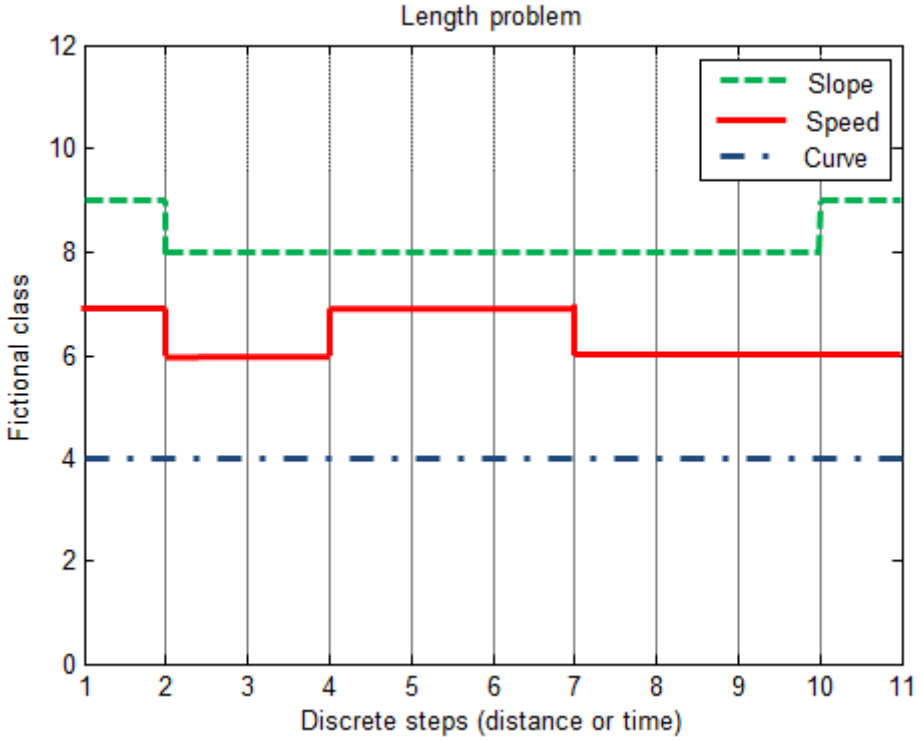


Figure 4.2 - Length extreme shortening issue

Assumingthat the speed and the lateral acceleration are variables that due to their nature change more frequently than the slope (most of the roads are in their most part even), that feature is expected to appear in our simulated values. Therefore, it would not be acceptable to have a simulated slope that changed several times in a small period of time because nothing similar can be found in the data.

However, looking at the example and according to this strategy, our simulation could be at step 2, add the slope change from class 9 to 8 and expect the next event at step 4. Once at step 4, the tables are checked again but instead of picking the change observed at step 4 (speed), the one observed at step 10 (slope) could be equally picked at random from the tables, giving as a result a new slope change only two simulation steps after the previous one, which is unacceptable given the physical interpretation of the values.

4.1.2 CURVES DEPENDENCY ON THE PREVIOUS CURVE

That was not, of course, the only problem. In the way the lateral acceleration has been classified, two consecutive curves are always separated by a straight track (neutral class). This part can be short or long, but one thing is for sure: by this strategy, each time that a new curve is resampled, the current lateral acceleration class, which always will be the neutral class, is used as a reference. It implies automatically losing any information of the curves before the curve that is being added. It is plain to infer that if there is a sharp left turn in a mountain track, the chances for the next curve to be a sharp right turn increase. So without having developed statistical checks to detect this, the simulation would be wrong as many dependencies would be missed.

On a side note, the speed reached at the middle of a curve is not likely to be the same for a long than a short curve but, again, any kind of connection among the variables beyond their current value is ignored as no kind of length control is implemented.

4.1.3 THE PROBLEMS DERIVED FROM THE END OF THE FILES

There is also a very important issue that must be addressed in order to continue with the simulation, and it is related with the following question: what can be done when the simulation reaches a situation at the end of the file that has never been recorded before and there is no valid new class to pick? In other words, it must be decided what to do whenever the algorithm looks for 3-coordinates (a set of classes of the 3V) that do not match any input in our statistical table.

Something must be done or the simulation would eventually get stuck. The solution adopted here was post-processing the table after it had been generated in search of those situations or “dead ends”, deleting all the outputs that aim at empty positions in it before the table can be used. In order to produce good results only a small part of the table should be deleted by this technique. Nevertheless, when the deleted information is the only matching data for a given input, the whole entry or position at our table would get deleted, requiring performing a second search for the outputs aiming to that now empty entry. Taking care of the number of deleted entries and the amount of information lost is consequently important.

So as the process go on, several steps could be necessary and in the worst case scenario the whole table would be deleted, meaning that the data has never closed a loop. In other words, this issue appears when trying to create a long table with two or more variables as inputs and the program has not found the same 3-coordinates twice in the data.

The natural advantage of post-processing the table is that the simulation is never going to reach a situation for which the algorithm does not find information to go on. This technique has been exposed to clarify the problems that were later encountered after having changed the simulation strategy when this technique can no longer be applied. However it will be used again with the fourth and last strategy.

The amount of available data (total number of recorded kilometers and events) is the main limitation all along the project, especially considering in the final simulations only files of one kind of track (mountain, land or city) will be used at a time. Further separation of the data by other parameters of the 3D model such as the behavior of the driver would reduce even more our available set of data.

4.1.4 TOWARDS THE NEXT MODEL

It has been made clear that there is a need to implement some kind of length control in order not to implicitly break the statistics and to prevent physically incoherent variable changes. The curve dependency problem suggests that the particularities of each of the 3V require each of them to be treated differently (the lateral acceleration needs also information from the previous curve, the speed is influenced by what the driver is seeing coming ahead, the slope registers less occurrences…)

4.2 SIMULATION STRATEGY Nº2: MODULAR

For the reasons exposed above, it was decided to split the table in three different new tables, one for each of the 3V, to always make use of the most information available in each step. It must be noticed that all the work is strongly limited by the relatively low amount of data available. The more specific the identified situations are, the less frequent those actions occur and the emptier our tables will become.

At this point, the chosen approach was a logical one paying attention of how the roads are indeed built. Before the track, there is only the terrain, which will be usually flattened and modified where needed. Then, and always depending on the characteristics of the terrain, the local corresponding laws and economic reasons, the path of the road -and therefore its curves- is outlined. Once the road has been built, the driver drives through it at a speed that depends of the other 2 variables exposed above (slope, curves) and many other factors such as traffic conditions, power of the driven car and the behavior of the driver itself.

So it seems reasonable to consider that the curvature of a road is influenced by the slope of the terrain (the hillier a terrain is, the more curves the new road is likely to have); the speed is influenced among other factors by both slope and curves, and the slope is not influenced by any external factor. This is why slope-lateral acceleration-speed has been selected as the optimal order of simulation when the 3V are not generated simultaneously.

4.2.1 ALTERNATIVES TO POST-PROCESSING

It has been proved that it is absolutely vital to keep control of the added track lengths. I used 3 counters to store the values of the last positions when information were added (pos1, pos2, pos3) and then I sample new event information for the variable which counter holds the minimum value. Draws are broken randomly.

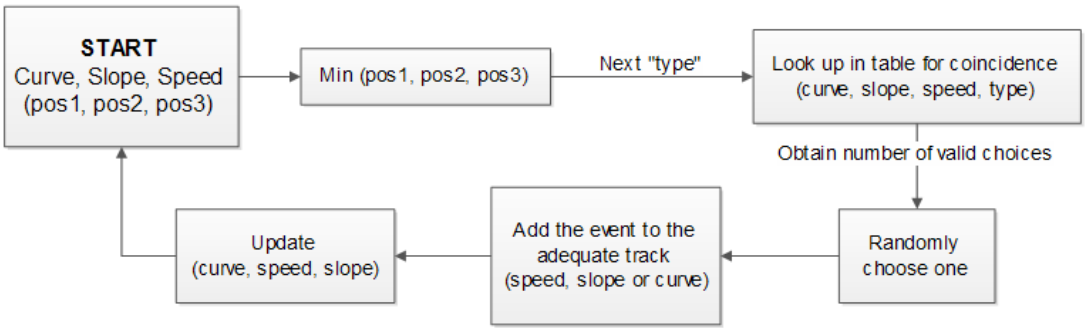


Figure 4.3 - Strategy of approximation to avoid post-processing

Note – Whenever "curve" is written, it refers naturally to the lateral acceleration and not to the curvature. The reason for this is that the criterion was modified by the supervisor during the development of the project.

Unfortunately, having 3 different "simulation times" prevent us from post-processing the table due to the fact that the kind of the following event depends always on the smallest counter and that of the length of the tracks that are being randomly added. Checking all the possibilities would lead into deleting always the whole table because if only one single combination of lengths made one output void, it would have to be deleted for the simulation or the simulation could get stuck. This constitutes another huge problem because it is no longer possible to be always sure that the algorithm is always going to find stored data that matches the current situation of our variables.

At this point, my suggested approach was to be tolerant with errors and try to randomly pick similar information to the missing information, but only among the possibilities available that are the closest to the needed ones. So, instead of picking one random input, a transfer function must be created to classify how "desirable" an approximation of the input values is given our input data.

I would like to show an easy example to explain this new approach better:

- For the example, let's imagine this is our full table:

Input			Output 1		Output 2	
S	C_c	P_c	class	length	class	length
13	22	30	22	105	22	45
12	22	30	22	65	22	21
7	22	35	22	13	-	-
7	30	22	14	45	28	67

Table 4.1- Approximation method, initial table

Where,

S=slope

C_c=current curve

P_c= Previous curve

Now the different possible scenarios are explored:

Case 1: no approximation needed

- As an example, if this were desired input in the table (S=13,C_c=22, P_c=30)

Table 4.2 - Approximation method, initial table

Input			Output 1		Output 2	
S	C_c	P_c	class	length	class	length
13	***22***	***30***	22	105	***22***	***45***
12	22	30	22	65	22	21
7	22	35	22	13	-	-
7	30	22	14	45	28	67

The input is directly picked and one of the possible outputs is randomly selected for example the first one. Nothing has changed compared to the previous strategy.

Case 2: required case missing

- Desired input: (14, 22, 30)

It cannot be found in the table.

Table 4.3 - Approximation method, case 2, step 1

Input			Output 1		Output 2	
S	C_c	P_c	class	length	class	length
13	22	30	22	105	22	45
12	22	30	22	65	22	21
7	22	35	22	13	-	-

7	30	22	14	45	28	67

As a first step, the error weights must be defined. For this example, the considered error coefficients are (1, 2, infinite), they will be explained below.

$$\text{Total error} = 1 * \text{error\_S} + 2 * \text{error\_P\_c} + \text{inf} * \text{error\_C\_c} \quad \textbf{(4.1)}$$

The coefficients depend directly on the importance of the committed mistake, having accepted a certain approximation

Sorts of mistakes

1. Inadmissible mistakes → to this category belong those which would prevent the simulation from going on, such as having to sample a slope change and sampling a speed change instead. Also, for those which outputs match the input, namely those which would not produce any change of our 3-coordinates at all, the algorithm would resample again without replacement. Those approximations are given an infinite coefficient so they can never be picked.

2. Undesirable mistakes → to this category belong those which would better be avoided. As an example, when sampling a new curve, the information of the previous one is more important for us than the information relating slopes. Those are given higher coefficients, but allowed if there is need to make them.

3. Tolerable mistakes → to this category belong those which would not modify substantially the given conditions. The goal is to always choose these so they are given the lowest coefficients.

In short, first the initial values (input) are defined, and then all the possibilities that contain only undesirable or tolerable mistakes sorted. Finally, each possibility is assigned a global coefficient (GC) according to their suitability. Once all the possibilities are ranked, one is randomly chosen from all of them that share the lowest global coefficient.

$$GC = C1(\text{binary}) * \text{inf} + C2 * \text{distance in classes} + C3 * \text{distance in classes} \quad \textbf{(4.2)}$$

Most of the times, this strategy results in modifying one of the least important variables by only one class. To clarify it a little bit more, here is an example of what would happen when sampling a speed change. As it has been previously defined, that would make compulsory to choose an input that corresponds to a speed change, which is preferred to be accurate in speed, while allowing slight deviations in the slope and lateral acceleration classes. In the followings is described how these coefficients are calculated:

Continuing with the example: our simulation is still at the case 2: the required input is missing

- Step 2: calculate the error and sort the table

The algorithm picks the first ranked approximation and draws are solved randomly. The error committed at each iteration is stored for the statistics.

- Desired input: (14, 22, 30), error= **1** * **error_S** + **2** * **error_P_c** + **inf** * **error_C_c**

Table 4.4 - Approximation method, case 2 step 2

Input			Errors				Position
S	C_c	P_c	S	C_c	P_c	total error	class
13	***22***	***30***	***1***	***0***	***0***	***1***	***1***
12	22	30	2	0	0	2	2
7	22	35	7	0	5*2	17	3
7	30	22	7	8*inf	8*2	inf	4

After the previous steps, the value of the GC must be recorded to understand how accurate the accepted approximation is. Obviously another question arises: would not this break the statistics as well? It is true that tolerances have been introduced but theoretically, as long as the choice among the possibilities is correctly randomized and generate a long enough simulated track, all the mistakes should be counteracted and have no influence at all. However, the algorithm is forced to forsake some accuracy to be able to keep simulating regardless the data inside of the tables. This is its greatest weakness, it will no longer be as accurate as it was before, but the more data there is available, the better the result will be. This technique scales well for any data with narrower confidence levels if fed with more data so it has to approximate less often.

- Dealing with loops

Another question that arises is if it would be possible that the simulation entered a loop as a consequence of having a table which approximations pointed to themselves. The question has not an easy answer, since those loops could take place also without any approximation in extremely infrequent dispositions of the original data table. In other words it would not be caused exclusively by this approach as long a correct randomization of the draws is guaranteed. There are alternatives in the way the options are sorted that are better in terms of avoiding loops although they all produce a higher approximation error than the current one. The way exposed above has as its main goal to minimize such error.

The main of those alternatives would be to calculate the cumulative error of each possible new 3-coordinates and resampling inversely proportional to that value. The values with high error coefficients would be rarely resampled and the ones with infinite coefficients not even once. Comparatively, it would increase the approximation error, although it would offer a better response to the loops.

However, there is one circumstance that helps us to break any loop in case it would appear. Since the 3V, are generated in different steps, if the simulation fell into a loop in the lateral acceleration, for example, it would be eventually broken as soon as it reached a change in the simulated slope. Therefore, the simulation could only enter theoretically an infinite loop for the slope, but as the slope does not depend of any other variable, no approximation is needed to simulate it. The process is hence guaranteed.

- Advantages

The only alternative would be to this technique under this strategy would be to reduce dramatically the number of variables and classes so the table is filled for any input, and even so, a single empty space in it would prevent us from continuing the simulation. On the way exposed above, exact 3-coordinates are picked when possible, and when not, the most important of them are held and the most similar values are searched for the rest. Therefore, this strategy will always use the same or more information than just reducing the number of variables, even at the worst case scenario. The problems are caused by the lack of enough data itself to use as many variables as needed.

4.2.2 ADVANTAGES OF MODULAR STRATEGIES

From now and on, instead of trying to build the whole simulation at once, the simulation is going to be constructed by modules. Initially, it is composed by the three obvious blocks (slope, lateral acceleration and speed) that are generated in Matlab in three separated files and will be implemented in Simulink only if they work.

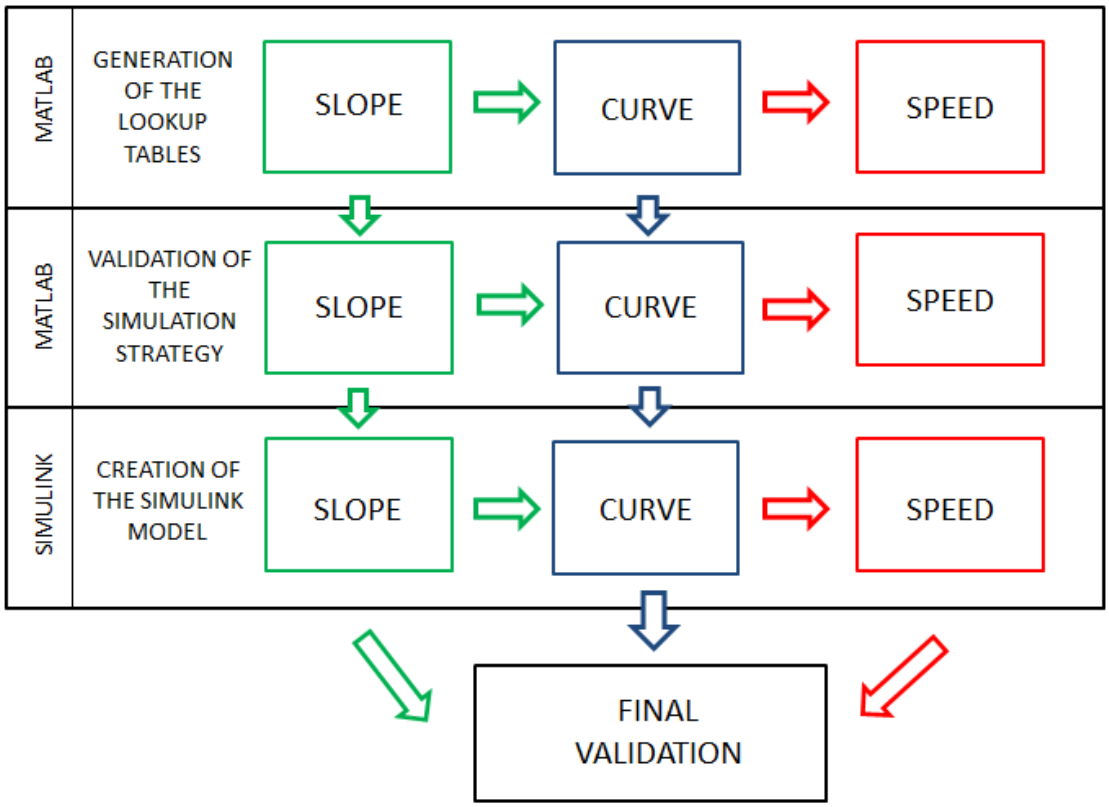


Figure 4.4 - Strategy 2: global program flowchart

The objective behind this new focus was not only to be able to obtain the maximum of available data, but also to make the debugging and testing processes easier by dividing the problem in smaller programs. Due to its nature, one of the stages can be modified or replaced without affecting the rest of them and it is also easier to trace the flows of information between the different phases of the simulation.

On the other side, it will of course increase the work required to model it in Simulink, reason why the work with Simulink only makes sense once the Matlab work is finished and verified. This does not affect the problems derived of not being able to post-process the tables; it is the existence of three different simulation times itself what causes it.

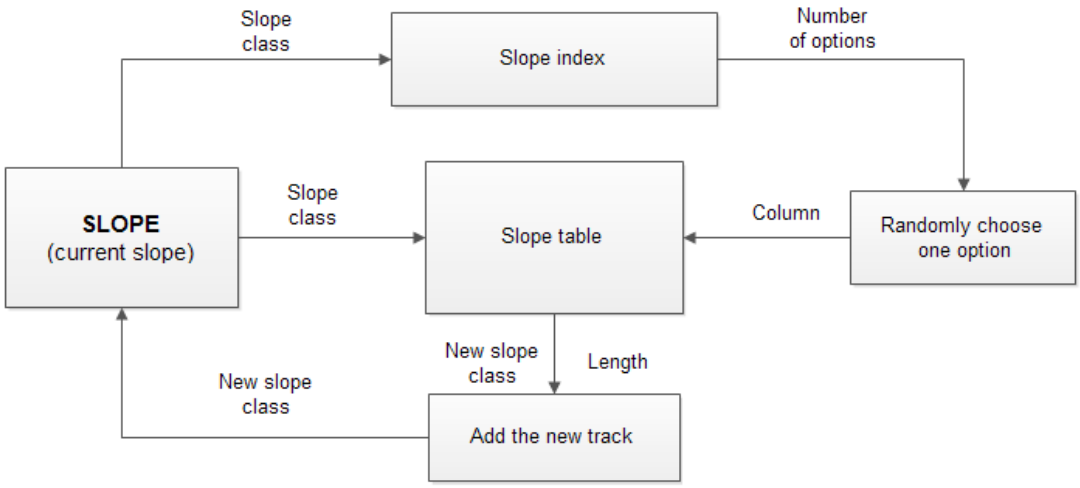


Figure 4.5 - Strategy 2 overview

The model above must be repeated 3 times, one for each of the 3V.

4.2.3 SLOPE

The slope is the easiest variable to model as it does not depend on any other variable and, consequently, the required table is as small as 22x22. It is convenient to remember that there are only 21 different classes of slopes in our classified slope, the last one is fictional, unused at the final strategy, and its purpose is explained below.

This is a real data slope graph:

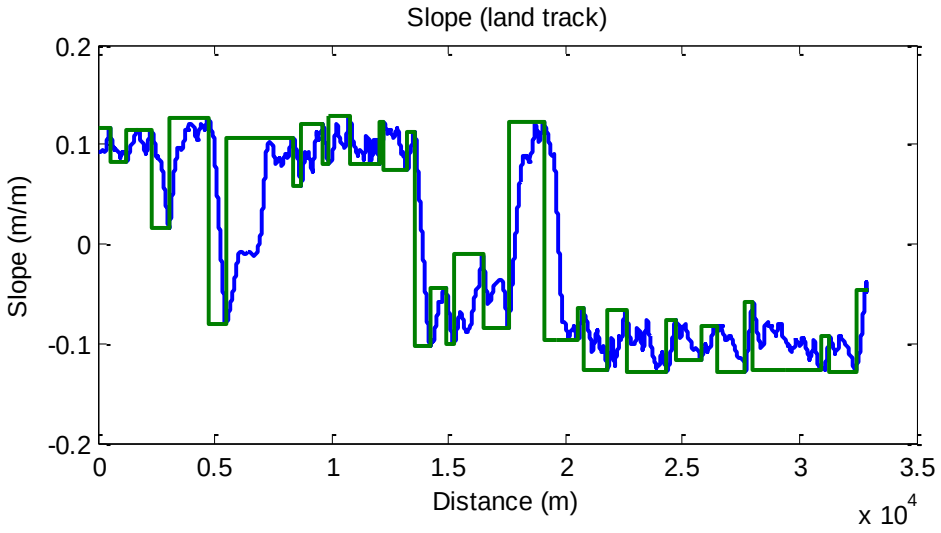


Figure 4.6 - Strategy 2: Slope of a an even road

And this is how it looks after the classification:

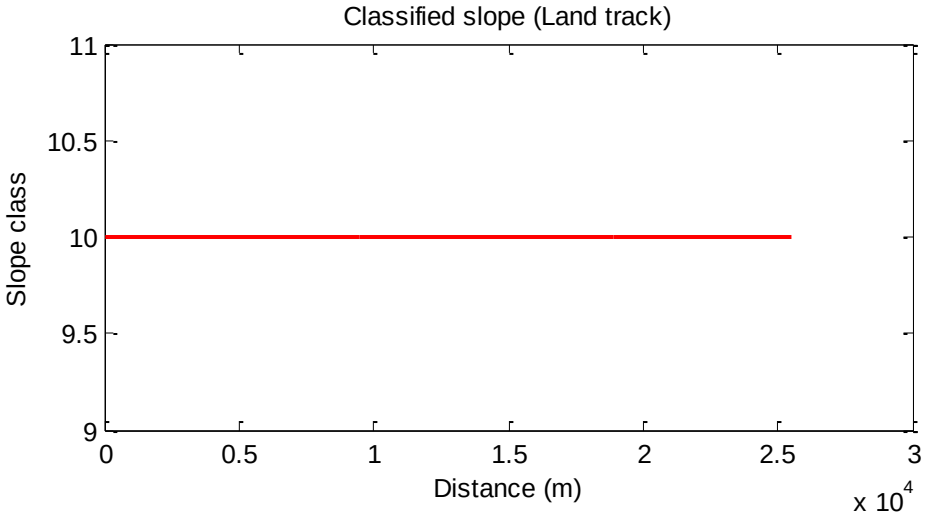


Figure 4.7 - Strategy 2: Classified slope of a an even road

The reason is that in the whole length of the track the algorithm has never seen any slope below -0.03 or above 0.03 and because of it; to the effects of the slope tables, it is equivalent to an empty file. It may not be obtained useful change information but it affects the statistics anyway. This file represents that the test driver encountered an even road at least 25.532 meters long and if it were just skipped, it would be distorting the real frequency statistics by systematically eliminating the possibility of having even tracks longer than the distance for which the sensors were on and measuring. And in addition to that, even if there were a change at all, there would still be no way to use the information placed from that point to the end of the file. An example below shows it, again with fictional classes and time steps:

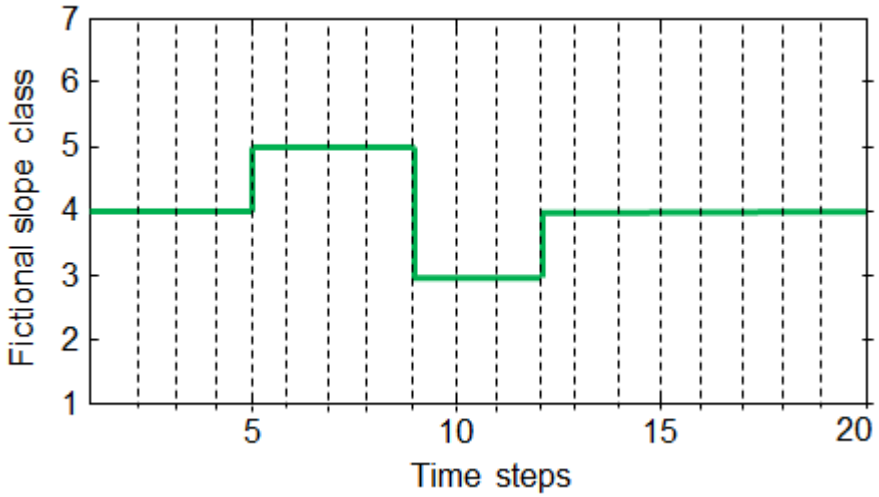


Figure 4.8 - Strategy 2: Slope of a an even road

Considering the example above, a slope with three changes, one at time step 5 from class 4 to 5, another at time step 9 from class 5 to class 3, and the last one at time step 13 from class 3 to 4. There is a total of 12 time steps with a change frequency of 100*3/20=15 changes per 100 time steps. However, none of them takes place after time step 12. As a consequence, having only added data of this file to fill the tables, as soon as our simulation arrived to step 12 with a slope class to 4, the algorithm would resample and add again the change at the beginning of the file with a duration of only 5 time steps; because desired length of the simulated track has not yet been reached. At the end, the simulated track would have more changes and therefore a higher frequency. Such frequency would tend to 100*3/12=25 changes per 100 time steps, an error of nearly 67%.

The objective of this fictional class is to create an unreal change at the end of each file that connects the ending of a file with the beginning of another. This is the so named fictional class 22. Every time the simulation reaches this class, the slope must be resampled once more without adding anything, reaching information obtained from the beginning of other (or the same) file. It should be highlighted that by allowing to continue the end of a file with the beginning of another that could start with the same class, not every time that the fictional class is visited there will be an actual variation of the slope in our reconstructed slope. However this is not undesirable, it is indeed made on purpose so even tracks longer than the recorded data can be generated at a low probability, overcoming one of the limitations of the way the data is measured.

Nevertheless, there is unfortunately a downside that comes along, at the transitions between files changes from one class to another that were never in the original data are observed. This error will be equally randomized and therefore dispersed along the simulation, unlike the previous problem that was systematical.

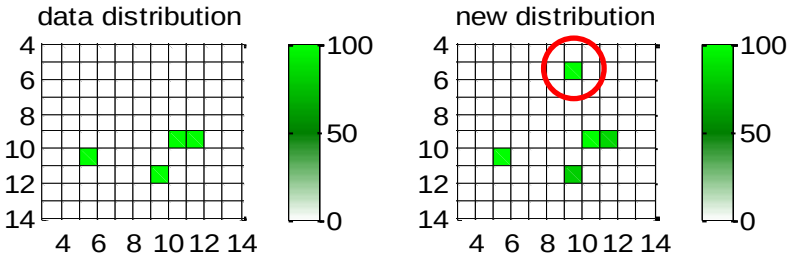


Figure 4.9 - Strategy 2: Extra transition at the joint between files

At the example above, obtained on purpose from one single file, the existence of the never observed transition is clear. This transition takes place at the joint of the ending of that file with its own beginning, to be able to run simulations longer than the original file. This error is minimized and almost eliminated when the tables are filled with a huge number of files.

As a conclusion on this topic, under the limitations of this strategy that issue cannot be avoided. The difference is that in order to be comparable, the number of occurrences in the frequency statistics matrix must be compensated by a factor equal to the number of files -1 in the original statistics.

4.2.4 LATERAL ACCELERATION

These are the variables I considered the most influential ones for the lateral acceleration:

- Current lateral acceleration class (43 classes)
- Previous lateral acceleration class (43 classes)
- Current slope class (21 classes)

Resulting in a combination of 43 x 43 x 21 = 38.829 different entries in the table

As seen before, having a fictional class to deal with the endings of the files, along with the absence of dependence of other variables, allowed the slope not to require any approximation when retrieving information from the lookup table. This however does not happen with the lateral acceleration, the simulation could arrive to a step at which it has already met the slope and current lateral acceleration classes before, but being the first time it meets that combination of both classes together.

These are the suggested weights for the approximations:

- Current curve: infinite
- Previous curve: 2
- Current slope: 1

They represent that if the class of the current curve was wrong, it could be allowing a curve to be followed by another curve without a no-curve state in between. Alternatively, it could be adding the same class from which it came, resulting in no change of class. That is why I considered this an unacceptable mistake.

For the other two variables, I consider the previous curve much more influential to the next curve than the current slope, because the connection between slope and lateral acceleration is much weaker. Thus the slope class receives the lowest coefficient of them all.

In a representative simulation of 100 km, an average approximation error of 0.0053 was registered (not to confuse with the frequency or average value errors), meaning that for almost each 200 times that the tables are consulted, it was required to permit an error once of value 1 (one change in the current slope class to a directly upper or lower class). Depending on the amount and type of files used, the error can get as low as 1/1000. It suggests the strategy works quite well at least for the slope.

4.2.5 SPEED

Current classes as an input

As a first step, the speed was simulated from the same old parameters that the old simulation strategy was using (current slope, current lateral acceleration, and current speed).

These were the considered weights:

- Current slope: 1
- Current lateral acceleration: 1
- Current speed: infinite

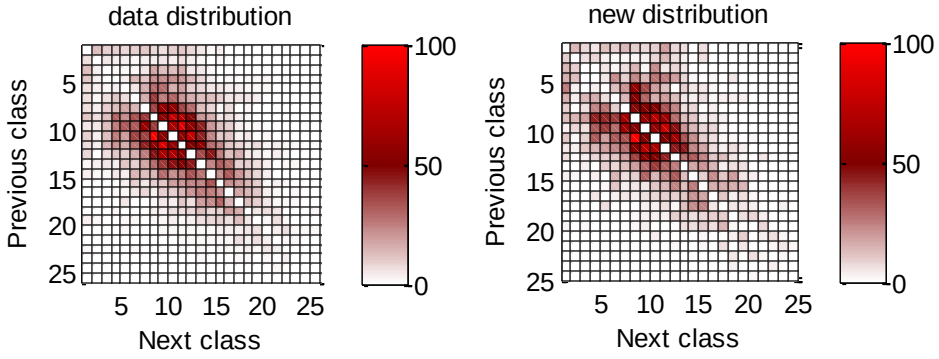


Figure 4.10 - Strategy 2: speed distribution using old inputs

The distribution speed graphs may not look bad at all (except for a higher dispersion) but, as expected, this did not provide the desired results.

When the 3V are plotted together, the result becomes incoherent as in the following example in the figure 4.11, where the simulated driver attempts to speed up through a significantly sharp curve instead of slowing down.

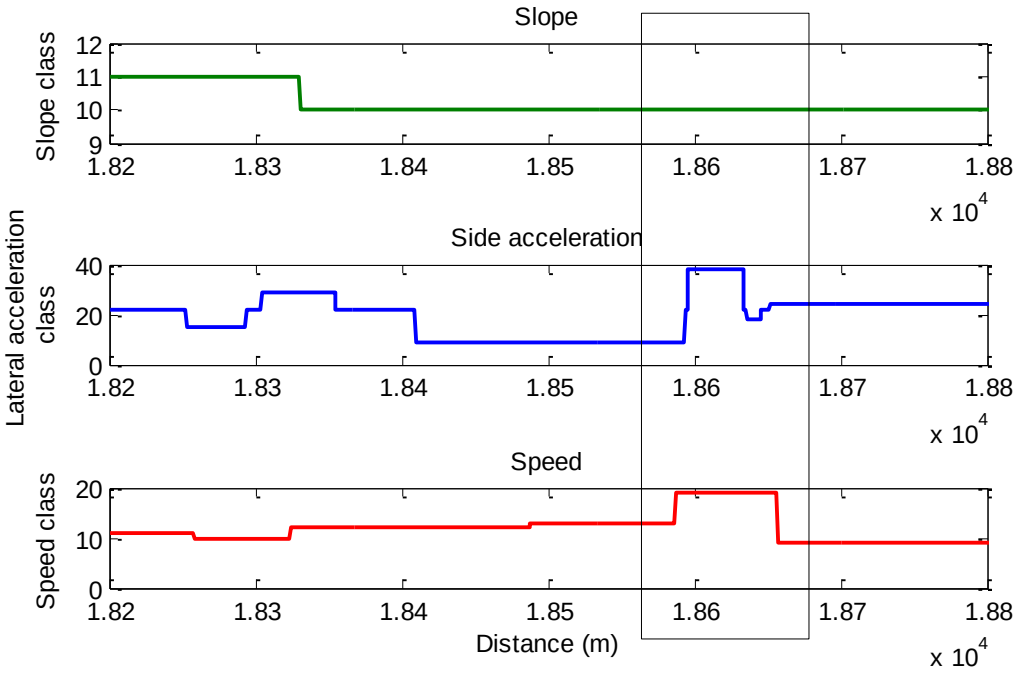


Figure 4.11 - Strategy 2: incoherent variable relationships

I expose the reasons:

First of all, the driver has more information while driving than us, he can see the road coming ahead and he can anticipate to the events with great foresight. A mild driver anticipates the curves from a long distance so instead of braking hard he starts to decelerate the car earlier.

Besides, whenever the algorithm sample the speed class for a section, it does it with the class of the oriented speed profile, in other words, it adds the final value the speed will acquire once the driver has completed the desired maneuver. It implies that, for example, it may be in the need of adding a reduction of speed caused by a curve that has not been reached or seen yet at our current simulation time. However, in the way this strategy logic works, the algorithm is forced to add the whole length of the speed section before it has information about the upcoming curve. This issue is highlighted with a rectangle at the figure 4.11 above.

So it seems reasonable and better focused to look at the events that are ahead of the driver, being the current speed the only variable that makes sense to consider of the current class values at each simulation step.

Upcoming classes as an input

The next step is trying to identify the events that may cause a speed change by looking in their surroundings for changes and values in the other variables. But the problem is in fact really complex in fact; I will illustrate it with a simplified example that does not include the slope for an easier representation.

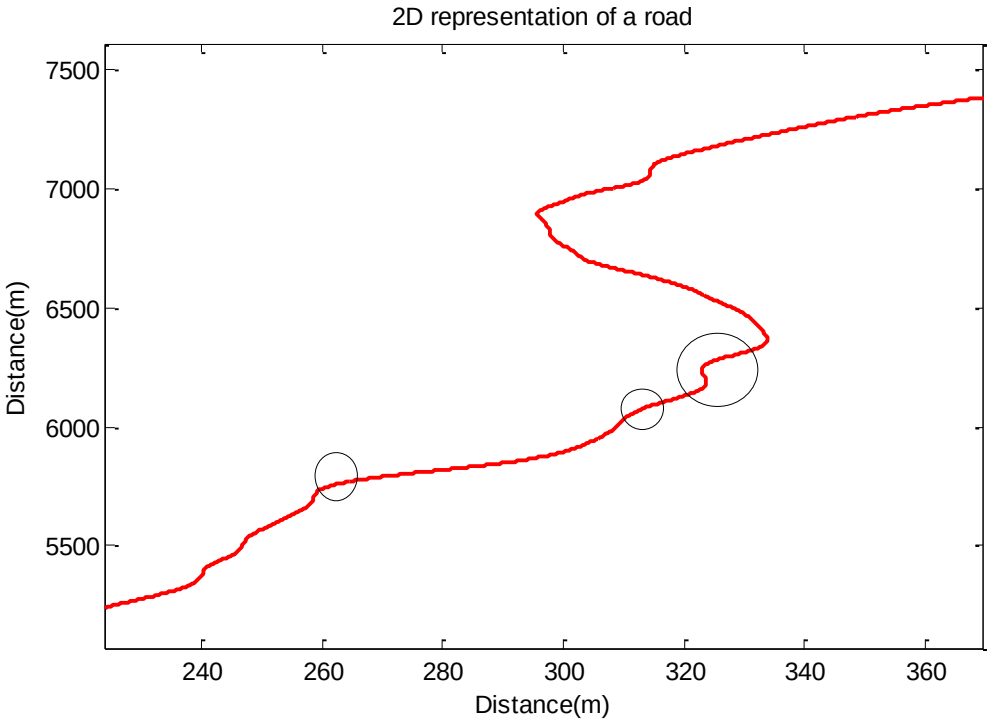


Figure 4.12 - Hidden curves

The graph above is a zenith projection of real data track in which the size of circles drawn at some curves are proportional to the lateral acceleration peak class of such curves. As shown, the same kind of curve is observed twice, but whereas after the first one comes a relatively long straight track, after the second a sharper curve hidden behind is encountered.

Effectively, if our speed generation reaches the position of the small circles, the simulation will decide what to add based on the lateral acceleration of such points. However, if somewhere else in the data it was encountered a huge straight track after a similar curve before, the algorithm could randomly pick it and add a huge constant speed stretch despite the following curves.

4.2.6 TOWARDS THE NEXT MODEL

Hence even another strategy else is needed. As these problems appear, the simulation models required to avoid them force the simulation to become more complex while the time required to program and test them grows exponentially. With that in mind, the problem was divided at this point, trying to solve the problem first without the slope, using only files where the changes of slope are not frequent or significant (land and city files, most of them).

4.3 SIMULATION STRATEGY Nº3: SPLIT SPEED

My next attempt to avoid the problem was splitting the generation of the simulated speed track in two or more stages so I could use more variables and therefore more information for the simulation of the speed. This modification only affected the speed generation because the slope and the lateral acceleration were already good enough.

At first, only the dependence between the speed and the curves was considered, leaving the slope for a later generalization of the model. At this point, the objective was to build a simplified model that only uses 2 of the 3V, and only if it proved itself valid, it would be completed. The challenge is that not only must the reconstructed track match the data statistics in both average values and distribution, but also the time in which the events take place compared to their surroundings is as important as the event themselves.

4.3.1 STAGE ONE: PRE-DIMENSIONING OF THE CURVES

To avoid the problems of the hidden curves, and knowing that usually the minimal speed is reached at the vertex of the curve. Therefore, in this stage thee whole simulated lateral acceleration track will be followed from curve to curve assigning to each curve the speed classes that will be reached there. Due to memory limitations, a simplification will be done: it is considered that in a single curve are never going to coexist more than 2 different speeds, one entry speed and one exit speed. For most of the curves, specially the loose and fast ones, both speeds will be the same. This is an approximation and assumes some error.

Now I am going to compare the real data with the result of the simulation of this first stage. This kind of comparison is only possible for single files as the purpose is to isolate the effect from any other source of randomness to have comparable roads. Frequently these results are much better than the real simulations where the data has been collected from multiple files.

This graphic shows the comparison between the original data minimal speed at the curves (left) and its simulated equivalent (right).

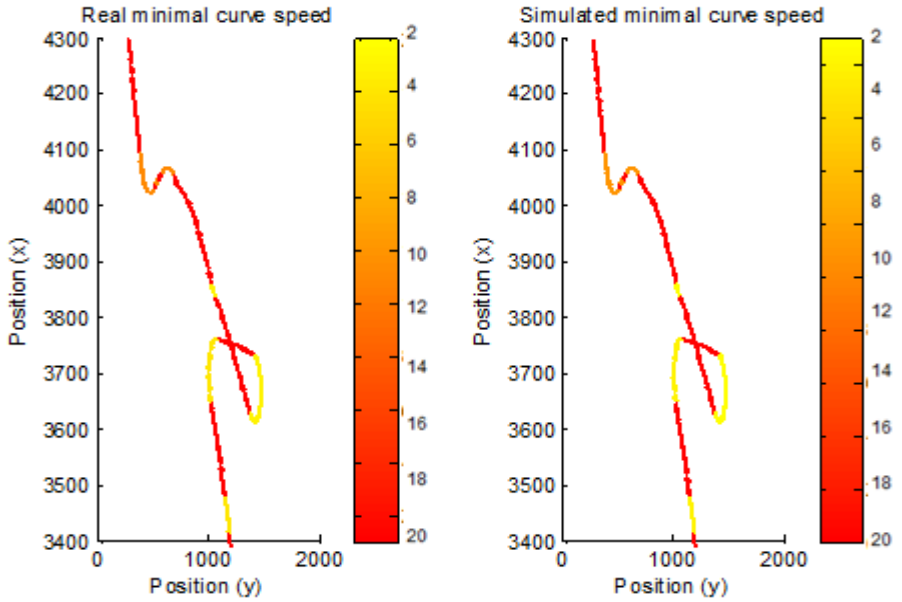


Figure 4.13 - Strategy 3: Comparison of the speeds at curves

Now the complete two road speed curve profiles are overlapped:

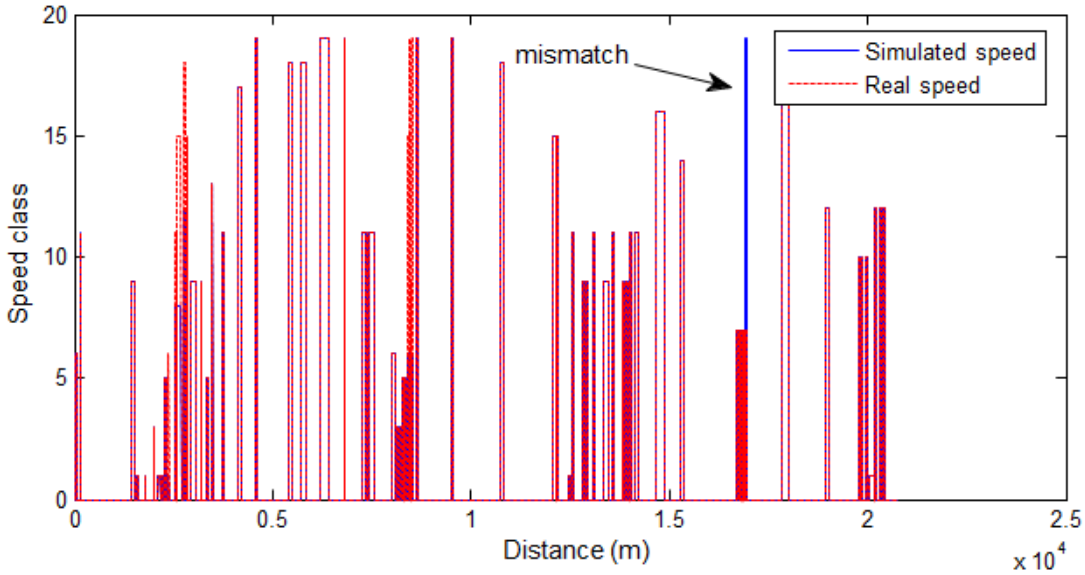


Figure 4.14- Strategy 3: Comparison of the minimal speeds at curves

Again, the comparison has been created from a single file. The high, fast, abnormal outliers (mismatch) are caused by the randomness of the simulation at the previous steps and they introduce new problems when non observed situations are reached for the first time.

4.3.2 STAGE TWO: FILLING THE GAPS

In the previous stage, a speed guideline has been created using the curves. Now, the speed profile at the speed curves is already fixed but the sections between two curves must still be completed. It is a very different problem because the straight sections can be in some cases extremely long (in highways for example) and an unbounded number of overtaking maneuvers can take place, making impossible to reserve a fixed space in the tables for each one.

To deal with it, the straight section must be divided and each event added individually and then resampled until the next curve has been reached. It is not possible to store exactly the distance to the upcoming events due to memory limitations as explained before, it has to be classified first. The size of the classifying length vector was set to 72 different classes, to guarantee that the error accepted is in any case lower than 10% or too much information at the classification would be lost. Another consequence is that a small difference of distance can ruin completely the simulation as it could represent the difference between a value falling inside a curve or outside of it. To avoid it, all the output distances at the table are stored as ratios instead of fixed distances, so they can be scaled to match any curve regardless of the exact size.

In the following picture, the solid lines represent the speed at the curve sections, previously calculated at the first stage. The dash-dotted lines represent the straights tracks that fall between the curves and show how they are filled.

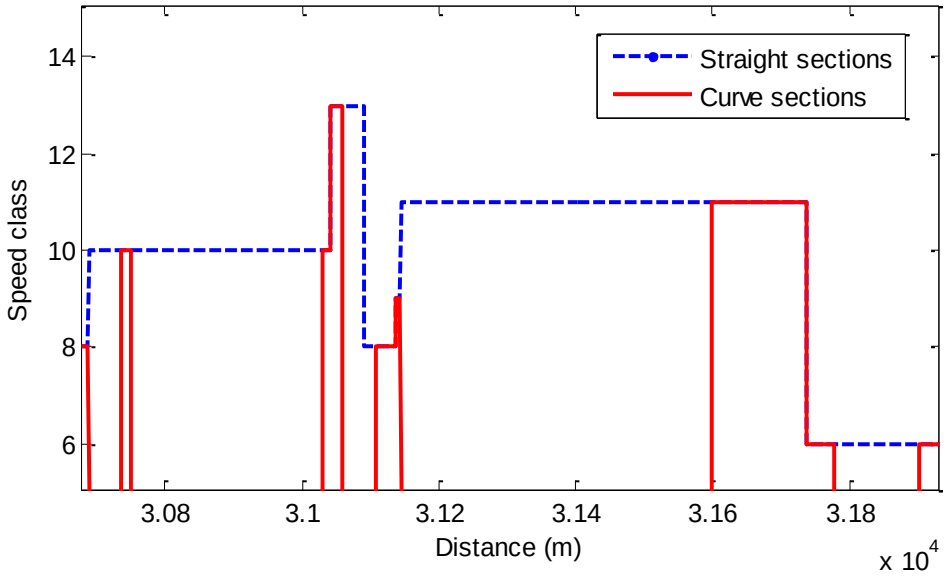


Figure 4.15 - Strategy 3: Generation of the straight sections

Most of the times, if both sides of the straight stretch hold the same speed class value, and the distance which separates them is low, the new track will be constant. When there is a change, this second stage will decide when the transition takes place. In longer sections, several changes can occur within two curves.

4.3.3 RESULTS OF THE SPLIT SPEED STRATEGY

Unfortunately, the results are still very poor, even in the simplified model. Failing in reaching a better correlation would make senseless to try to generalize the model to a more complex on that includes the slope. It would only get worse.

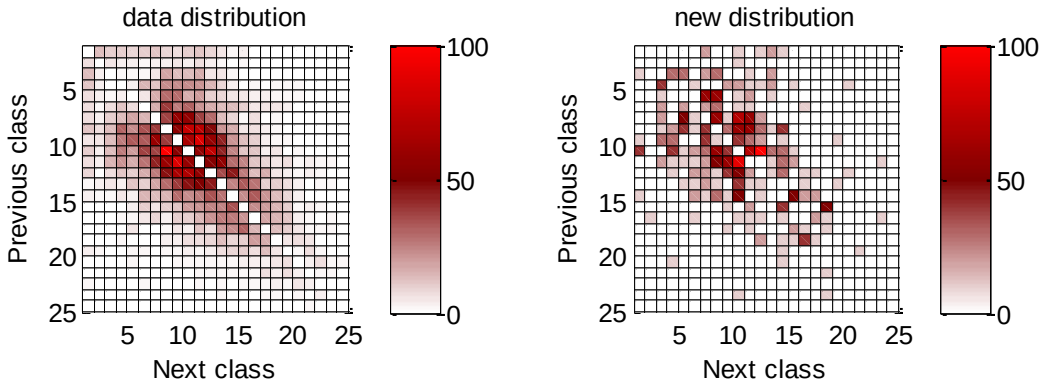


Figure 4.16 - Strategy 3: Speed distributions

Having discarded coding bugs, and realizing that the plotted files seem to be coherent, the causes are analyzed and an extremely high approximation error is observed in both stages, being the first one critical.

At this point, the incoherent variables relations of the previous strategy have been solved, at the price of ruining the speed distribution in the process.

For the stage 1 table:

Number of entries: 51 (speed) x 43 (curvature) x 75 (length) = 164.475

Number of filled entries: 5015

% of entries filled = 3’05%

For the stage 2 table:

Number of entries: 51 (speed) x 51 (speed x 75 (length) = 195.075

Number of filled entries: 6153

% of entries filled = 3’15%

If it were wanted besides to use only files that belong to a single kind of driver or vehicle besides the track type, the result would be even lower.

Although not all the entries need to be filled (for example a speed of 250km/h or above in the middle of a sharp curve), the percentage of the tables filled with the data that is available is so low that the algorithm is forced to look for an approximation almost at every step. That approximation methodology would only work with a much lower approximation rate. On top of that, this was only the simplified model. The slope has yet to be somehow added, diluting the density of the matrix by a factor of 21 (the number of slope classes), even if the limitations of memory were avoided.

Therefore I could not encourage continue in this direction and a new strategy had to be found.

4.4 SIMULATION STRATEGY Nº4: SYNCHRONIZED BLOCKS

Arrived to this point, the desired strategy must be one that allows the post-processing of the data by never exiting the table positions for which there is stored information. At the same time, it is vital to introduce some kind of length control or abnormal transitions would appear.

As discussed in section 4.2.1 it is not possible to keep an explicit control length while generating the track from a single table in a single step. Therefore, to stick to the available data, the explicit length control has to be sacrificed. However, the length and length related parameters can be still considered implicitly making use of a partial randomization as it will be described in this strategy.

4.4.1 CREATING BLOCKS

Instead of resampling every single variable change, the new idea implies cutting the files into smaller blocks of information of variable size. These blocks contain inside the profiles of the 3V, copied exactly as they are. Those will be our minimal units of information to be randomly reassembled on a posterior step.

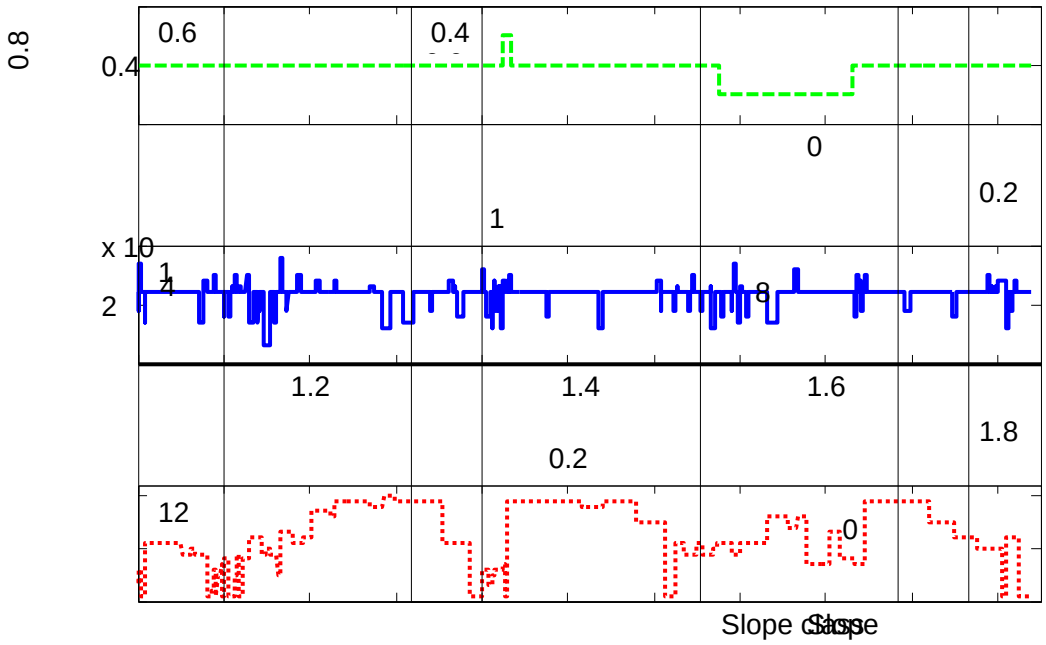


Figure 4.17 - Strategy 4: Division of the data in blocks

Under these conditions, the way to keep some sort length control of the variables is to allow the size of the blocks to be variable and that size must be directly related with the length of the events involved in it. So the new system would have two clearly different parts, in the first one the blocks are created, and in the second one they will be sampled one after another until the desired simulation length is reached.

The clearest limitation is that the process is not fully randomized; instead, the information is saved already grouped. This also represents a huge source of advantages as a lot of codependency relationships are not skipped even if they are not detected; circumstance of great importance for this problem given the limitations of the tables that can be created. Also, the statistics are very likely to match since all the transitions take place inside the blocks and not at the joints.

4.4.2 CHOOSING THE ELEMENTS SIZE. SYNCHRONIZATION.

If the extremes of the each element provide information about their connections, the other degree of freedom, the size of the element, has to be responsible for the length control. On a first approximation, it is attempted to cut in those points for which the variables reach the same state of evolution, namely, in which points the distance from the previous change is equal to the distance to the next change at the same time for the 3 variables. I will call this “synchronization of the 3V”. By this way, it is guaranteed that adding two completely random blocks the lengths of the transitions at the block joints will necessarily match an existing situation stored in the data.

For such purpose, three new Matlab vectors are created to keep track of the percentage of development of each variable, where 0% represents the previous change, 100% the next change and 50% the ideal point of interruption.

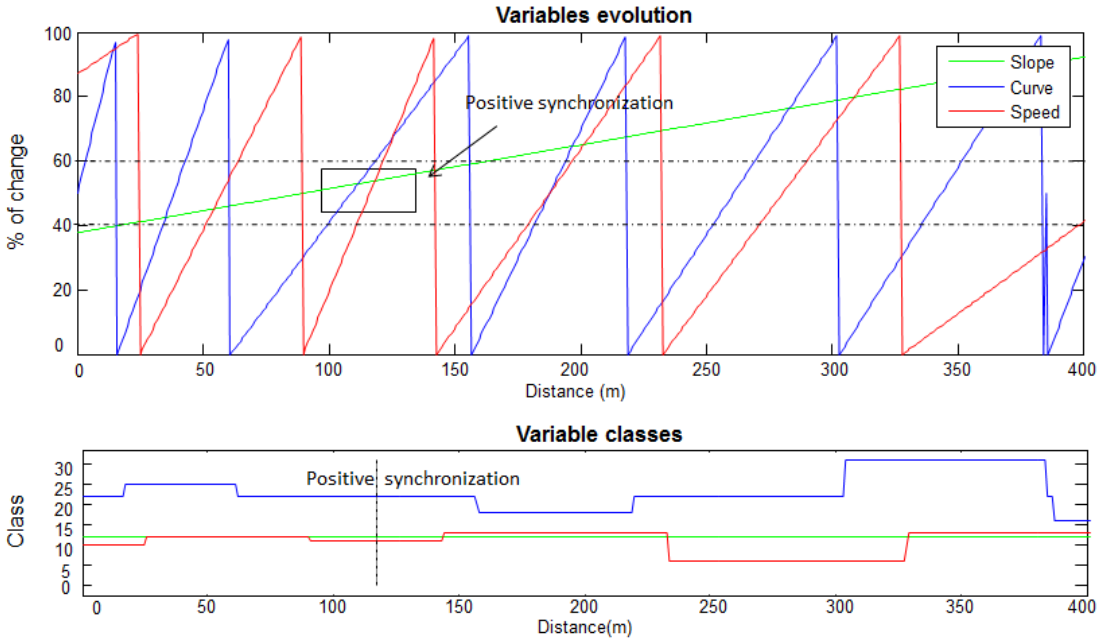


Figure 4.18 - Strategy 4: Total synchronization

The area defined by the rectangle in the figure above is an ideal point of cut. Due to non-idealities, it has to be an area with a certain tolerance instead of a single point or otherwise only a few synchronizations would be recognized.

4.4.3 TOLERANCE RANGE

The relationship between tolerance and the number of blocks is clear: the higher the tolerance is, the higher the number of occurrences that will be found and subsequently the higher the number of blocks that will be created. The following table contains an analysis of the evolution of the average block length for each track category and tolerance level.

Table 4.5 - Influence of the synchronization tolerance

Tolerance range	Mountain	Land	City
Number of files	37	19	23
Data lengths	1035km	623km	527km
±10% (40-60%)	895	241	383
Average block length (km)	1.16	2.59	1.38
±20% (35-65%)	2031	510	898
Average block length (km)	0.51	1.22	0.59
±30% (30-70%)	3760	932	1685
Average block length (km)	0.28	0.69	0.31

Looking at the data a tolerance of ±20% (range of 40%) seem to generate enough blocks to guarantee that the average block length remains under 1km in all the cases.

When a possible cut is detected, usually more than only one point will be eligible for it. The system must prioritize those points in which the deviation from the 50% of evolution is minimal. As the evolution of a variable between two extremes is always a linear function, that point corresponds to the middle point of the region for which the 3V are synchronized.

However, there can be of course even roads or with constant slopes, especially if the driver is driving through a city. This leads to the need of defining extra conditions for the creation of the blocks to assure there can never be 20 kilometers long files that are divided only in two or even just one single block. This condition will be a distance tolerance parameter explained below. I will call this "partial synchronization" and it allows us to cut once more whenever the following requisites are satisfied:

1. The lateral acceleration and the speed (but not the slope) are synchronized under the tolerance and conditions described above

2. The distance from the cut point to both the previous and the next slope change is greater or equal than the distance tolerance value

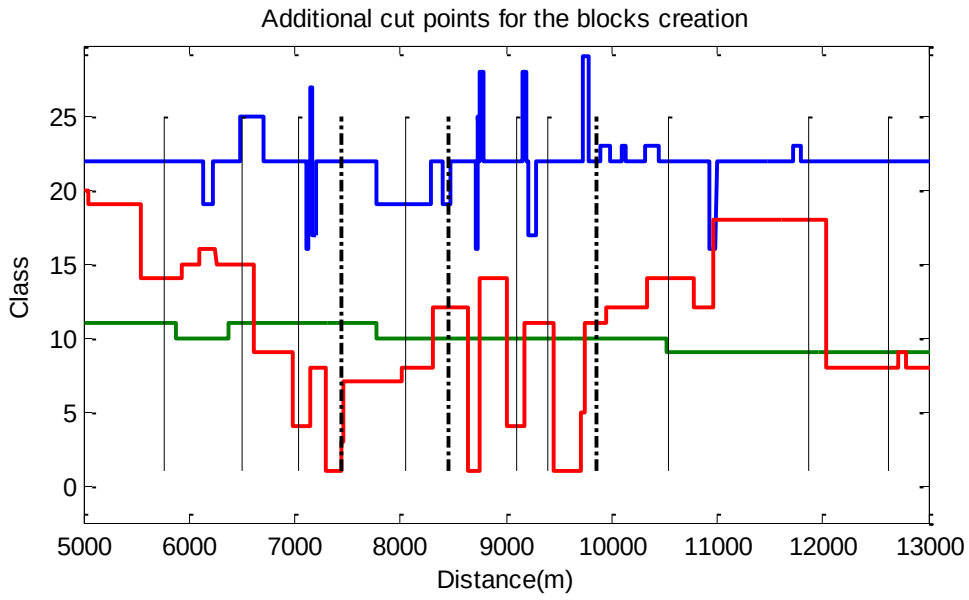


Figure 4.19 - Additional points created with partial synchronization

In the figure 4.19 can be noticed the cut point displayed: with dashed lines, the points obtained naturally from the initial conditions, and with dotted lines, those additional points that are generated through the relaxed conditions considered just above. The effect becomes of greater importance when there are very little changes of slope but the analog approach is not required for the other 2 variables due to their characteristics.

Figure 4.20 - Influence of the distance tolerance in the number of blocks created

±10% (40-60%) / 500m	Mountain	Land	City
Number of files	37	19	23
Data lengths	1035km	623km	527km
Previous blocks	895	241	383
New blocks	**1226**	**647**	**898**
% Increase	37%	168%	134%

As seen at the table 4.6, the partial synchronization becomes important at city and land tracks where the number of blocks created is more than doubled.

4.4.4 LIMITATIONS

Impossible roads

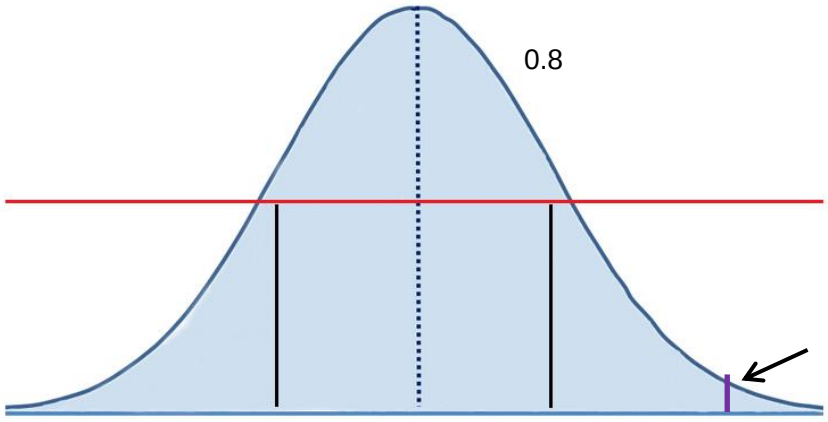
If the generated roads were displayed using the 2D or the 3D representation detailed in the section 3.3, it may result in the creation of impossible roads as no importance is given at the orientation of the car at the ending of the blocks. It could result in unrealistic road overlapping if added consecutively several turns in the same direction, which however is not relevant for the purposes and scope of this project.

Impact of the tolerance in the lengths distribution

A second underlying inconvenience is the fact that after having allowed a tolerance of, for example, 20%, durations in a range from 60% to 140% of the original data can be generated through random resampling. These situations take place very rarely and only in the cuts but generate non-observed data.

Normalization of the lengths distribution

A second underlying consequence is caused by the fact that the system will find more common durations more often in the resampling process. The best analogy are the recessive genes in genetics, the only way to preserve a recessive gene requires both the constituting elements to be recessive as well. As a consequence, if the probability of sampling a recessive value is 10%, the possibilities of a recessive in the next generation from the joint of two genes are of only 1% (0.1^2). The same idea applies to the lengths and the most extremes situations will only be observed where both parts of the block hold the longest or shortest values respectively. Therefore, the chances of obtaining such extreme values decrease quadratically.



The global effect is that the distribution of the lengths reassembled will lean towards a normal distribution in which the average values will be resampled more often, and the tails of the distribution will be heavily penalized, giving a new normal curve with the same average but a much smaller deviation.

From a statistical perspective, given that the joint between two blocks is given by the addition of two randomly distributed variables, the resulting distribution will become a normal distribution if enough variables take part in the process **[5]**. Although only two variables are added and the conditions are not satisfied, the resulting distribution will be altered anyway.

***Central limit theorem*:** Given X_1, X_2, …, X_N a group of random variables independent and equally distributed with average **μ** and standard deviation **σ.**

Let $S_N = X_1 + X_2 + \dots + X_N$. Then

$$\lim_{n\to\infty} Pr\left(\frac{Sn-n\mu}{\sigma\sqrt{n}}\right) \leq z = \phi(z) \qquad \textbf{(4.3)}$$

In our case, the number of variables (n) is only 2 so the effect is not too big.

4.5 SIMULINK IMPLEMENTATION

The final resulting files consist of two M-files that contain all the information necessary for the Matlab and Simulink simulations respectively:

FOR MATLAB - "blocks.mat" contains the following variables:

Block_table: a struct that holds actually all the information contained in the other 3 tables (Index_block_table, Simulink_table and Stored_blocks) together. It is the most natural way to keep together related information of different sizes but it is not appropriated for Simulink.

Index_block_table: an auxiliary table used to speed up the lookups by pre-recording the number of valid output options for each input.

Frec_data_slope, frec_data_curve, frec_data_curve2, frec_data_speed: frequency tables used for the statistics.

Avg_slope, avg_curve, avg_speed: average class values used for the statistics.

Slope_vector, lateral_acceleration_vector, speed_vector: classification vectors used to classify the data and define the positions of the logical indexing of slope, curves and speed.

Long_data: the total number of meters analyzed to build the tables. It is needed to compensate the statistics to the length of the data.

FOR SIMULINK - "for_simulink.mat" contains the following variables:

Index_block_table: an auxiliary table used to speed up the lookups by pre-recording the number of valid output options for each input.

Simulink_table: the main table that points to the positions of the new events. It links the inputs to the outputs.

Stored_blocks: the table that stores the full vectors containing the slope, lateral acceleration and speed. Each block is stored on a different unique entry no matter its initial conditions.

Provided that Simulink is not able to hold two different simulation times by expanding a signal after it has been generated, the model was split into two blocks activated alternatively

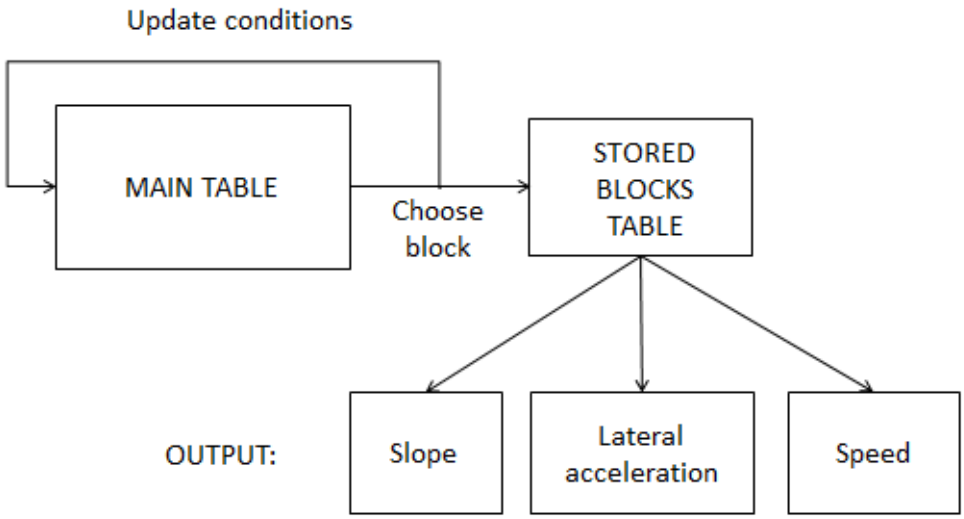


Figure 4.21 - Tables flowchart

The main table is responsible for selecting which information must be copied while the second table copies the information inside of each block during a number of steps equal to the length block, at 1 meter of data per step.

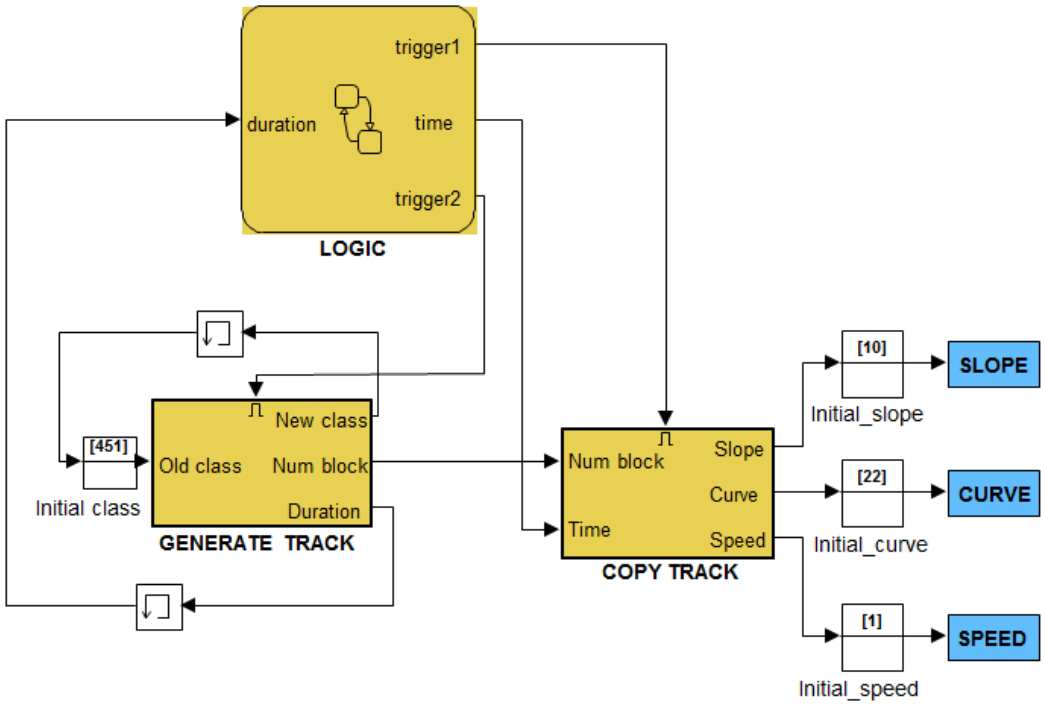


Figure 4.22 - Simulink model

The initial classes must de set to eliminate the delay of the reception of the first value, since the “COPY TRACK” block is not executed at the first time step.

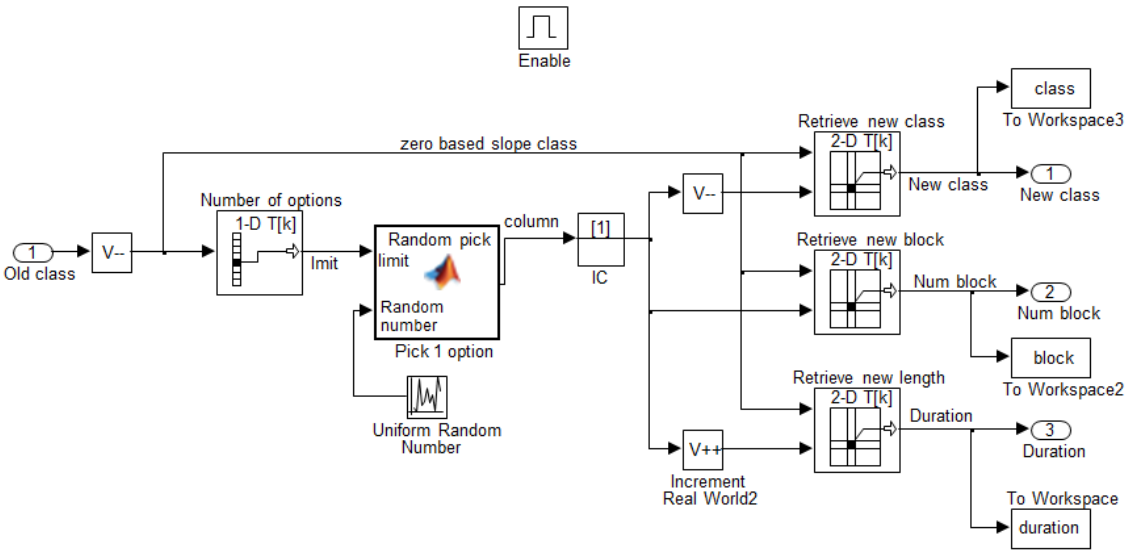


Figure 4.23 - Detail of the "GENERATE TRACK" subsystem

A closer look to the model shows a third table block (number of options) that contains the table “Index_block_table”. This table is used to precalculate the number of possible output blocks given a fixed entry without searching inside of the main table. The user defined function (Random pick) just chooses one of those possibilities.

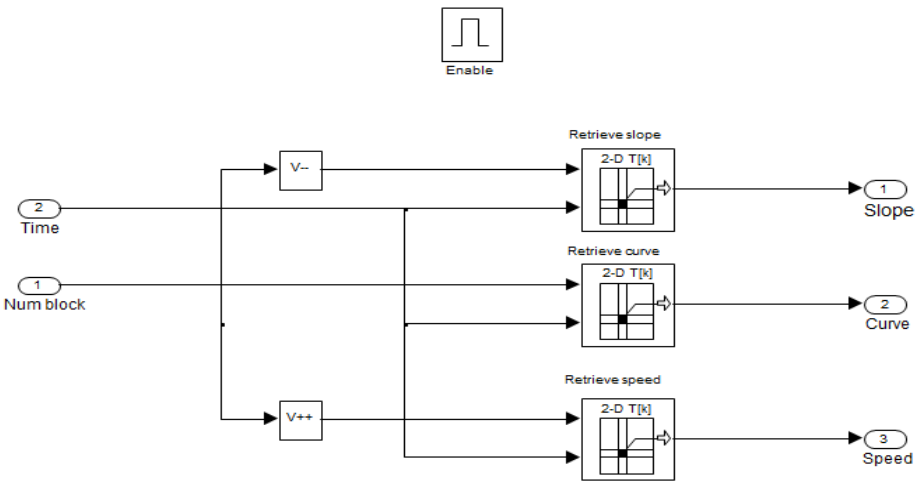


Figure 4.24 - Detail of the "COPY TRACK" subsystem

The second block just outputs one value of each variable per time step.

5 RESULTS

5.1 COMPARATIVE OF THE DIFFERENT STRATEGIES

STRATEGIES	Slope	Curves	Speed	Randomization	Problems	Judgement
Strategy 1 (all in one)	Bad	Bad	Bad	Full	Incoherent fast changes	✖
Strategy 2 (modular)	Good	Good	Bad	Full	The speed is incoherent. Slower	✖
Strategy 3 (split speed)	Good	Good	Bad	Full	The speed doesn't match. Much slower	✖
Strategy 4 (blocks)	Good	Good	Good	Partial	Some limitations	✔

Table 5.1 - Strategies comparative

Taking into account the limitations previously described, the fourth and last strategy is the only one that keeps the results coherent at the same time that preserves the statistics with acceptable deviations. Not less important is the fact that the first strategy had no potential of improvement while the second and the third grew extremely complex and slow to try to gain accuracy to a state in which their Simulink implementation would have been far from simple.

All the following results have been calculated using that final strategy based in blocks.

5.2 AVERAGE CLASS AND FREQUENCY VALUES

These are the statistical results:

Frequency and average value stats:

Table 5.2 - Average frequencies

FREQUENCIES 10 x 1000km simulated	SLOPE		CURVE		SPEED	
	Frequency error %	Value error %	Frequency error %	Value error %	Frequency error %	Value error %
Mountain road	4.63	1.12	2.01	0.13	1.97	1.16
Land road	5.03	0.14	2.60	0.11	2.40	2.83
City road	1.76	2.95	22.44	0.21	10.71	4.65

They were calculated out of the average of 10 simulations of 1000km each one.

Number of blocks created under a tolerance of +-10% and a distance tolerance of 500km:

Table 5.3 - Average classes

AVERAGE CLASS	Mountain (37 files)	Land (19 files)	City (23 files)
Data lengths	1035km	623km	527km
Total blocks	1226	647	898

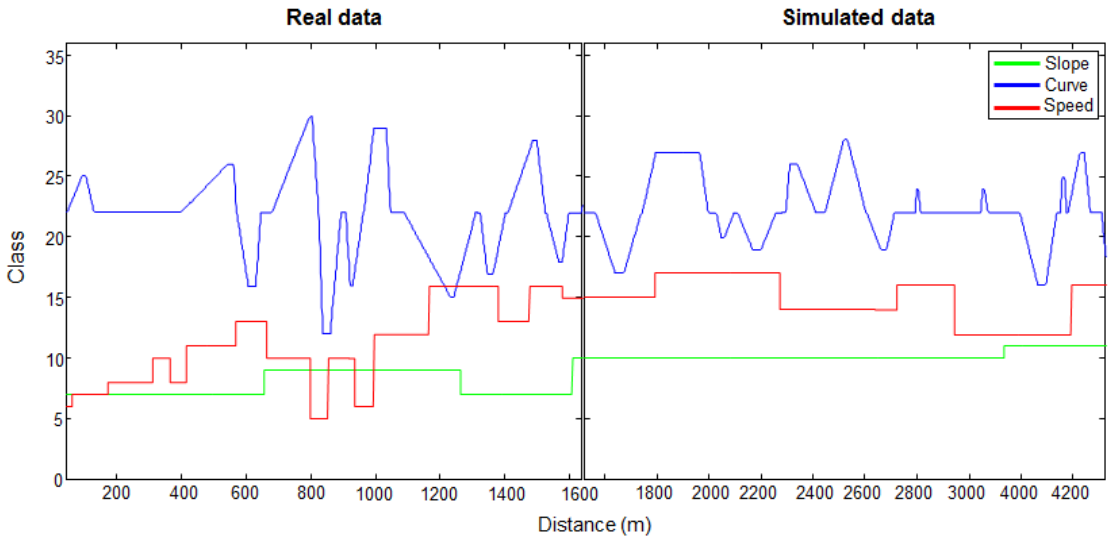


Figure 5.1 - Comparative between a real and a simulated track

5.3 DISTRIBUTION GRAPHS

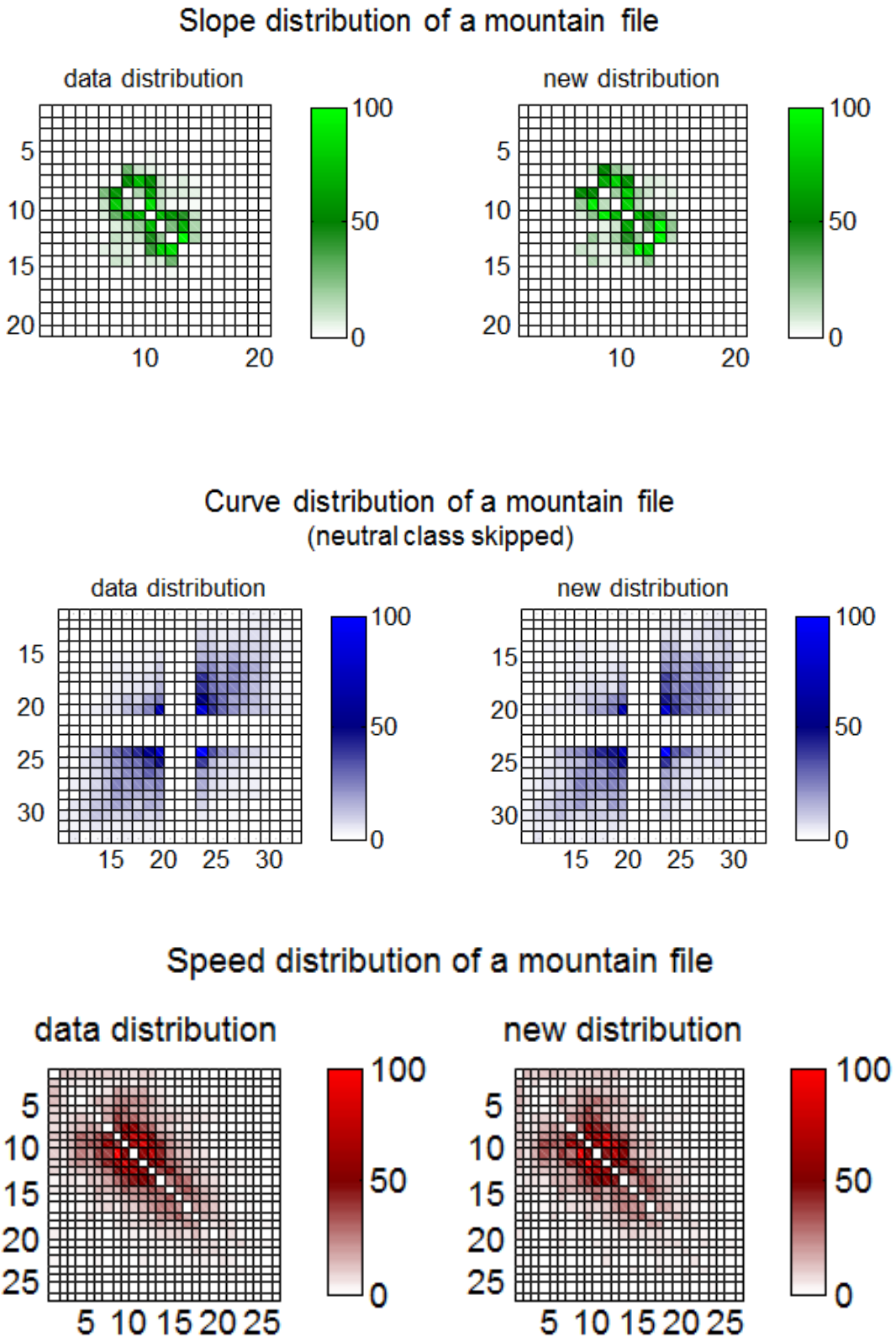


Figure 5.2 - Final results: mountain road distributions

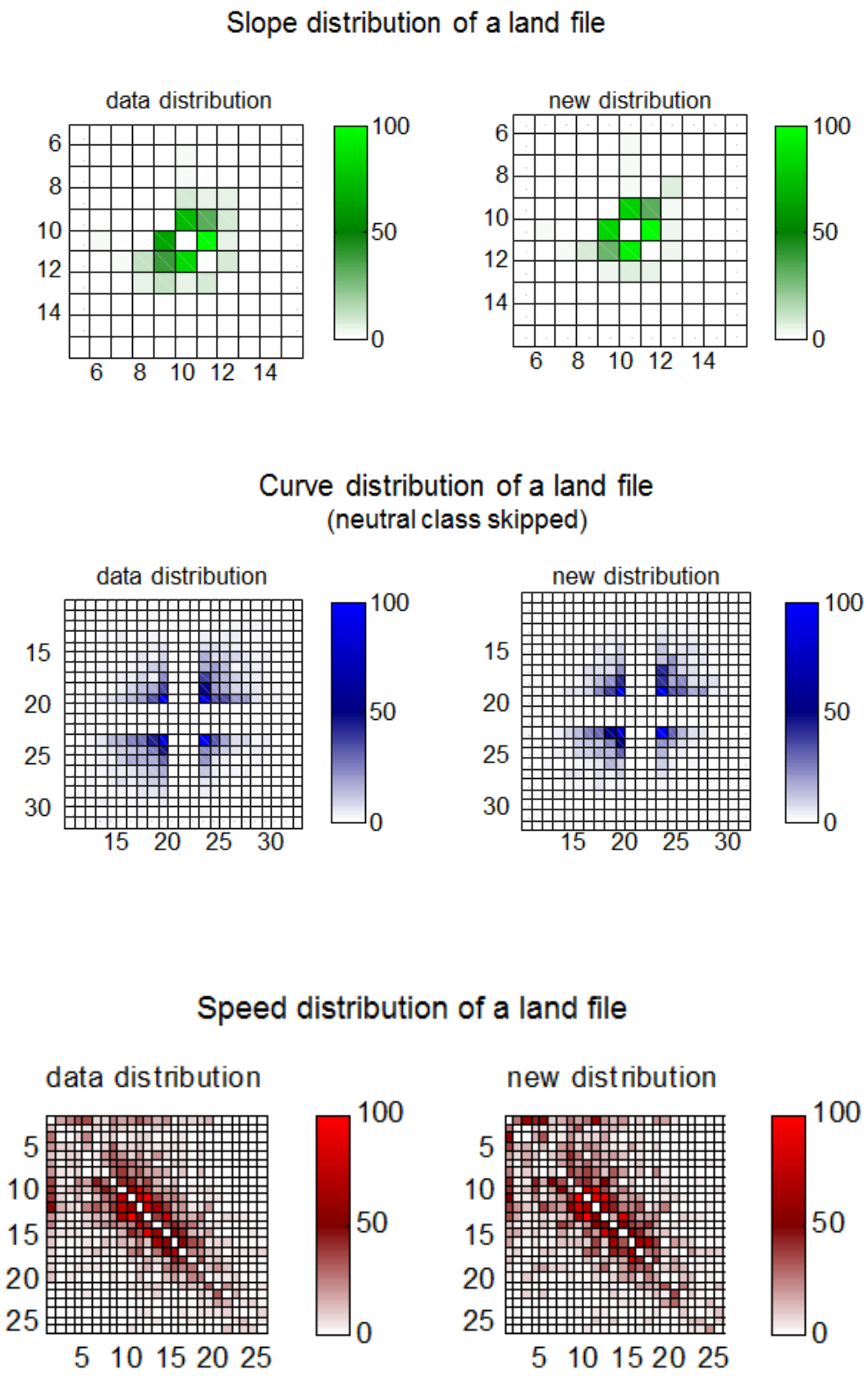


Figure 5.3 - Final results: land road distributions

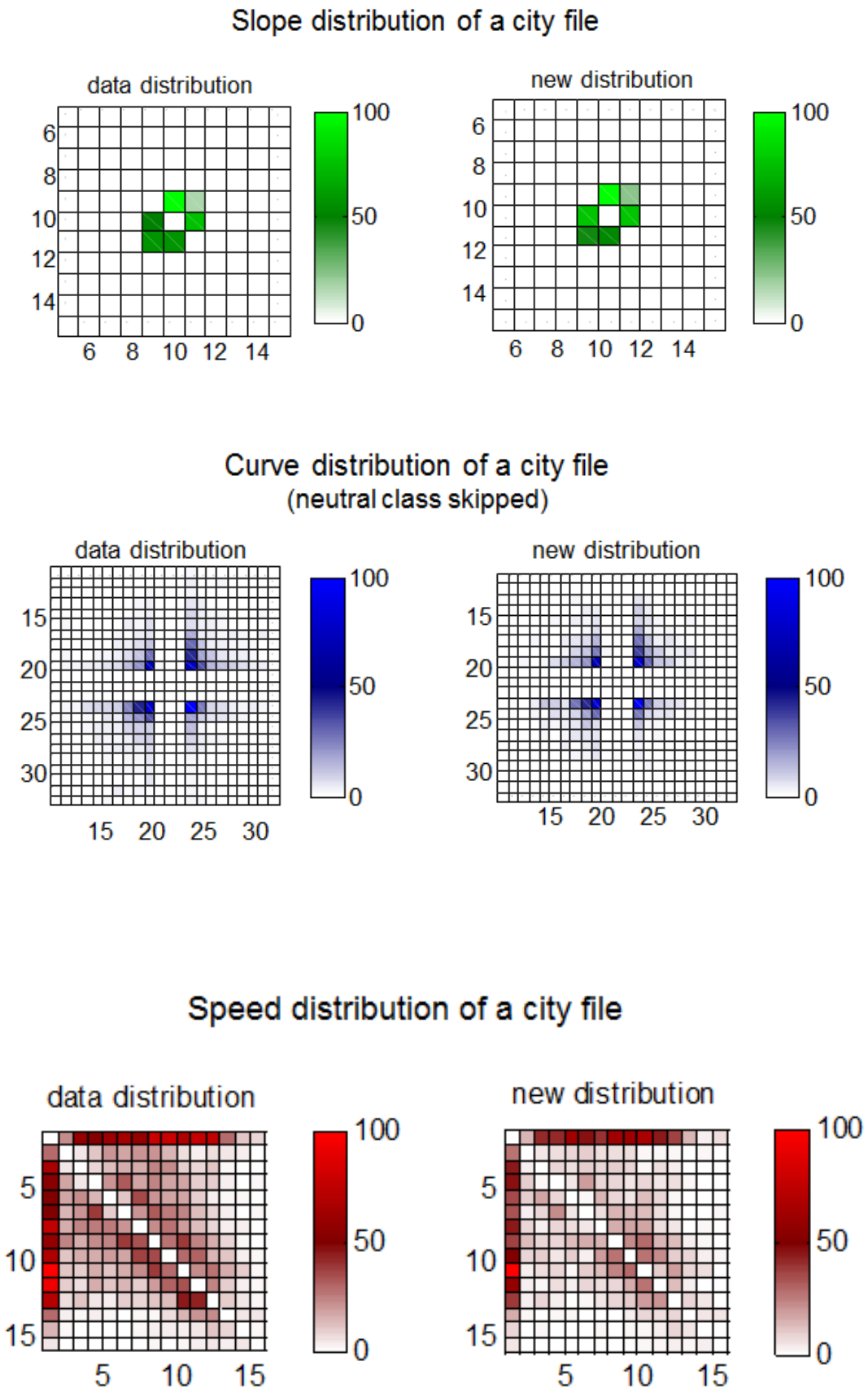


Figure 5.4 - Final results: city road distributions

The following graphs were all obtained from city files although the distribution that the other road types follow is similar. All of them were calculated too with a sync tolerance of ±10% and a distance tolerance of 500m.

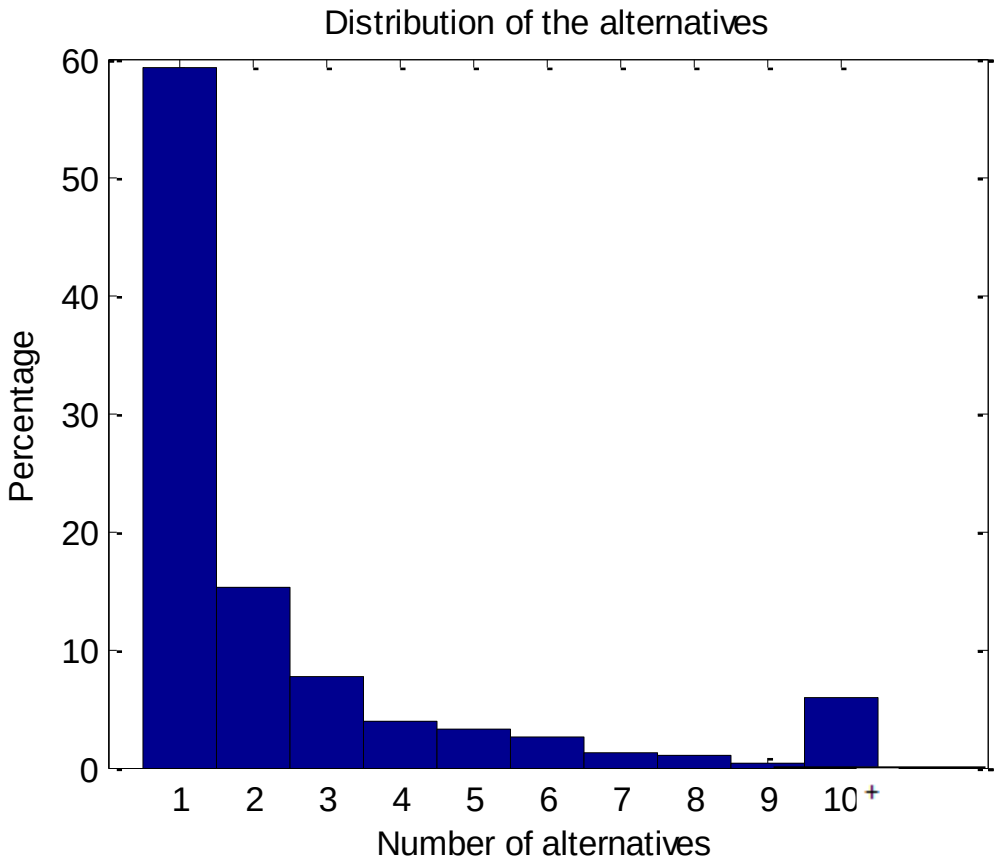


Figure 5.5 - Analysis of the number of different possible alternatives to continue a given block

On the figure 5.5 it is shown how many different options we have to continue a certain block, giving a good measure of how random our road can be. It seems to follow an exponentially decreasing distribution, where the last class represents the number of occurrences with 10 or more alternatives.

Unfortunately, most of the times (around 60%) the algorithm will not be able to choose the next block as there is only one possible option. However, forcing the cuts to be made only when the lateral acceleration class is 22, as well as feeding the system with more data will improve this distribution. If use all the data available were used, while maintaining the total number of classes and therefore of possible reachable situations, the result would be that the tables will get more populated, displacing the distribution to the right.

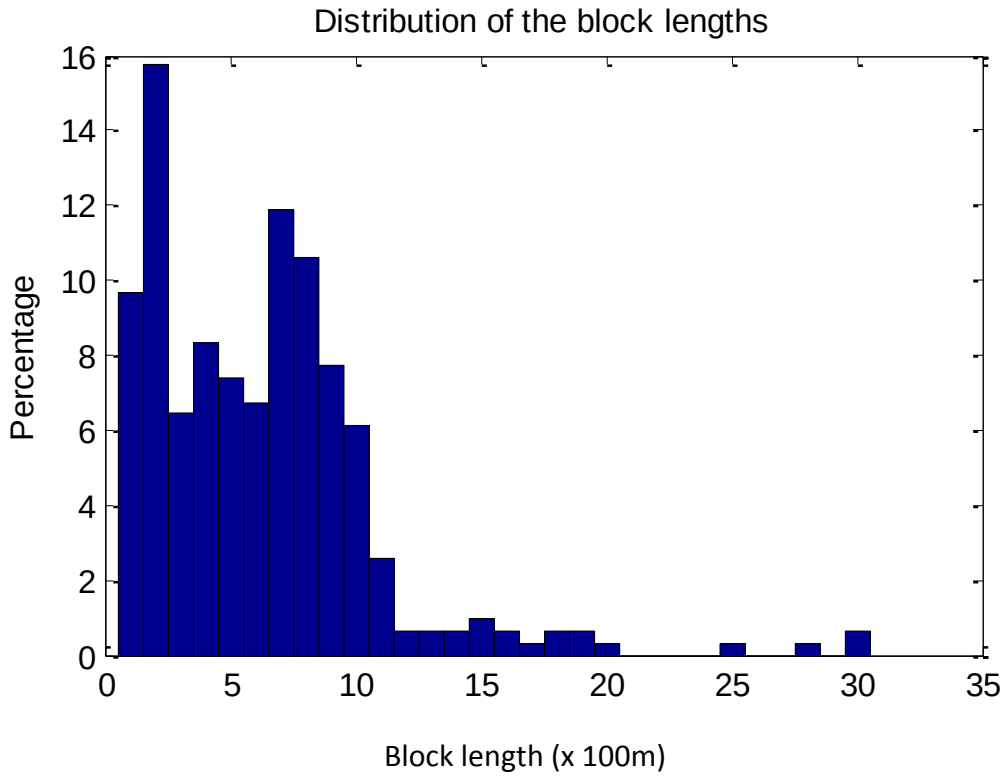


Figure 5.6 - Analysis of the distribution of the resulting block lengths

Transformed to a representation of the lengths of those blocks, the expected results were obtained: the lowest lengths happen more often while the conditions of partial synchronization limit the longest block. However, there can be observed outliers of up to 3km, or those particular situations in which despite the distance tolerance, the speed and the lateral acceleration do not synchronize for a long time. These outliers would be minimized incrementing the sync tolerance range.

But the best way to characterize our results is an analysis of the number of variable changes that take place inside each block. All the classes are displaced one position so the first class represents 0 changes recorded, the second class 1 change and so on. The slope is the only variable that can fill the first class as a consequence of the partial sync.

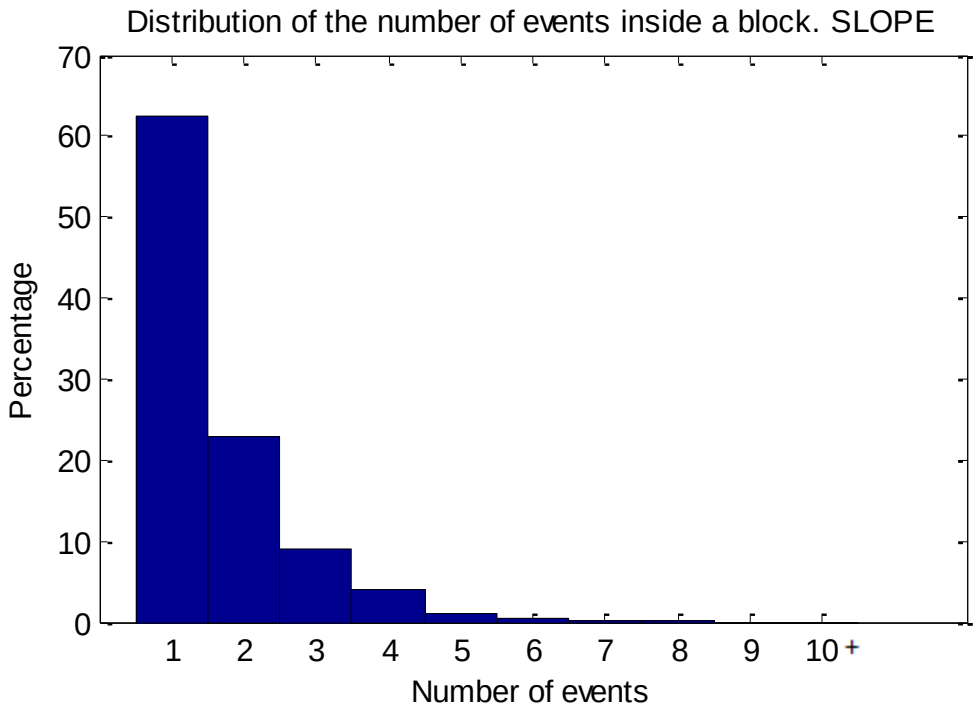


Figure 5.7 - Analysis of the distribution of the number of slope changes inside a block

The results suggest that there is a few number of events registered inside of each blocks for the three variables, proving that this block splitting strategy provides often units of information almost as small as the original strategies of cutting at each event. It reinforces the assumptions made at the definition of the strategy that creating enough blocks of a reduced length would provide a viable alternative statistically.

Naturally, there is still room for improvement, and as shown in the figures 6.8 and 6.9 the lateral acceleration and the speed sometimes hold a big number of changes inside a block. Those particular cases should be analyzed because it is not desirable to have such long blocks. It is also remarkable that those outliers happen often at special particular classes indicating an unexpected pattern and a possible bug in the code or unidentified special circumstances in the data.

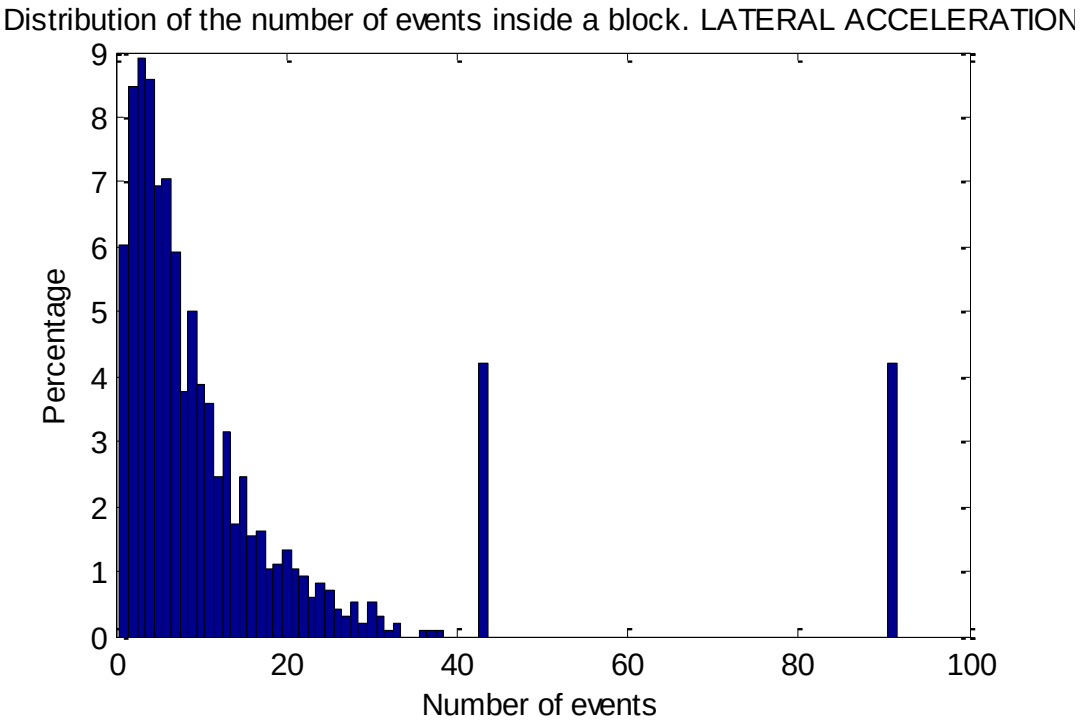


Figure 5.8 - Analysis of the distribution of the number of lateral acceleration changes inside a block

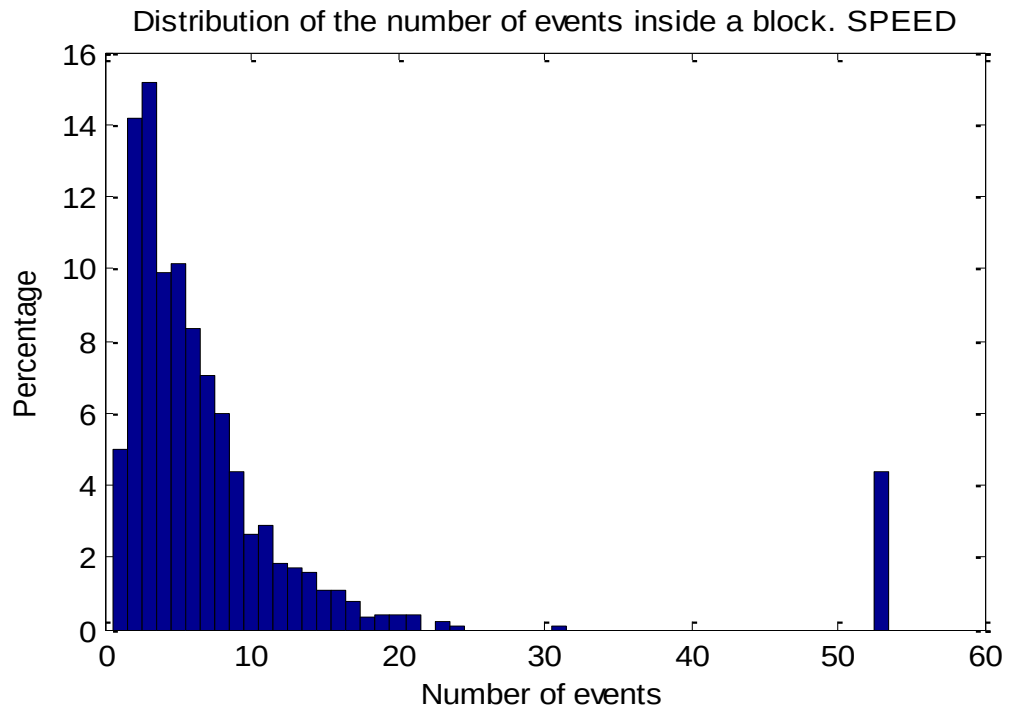


Figure 5.9 - Analysis of the distribution of the number of speed changes inside a block

6 CONCLUSIONS

6.1 OVERALL RESULTS

From the comparison of the statistical results can be deducted that the simulation model satisfies the expectations and goals that had been set. These objectives have been reached regardless of the track type studied because the distribution graphs match and the average class and frequency values are under limits. Only the city roads deviate significantly from the values and that should be analyzed and improved.

The code written is fast enough although the focus has been on the organization, readability and modular writing to ease future work of other persons that may use it, allowing modifications in any part of it leaving the rest unaltered. Nevertheless, whenever it did not compromise the characteristics exposed above, the code was optimized to make it faster and as versatile and minimalist as possible. As an example, the distance at intervals of 1m was chosen as the reference step instead of time, to reduce the number of points and speed up the code. Having exposed that, the speed of execution is much more important in the Simulink model. I am satisfied with the state of both the Matlab code and the Simulink model developed, although naturally there is always margin for improvement.

The fact that the previous student did not leave any written document made it extremely complicated to understand her work. It forced me to spend more time tracing back the reasons behind her previous attempts, deriving in extra work that caused me a certain delay at the early stages of the project. To prevent that, this document collects not only the working strategy but also the failed strategies, the reasons why they were tried and the reasons why the failed; to become an aid to whoever that work in similar projects.

6.2 FUTURE WORK

Apart from considering improvements to deal with the limitations detailed at the section (4.4.4), the next step must be the creation of another program that given the oriented speed profile simulates the speed, from the calculation of the required acceleration and the car response. That would complete our statistical driver.

A detailed study of the influence of the tolerance parameters would help to tune up even more the statistical match. The code should be altered to block the creation of blocks out of the no-curve sections, expecting an increase of the average number of possible alternatives to continue a block, and probably of the statistical resemblance of the simulated city roads.

Also there is room for performance improvements in the algorithm used for the classification of the speed and the way the information of the blocks is stored at the tables. One possibility of a memory saving implementation consists of storing only the lengths and class values of the changing points of the blocks instead of all their values for every meter. Memory savings for that table of around 90% can be expected, but it is not of priority and it was not coded for time reasons.

7 REFERENCES

[1] F. Küçükay, T. Kassel, M. Eghtessad & H. Kollmer (2008): Requirement engineering using the 3D method. Institute of Automotive Engineering, Braunschweig, Germany.

[2] F. Küçükay & G. Hohenberg (2008): CO2-Potential in Customer Use by Intelligent HEV-Control Strategy. Motor & Umwelt, Braunschweig, Germany.

[3] U.S. Standard Atmosphere, 1976, U.S. Government Printing Office, Washington, D.C., 1976.
(Formula of the altitude pressure)

[4] U. Hintze, Studienarbeit: Erweiterung eines statistischen Fahrumgebungsmodells zur Berücksichtigung von Geländetopologien (2010). Institute of Automotive Engineering, Braunschweig, Germany.

[5] M.D.Ugarte, A.F.Militino & A.T.Arnholt (2009): Probability and Statistics with R, CRC Press. Chapter 6.5.1.2, pg. 215
(Central limit theorem)