Estimating trajectories of vehicles towards intersections

Jacobs, L.C.P.

Award date:
2019

Estimating trajectories of vehicles towards intersections

Eindhoven University of Technology

Industrial & Applied Mathematics
Master Final Project

2MMR30

Author:
Luuk Jacobs, ID 0853299

Supervisors:
Dr. Ir. M.M.A. Boon
Ir. B. van der Bijl

March 14, 2019

Abstract

This research is about trajectories of vehicles that are driving towards are traffic light. The research is in collaboration with the company SWECO. Within SWECO there is a department that develops a product named "Smart Traffic". "Smart Traffic" wants to improve traffic flows by controlling traffic lights based on the real time situation on the road. For optimizing intersections it is very important to know where vehicles are located and when they will pass the traffic light. In this paper a model is developed to calculated the complete trajectory of a vehicle based on sensor data. This model is required to be fast, since the application should be used in real-time. The developed model is an event based model, which for each car only stores transition points between two different points. Between these transition points, the trajectory is described as curves, mostly as parabolas. These type of functions are fast, and intermediate points in the trajectory can be requested easily. Given sensor data of the road, the trajectories of all vehicles are calculated based on a predefined traffic light control system. Eventually the moment of passing the traffic light is calculated and compared with a simulation that is performed in the program SUMO. In every simulation 5000 seconds are simulated, and influences of several parameters have been checked. Also an extended model has been used. Trajectories will be corrected at a sensor points at the moment that the model differs too much from reality. Some results are that there are settings that the model gives good results, but other models give bad results. A main cause of this is that the model can not correct the trajectories at the moment that a vehicle passes a different green or orange period in the model than in SUMO. It is calculated that the vehicles that differ the most in passing time are caused by a car that left in the wrong period. Some of the parameters, like low acceleration and deceleration causes more of these problems. Differences in maximum speed do not seem to make a difference. The extended model does not work properly when vehicles drive to slow at the second sensor, in combination with a different number of vehicles waiting. Lastly, a different green and red period give reasonable results as long as the vehicles are not waiting on the second sensor. In the end "Smart Traffic" does not use cycle period times, and just want to know how long the traffic lights have to be green. In the basic model it is possible to calculate the moments of passing if at a random moment one sets the light to green. It is up to "Smart Traffic" to determine the green time, since optimization methods for green periods are not used in this paper.

Acknowledgements

I would like to thank the company SWECO, location De Bilt, for giving me the opportunity to do my final project at your company. Thank you for providing services to make sure that there was a good working environment. Especially I would like to thank the team Mobility Solutions for their interest and their willingness to help. Furthermore, I would like to thank Bas for making time during the office days and discussing the progress of the project, and the belief that the project will succeed.

Of course, I would like to thank Marko for the nice conversations at the Technical University of Eindhoven. Thank you for reserving an hour each week to discuss the progress of the project, and for motivating me at moments I thought that project was not going well.

Since I had to travel a lot: Many thanks, the company NS, for travelling without any delay during the past half year. I have spent 3 hours per day, for 2 a 3 days a week the last year, and that was something I was not looking forward to, but after all travelling happened successfully.

I would also like to thank all the people that were involved in my internships at BC Broekhin Roermond, and Trevianum Sittard. During these periods I really developed myself as a person, and it gave insight in what for type of job I would like to do after graduation.

I would like to thank my friends for the mental support during this period, and to discuss problems that apparently everyone has during the graduation period, which made me feel that I'm not the only one. It motivated me to keep going on. Also I'm grateful for the people I worked with the last 5,5 years at the TU, as well as the staff of the TU (Mathematics & ESOE). The people at the TU are always willing to help. From my fellow students I would especially like to thank Claudia. We also did the master Science, Education & Communication together, where we also finished our final project. During the entire period on the TU, we supported each other, and now we finish our final project in the same period, and hopefully graduate.

I would like to thank my family and friends for the supporting during the past half year. Special thanks to my mother, father and sister for their involvement and believe in me. I can discuss anything with them and they helped me when I had a bad day.

Contents

Chapter 1

Introduction

Over the last decades, the number of cars on the roads increased rapidly. More and more people take the car to travel to their work, or making trips. The number of highways, and the capacity of those highways increased enormously to be able to handle the number of cars on the road. Also cities change their road networks, since the old network does not meet the demand anymore. One of the bottlenecks in traffic are intersections that are being controlled by traffic lights. A disadvantage of those traffic lights is that there are always traffic flows that conflict with each other. This means that one can not prevent that there are cars that have to wait for the traffic lights. Since everyone would like to travel without any delay, long waiting times are frustrating and therefore not desirable. In the current situation on the road, controlling the traffic lights is mostly done by cycles. Which means that the flows are being served in the same order each cycle, with the same green period duration. This does not take into account roads that are more crowded than others, making waiting times even longer, which results in more frustration. Therefore one would like to minimize the waiting times of all vehicles, such that everyone can reach their destination on time. Another reason for optimizing waiting times, is that there would be less air pollution if cars are not standing still letting their engine on. Next to pollution, long waiting times costs also more money. Time is money is the proverb, and in this case true. If vehicles arrive to late at their destination, it can be that appointments have to be postponed, or that import shipments arrive too late. This results in economical dammage. All of these issues are good reasons to come up with a better alternative.

One company that came up with a better alternative is SWECO. To make the roads better, the company SWECO has developed a method to make intersections more efficient. This method is called "Smart Traffic". With this Smart Traffic, multiple sources, which are mainly sensors on the road and floating-car data, are used and combined to predict what will be the situation on the road at a certain point in the future. In that way, it is possible to predict the number of vehicles that are going to enter the intersection. Smart Traffic sees these cars coming and it can control the traffic lights in such way, that they can pass the intersection faster. Also vehicles that arrive in a plattoon can be served in such way that they can join the traffic flow that just passed the intersection. Also when the last car has passed the intersection, the traffic light can be set to red again, so that another flow can be served. The main difference that people may notice on the road is that green periods are not cycled anymore, and eventually the main goal should be that the overall traffic flows will be improved, and that everyone reaches their final destination on time (van der Bijl and Brouwer) (Smart Traffic).

One of the issues with Smart Traffic is that every vehicle should be in the system and must be followed. These vehicles are given by data. This data comes from the sensors in the roads, and from floating car data. Since the amount of data is also increasing, systems become more complicated and need a lot of computation power. Smart Traffic has its purpose to be used continuously and should therefore have computations that are light enough to ensure that the product can be used in real time. If the calculations are too slow, then the car already passed the intersections, before the Smart Traffic system controlled the traffic lights. In this research, the main goal is to develop a model, such that the trajectory of individual cars is predicted in an accurate way, but is fast enough in realtime. This will result in an algorithm that will calculates the entire trajectory of a vehicle based on sensor input. If needed the trajectory will be adapted based on new information on the road.

Chapter 2

Literature

In the literature there are multiple researches done about traffic models. In this chapter, several kinds of models are discussed. Thereafter, the most important models for this research is described in more detail.

2.1 Traffic models

For describing the traffic on the road there are multiple models in literature. There are three types of traffic models that are used most frequent. These types are based on different levels looking at the traffic. These levels are macroscopic traffic models, microscopic traffic models and mesoscopic travel models (Hoogendoorn (2001)).

2.1.1 Macroscopic models

This type of model only looks at the traffic in its entirety. This means that there is no information about every vehicle, but only in groups. It uses flows of traffic and is based on fluid dynamic models, and the mathematical formulation is therefore often in terms of partial differential equations. It takes into account volume, density and the average speed of vehicles. One can determine what is happening on a road section, but not the behaviour of the individuals vehicles. With these models, the total waiting time can be calculated, and the total distance vehicles have travelled. For this the model uses arrival rates that are estimated. This type of model is therefore not very accurate but can often be used to give insight in the traffic situation. This does not mean that the performance of this model is bad, but the disadvantage is that there is little information. An advantage is that it can be used more easily to describe larger networks, since the complexity is not high. Information that can lead to better performance. One of these macroscopic models is the model of Lighthill and Whitham (Lämmer and Helbing (2008)).

2.1.2 Microscopic models

In contrast to the macroscopic models, the microscopic models model individual vehicles or road users. The model takes into account the behaviour of every single vehicle by following the vehicle. These models are more detailed, but therefore also more complex. The information of every vehicle is known and when performing a simulation, and the information of every vehicle will be updated every time when a vehicle switches from state. The main information that these models use are distances, time and trajectories. This type of models can be used for discrete event simulation for single cars. But in contrast to the macroscopic model, the microscopic model can only be used for small networks. In section 2.3 a car-following model is presented that shows some properties of how a microscopic model looks like.

2.1.3 Mesoscopic models

The macroscopic model had the advantage of low complexity which can be used for a large network, but the disadvantage for the level of detail. For the microscopic level it was the other way round. Now the mesoscopic model is more balanced by the complexity and the level of detail. Mesoscopic models are often extensions of macroscopic levels, but from a perspective of microscopic models. It uses the individual vehicles from the microscopic models in the fluid-dynamic models from the macroscopic models. One of these models, which is going to be used further in the report, is a Fluid-Dynamic Model. This model is described in section 2.2 In this model there are no individual cars taken into account, but expressions for growth of waiting time can be found based on in and out flow of the road. Smart Traffic eventually wants to minimize these waiting times.

2.2 Fluid-Dynamic Model

The Fluid-Dynamic traffic model of Lighthill and Whitham is a model that is now used in Smart-Traffic. The model of Lighthill and Whitham is a macroscopic model. But in this approach, it is a little bit adapted and it becomes a mesoscopic model. In this model, the goal is to adapt traffic lights in future situations instead of only the current situation of traffic. This is a requirement for Smart-Traffic since it is more innovative and leads to better performance than when only the current traffic situation is taken into account in the model. Although the model that is going to be developed in this paper is not quite of this type it may be helpful to understand this type of model and which ideas the company use. Some of these ideas we keep in mind for the model in this paper. This part is mainly needed for the link from the model that will be developed, and Smart Traffic. The model is described by (Lämmer and Helbing (2008)) (Lammer), and in this section parts of the model are described in own words. Also a little bit of optimization is included in this section to show how Smart Traffic handles this model. The optimization part we do not use further in the paper.

2.2.1 In and out flow

To start with, this model uses road sections i, with fixed length L_i, where the speed limit is known. For each road section there exists a saturation level $Q_i^{\max}$ which gives a maximum for the number of vehicles that can arrive at that road section during a giving time period. So for the arrival rate we have $Q_i^{\text{arr}}(t) \leq Q_i^{\max}$ for any road section. The same holds for the departure rate of a road section, since there can not leave more cars than the saturation level. This results in $Q_i^{\text{dep}}(t) \leq Q_i^{\max}$.

The model comes with an expression for the number of vehicles that leave a certain road section. There are two cases that can occur on the road. The first case is that every car can just drive at the maximum speed without delay. In this case the number of vehicles that leave the road section during a given period is determined by the arrival rate, the road length and the speed limit. This number for free traffic for road section i can be calculated by:

$$N_i^{\text{dep}}(t) = \int_{-\infty}^{t} Q_i^{\text{arr}}\left(t' - \frac{L_i}{V_i}\right) \mathrm{d}t' = N_i^{\text{exp}}(t) \tag{2.1}$$

The term inside the integral of the above equation is the arrival rate with a correction for the time that is needed to drive the entire road with maximum speed. If a vehicle leaves the road at time t, then the car arrives at time $t - \frac{L_i}{V_i}$. However, the situation above is not the interesting part since there would be no problem te solve if every vehicle could just drive without delay. The other situation that is more interesting is the situation where there is delay. In this case $N_i^{\text{dep}} \neq N_i^{\text{exp}}$, since the integral above is used for maximum speed. This quantity can be calculated by:

$$N_i^{\text{dep}}(t) = \int_{-\infty}^{t} Q_i^{\text{dep}}\left(t'\right) \mathrm{d}t' \leq N_i^{\text{exp}}(t) \tag{2.2}$$

Now $N_i^{\text{exp}}(t)$ is the number of vehicles that should have left the road if there is no delay, and $N_i^{\text{dep}}(t)$ is the number of vehicles that actually do leave the road without delay. So the number of delayed vehicles can be calculated by:

$$n_i(t) = N_i^{\text{exp}}(t) - N_i^{\text{dep}}(t) \tag{2.3}$$

The vehicles that did not have delay are not waiting so they do not join a certain queue. But the vehicles that do have delay are waiting and can join a certain queue. This queue is not defined yet. But we already know the size of the queue, namely $n_i(t)$. So for each road section we can determine $n_i(t)$ so that we know on each road how many vehicles have delay. Eventually the model wants to reduce the waiting and the number of vehicles in the queues. At first the total waiting time of the entire queue has been described. Is is easier to describe the rate of which the waiting time grows. The total waiting time depends only on the number of vehicles in the queue, so the waiting time grow with $n_i(t)$. So:

$$\frac{\mathrm{d}w_i}{\mathrm{d}t} = n_i(t) \tag{2.4}$$

In the period that no vehicle joins the queue that already contains vehicles, the total waiting time of all vehicles increases linearly. When a vehicle enters the queue, the rate increases and therefore the total waiting time increases even more.

2.2.2 Influences of traffic light states

The next model assumption is that Kirchoff's Law applies on the network. For this law we need a node with links. Some links are incoming, and others outgoing. Kirchoff's Law suggests that the rate of an outgoing flow

is the sum of the incoming links with each a certain proportion. In mathematical symbols:

$$Q_j^{\text{arr}}(t) = \sum_i \alpha_{i,j}(t) Q_i^{\text{dep}}(t) \tag{2.5}$$

The conditions that need to be satisfied are $\alpha_{i,j}(t)$, with $\alpha_{i,j}(t) \geq 0$ and $\sum_j \alpha_{i,j}(t) = 1$.

The next thing to do is to develop a model for the switching of traffic lights. In this model only switching from red to green and vice versa are discussed. Therefore a variable $\gamma_i(t)$ is introduced. This variable equals 0 if the traffic light for road section i is red, and it equals 1 if this traffic light is green. Now we can describe equations of the queue length. There are 3 cases that we can distinguish. The first is a red light, the second is green light where the queue is not empty, and the third case is that the queue is empty. In the last case, there is no more delay for the traffic that is being served by this traffic light. These cases describe the change in the number of vehicles in the queue.

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = \begin{cases} Q_i^{\text{exp}}(t) & \text{if} \quad \gamma_i(t) = 0 \\ Q_i^{\text{exp}}(t) - Q_i^{\max} & \text{if} \quad \gamma_i(t) = 1 \quad \text{and} \quad n_i(t) > 0 \\ 0 & \text{if} \quad \gamma_i(t) = 1 \quad \text{and} \quad n_i(t) = 0 \end{cases} \tag{2.6}$$

If the traffic lights are red, like in situation 1, then the queue length raises with the arrival rate at the time that the vehicles enter the road section. The notation for this is $Q_i^{\text{exp}}(t)$, which is equal to $Q_i^{\text{arr}}(t - L_i/V_i)$. This relation we are going to use later on as suggestion to combine the model to be developed with Smart Traffic. Since there are no vehicles leaving in situation 1, the number of vehicles in the queue is increasing.
In situation 2 there is still a queue, but the vehicles in the queue are being served. Since cars are waiting the outflow is equal to the saturation flow $Q_i^{\max}$. But there are also new vehicles that arrive at the traffic light, so the queue is moving backwards. The arrival rate is therefore again $Q_i^{\text{exp}}(t)$, which is the inflow. So the number of vehicles in the queue is decreasing, or in the worst case does not change when the inflow is equal to the outflow. In the last case there is no queue, and therefore there are no vehicles waiting. This situation take place when there is an extended green period. Platoons can be served in this period in such way that they have no delay. Since there is no queue, the number of vehicles in the queue stays at 0.

2.2.3 Strategies to determine the length of green periods

One of the main points is to determine how long a traffic light should be green to have a good performance. In the time that a traffic light is green, the queues will decrease, but the time can also be extended such that vehicles farther away from the traffic light can be served. An effective case is stated in Figure 2.1.

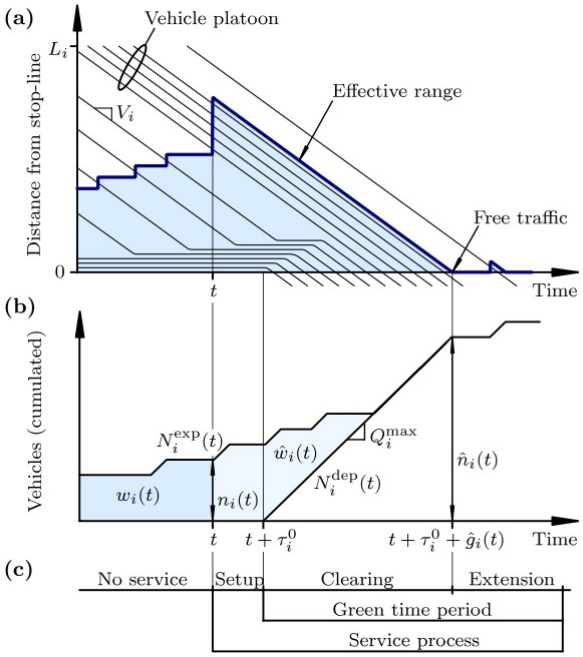


Figure 2.1: Traffic situation

In this figure we can see the movements of each single vehicle, related to the distance to the stop line. The dark blue line helps to determine how long the green time should be. Every vehicle in the light blue area will eventually be served. This means that every time a vehicle enters the blue area, the graph will make a jump, since the traffic light switches to green if this does not happen. At the time t, there is first a set-up time, and this is exactly the time such that all cars have been served that were waiting in the queue, but also the green wave that is coming after the vehicles already waiting. This way the green time is used at maximal flow, and are there no gaps. The time to clear the queue and to serve the green wave is denoted by $\hat{g}_i(t)$ At time t there will be determined which part of the network will be served, based on the waiting times. This does not mean that after time t there is no more delay. The vehicles move one by one, so this means that car 2 has to wait until car 1 is away. The waiting time of the vehicles in the queue is therefore denoted by $\hat{w}_i(t)$. We already used $n_i(t)$ for the number of delayed vehicles. Another interesting quantity is the expected number of vehicles to be served during the clearing state $\hat{n}_i(t)$. This number is all vehicles in the queue combined with the vehicles served in the green wave.

To determine $\hat{g}_i(t)$, we need in fact the intersection point of Figure 2.1b. But there are many intersection points since at the last part two lines overlap. But if the green time ended at the first intersection point, a large group of cars has to wait, and then unneeded delay occurs. So the best choice is to choose the largest intersection point, because a platoon is served. To find the best option we have to solve equation.

$$N_i^{\text{dep}}(t) + \hat{g}_i(t) Q_i^{\max} = N_i^{\text{exp}}(t + \tau_i(t) + \hat{g}_i(t)) \tag{2.7}$$

In words it says that the number of vehicles that depart from the existing queue plus the number of vehicles that arrive during the clearing time equals the number of vehicles that do departure if there was free traffic during the entire period and therefore reached the point departure. Just like the waiting time $w_i(t)$, also the future waiting time can increase. This increase is given by:

$$\frac{\mathrm{d}\hat{w}_i}{\mathrm{d}t} = \begin{cases} \hat{n}_i(t) & \text{if flow i is not being served} \\ 0 & \text{if flow i is being served} \end{cases} \tag{2.8}$$

But there is a problem when the serving has ended earlier, and if there are vehicles still waiting in the queue. Also this increase can be calculated. This expression is:

$$\Delta\hat{w}_i(t) = Q_i^{\max} \int_{\tau_i(t)}^{\tau_i^0} \hat{g}_i(t, \tau') \mathrm{d}\tau' \tag{2.9}$$

2.2.4 Optimizing green periods for multiple directions

Further in the paper that described the model, a strategy for optimizing the traffic lights is stated based on priorities. Each traffic flow i gets a priority index $\pi_i(t)$, and the flow with the highest priority will be served. This will be determined by $\sigma(t)$, which gives the index of the flow with the highest priority.

$$\sigma(t) = \arg\max_i \pi_i(t) \tag{2.10}$$

With this σ, a rule for switching the traffic lights is determined so that one can decide what traffic flow has the highest priority. These priorities depend on the waiting times of the flows. An investigation is done for only two traffic flows, in the hope it can be generalized to n flows. If a certain traffic light is chosen there are only two options that can occur. One has to decide whether to continue a green period, or switch to red such that another flow can be served. Choosing both options results in increasing waiting times for all vehicles in one or more flows. For both options the increase of the total waiting time is being calculated. For this we need a certain time t, and for both flows a remaining time to switch to green, τ_1 and τ_2. Suppose that $\sigma = 1$. If the choice is made to stay serving flow 1, then the service time for flow 1 is equal to $\tau_1 + \hat{g}_1$. For flow 1, there obviously is no increase of waiting time when these vehicles are being served. On the other hand the vehicles of flow 2 are waiting. In equation 2.8 is stated that the waiting time increases at the rate of the number of vehicles that could be served at that point, which in this case is equal to $\hat{n}_2(t)$. So the future waiting time for the vehicles standing in the queue at that moment is equal to $(\tau_1 + \hat{g}_1)\hat{n}_2$. The other option is to choose flow 2 to being served. This means that flow 2 has no increase in waiting time. For flow 1 we have an extra waiting time caused by the fact that the service period of flow 1 still has to end. The amount of waiting time is calculated by 2.9 and is thus $\Delta\hat{w}_i(t)$. Just like in option 1, the waiting time grows with the number of vehicles in the queue. So this is given by $(\tau_2 + \hat{g}_2)\hat{n}_1$. The total increase of the waiting time is therefore given by $\Delta\hat{w}_i + (\tau_2 + \hat{g}_2)\hat{n}_1$.

The two quantities for the increase of waiting time, decide which of the two options should be executed.

The option with the smallest waiting time should be chosen. So to continue serving flow 1, one should have that

$$(\tau_1 + \hat{g_1})\hat{n_2} < \Delta\hat{w_i} + (\tau_2 + \hat{g_2})\hat{n_1} \tag{2.11}$$

If this unequality is not satisfied, then option 2 is chosen and flow 2 is being served. If we rewrite this equation we get the following:

$$\frac{\hat{n_1}}{\tau_1 + \hat{g_1}} > \frac{\hat{n_2}}{\frac{\Delta\hat{w_i}}{\hat{n_1}} + \tau_2 + \hat{g_2}} \tag{2.12}$$

The priorities are now stated in equation 2.12. So $\pi_1 = \dfrac{\hat{n_1}}{\tau_1 + \hat{g_1}}$ and $\pi_2 = \dfrac{\hat{n_2}}{\frac{\Delta\hat{w_i}}{\hat{n_1}} + \tau_2 + \hat{g_2}}$.
This rule can also be extended for more than two traffic flows, but that part we will not discuss anymore. This gives an indication what the method is for giving one flow priority over another.

2.3 Car Following Model

Car following models are used to make sure that cars do not collide with the leader car. This means that there always should be a minimum gap between the cars. In such a model the positions of the car are calculated in a way that the speed of the car approaches the maximum speed of that car, but also that it has a safe distance on the leader car. So the gap between two cars can be stated. If x_l is the position on the road of the leader car, x_f the position of the follower, and l the length of vehicles (assuming the vehicles have equal length). Then the gap g between two cars is:

$$g = x_l - x_f - l \tag{2.13}$$

The gap between cars should be a minimum distance, but that depends on the speed of the cars. So next to the desired gap g_{des}, there is also a desired time that has to be taken into account, namely τ_{des}. So the gap should be larger then a certain combination of g_{des} and τ_{des}. So according to (Krauss (1998)) the inequality is:

$$\frac{dg}{dt} \geq \frac{g_{\text{des}} - g}{\tau_{\text{des}}} \tag{2.14}$$

A condition for cars to drive at a safe distance is given by

$$v_l - v_f \geq \frac{v_l\tau - g}{\frac{\bar{v}}{b(\bar{v})} + \tau} \tag{2.15}$$

In this condition, $\bar{v} = \dfrac{v_l + v_f}{2}$, $b(\bar{v})$ is the decelerate function, and τ is the time gap. In this case, if the deceleration is 6 $\frac{m}{s^2}$, then $b(\bar{v}) = 6$. So $g_{\text{des}} = v_l\tau$, and $\tau_{des} = \frac{\bar{v}}{b(\bar{v})} + \tau$. As a result they state that:

$$\frac{dg}{dt} = v_l - v_f \tag{2.16}$$

The above model is a continuous model. If one wants to update the speed at a certain time step in the future, a discrete formulation of the model is needed. We want to know what speed the follower would have at time $t + \Delta t$. By using equations 2.14 and 2.16, it results in:

$$v_f(t + \Delta t) \leq v_l(t) + \frac{g(t) - g_{\text{des}}(t)}{\tau_{\text{des}}} \tag{2.17}$$

Also the positions can be updated to time step $t + \Delta t$. This update will be:

$$x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t \tag{2.18}$$

Since the model is based on safe gaps, the gap should also be updated. The update of the gap is:

$$g(t + \Delta t) = g(t) + \Delta t(v_l(t + \Delta t) - v_f(t + \Delta t)) \tag{2.19}$$

Combining this with inequality 2.17 gives:

$$g(t + \Delta t) - v_l(t + \Delta t)\Delta t \geq (g(t) - v_l(t)\Delta t)\left(1 - \frac{\Delta t}{\tau_{\text{des}}}\right) + \Delta t\frac{g_{\text{des}}(t) - v_l(t)\Delta t}{\tau_{\text{des}}} \tag{2.20}$$

Krauss (1998) concludes that safety is guaranteed if the following holds:

- $g(0) - v_l(0)\Delta t \geq 0$
- $\Delta t \leq \tau$
- $g_{\text{des}} \geq v_l \Delta t$

With these equations, one can formulate the model. It is given by the following equations:

- $v_{\text{safe}}(t) = v_l(t) + \frac{g(t) - g_{\text{des}}(t)}{\tau_{\text{des}}}$
- $v_{\text{des}}(t) = \min(v_{\max}, v(t) + a(v)\Delta t, v_{\text{safe}}(t))$
- $v(t + \Delta t) = \max(0, v_{\text{des}}(t) - \eta)$
- $x(t + \Delta t) = x(t) + v\Delta t$

This η is a random quantity that slows the vehicle down with a random amount. This number is randomly distributed between 0 and ϵa, with $\epsilon \in (0, 1)$, and a the maximum acceleration.

In the model described in this section, the model does the updates for each vehicle. So for each time step, all vehicles in the system get a new position, a new $v_{\text{des}}, v_{\text{safe}}, v$ and x. If there are many cars in the system, and one wants to do this each second, then it is never going to be real time. For simulation it is an accurate way to determine the behaviour of all cars and to calculate trajectories. But since it is to expensive to calculate, there is need to make a car-following model, that may be a little bit less precise, but is still very accurate and less expensive.

2.4 Model parameters

For the model that will be developed there will be some parameters that have to be included. One of these parameters is the headway between vehicles. In 2.3 we discussed a car-following model that included safety gaps. For the idea of the value of this gap it is good to look at a practical research. SWOV (2012) wrote a paper that discuss the time gap there should be between vehicles and why this is the norm. It includes reaction time, and also for an emergency stop it is needed to have that distance. They suggest a distance from 2 seconds, while in practice it occurs that the distance is sometimes less than 1 second at high speed. In our model, if is going to be used, we have to make sure that the calculations are based on safety and therefor we will maintain a time gap of 2 seconds.

For the deceleration, Saptoadi (2017) investigated that the deceleration rate differs from vehicle type. Deceleration rates of cars can go up to 7 m/s^2. It may be good for the model to know when a vehicle can still stop before the traffic lights. The paper recommends 3.5 m/s^2 , but in the model we could choose any value for this.

The acceleration of cars can go up to 3 m/s^2, when the starting speed is very low. For other vehicle types this rate is different (Bokare and Maurya (2017)). The most important acceleration rate is the one that occurs when driving from rest. If the speed is higher, then the acceleration decreases linearly. Also this parameter in the model we can choose by ourselves.

Chapter 3

Models

In chapter 2, a car-following model is described, for which in free traffic flow at each time step, all vehicles get an update. New positions and speed are determined for them all. Now we want to have a model that calculates the trajectory of the vehicles, but only at certain time points. In the car-following model, many information is not needed to optimize the traffic flows and many calculations are unnecessary. The goal of the model is, using input data, calculate the entire trajectory until it departs the intersection at once time, using the trajectory of the vehicle it is following.

3.1 Situation description

In this case we use a single intersection, with four roads. Each road has one flow in the direction of the intersection, and the other in the opposite direction. All those road segments will be numbered with a number i. Each lane has road length l_i and has speed limit v_i. The intersection is controlled by traffic lights. These traffic lights operate with a fixed cycle the consists of a green period, a yellow period and a red period. The duration of the green, yellow and red period of lane i, is respectively denoted by g_i, y_i and r_i. For each lane at this intersection the following should hold: $g_i + y_i + r_i = c$, with c the total time of one complete cycle at the intersection. A visualisation of the situation is given in Figure 3.1. Note that Smart Traffic does not want to use cycle periods. But for calculating trajectories it does not make a difference to use cycle periods or variable periods since both control systems involve green, yellow and red periods. The trajectories that we will going to model will occur in both situations.

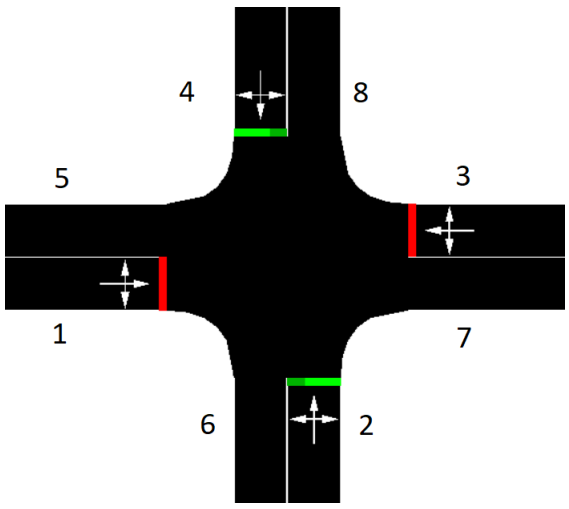


Figure 3.1: Example of an intersection that is used for the model

For this scenario we only observe traffic flows that go straight over the intersection. The reason for this is that cars who turn left or right influence the behaviour of accelerating and decelerating at the intersection. For this model it is easier to look first at a situation that all vehicles drive the same route. So cars can go from lane 1 to lane 7, from lane 2 to lane 8, from lane 3 to lane 5, or from lane 4 to lane 6. For this situation it was also fine when there is no intersection at all, but just let the cars stop when a traffic light is red. Eventually, in this situation we have enough at just one flow, say for example lane 1 to lane 7. In practice, there are sensors placed on the lanes to track the cars. These sensors give a signal when cars are coming in the direction of

the intersection, and in Smart Traffic, the traffic light must turn green in order to let the vehicle pass. These sensors will also be used in our model. A lane could have multiple sensors. The position of the j-th sensor at lane i, we denote with $p_{i,j}$. Here $p_{i,1} < p_{i,2} < ... < p_{i,nrsensors}$. The information we get from the sensors is all the information we can use in our model. Of course we could also produce floating car data, but this type of data gives to many information. The goal was to give a good prediction with as little as possible data. For the first model we only use data of one sensor that is far away of the intersection. In this way, the vehicles drive in the so called free traffic form if the road is not saturated. This sensor can give the time that the vehicle enters, and the time that the vehicle leaves the sensor. A simulation program can also determine the speed on that sensor. In reality this speed has to be estimated. For the model we assume that the speed is known. In case it is not known, we use the speed limit for that lane. In the rest of the paper, we only model the traffic flow from lane 1 to lane 7.

3.2 Model assumptions

- At the point that the vehicle reaches the sensor, it drives in free traffic.
- Vehicles do not pass each other and drive in the same order after each other, even when they passed the intersection.
- All cars are identical in type, such as length, maximum speed, acceleration and deceleration.
- Each car has a minimum distance to the car that drives in front of it.
- There is also a minimum time distance between cars, since drivers need a reaction time.
- Vehicles do never drive through the red light.
- The vehicles have the same maximum acceleration and deceleration.
- If possible, the vehicles accelerate and decelerate with their maximum rate.
- The trajectory of a vehicle only depends on the trajectory of the predecessor.
- If the speed is known at the start sensor, then this speed will be set to the maximum speed for that vehicle.

The vehicles should drive in free traffic at the beginning in such way that there is no interaction between cars at the start, and that a reaction can only start in the system. The advantage is that there is no dependency at the start which could have an influence on the rest of the trajectories. Also cars are not passing each other since this is practically caused by human behaviour when cars are driving too slow. Since in the model the vehicles are of the same type this will not happen and is therefore a reasonable assumption. Driving safe is one of the major points in traffic. This includes driving at a safe distance from the leader. This quantity will be in combination with the car length. Also a safe time distance is important since drivers need to react on actions of the leader. In this model every vehicle has to drive at least a safe time distance away from their leader. Obviously, passing red lights is illegal in practice and is therefore not allowed in the model. Passing the orange light can occur but we mostly try to avoid that. The orange light has for each driver another interpretation and is harder to model. Further we choose to let the vehicle use mostly their maximum acceleration and deceleration since then most parts of the trajectory are modelled by constant speed. Constant speed is for the following cars easier to react on than acceleration and deceleration. So the model is made less complex this way. The assumption that the trajectory is calculated by only the trajectory of the direct leader is reasonable since the leader trajectory has also been influenced by their leader. In practice one tends to look at the entire flow instead of one car. But for the model it is not necessary since all trajectories are already been adapted by the front of this flow. The maximum speed of the vehicles of all vehicles is set to the same value. However, we will use a simulation program that gives us the speed at the sensor, which can differ from the maximum speed. Since we want to include the true speed, we will use the speed at the sensor as the maximum speed. Vehicles will never drive faster than the maximum speed.

3.3 Trajectory of the first vehicle

In the model assumptions it says that a trajectory only depends on the trajectory of the predecessor. Since the first vehicle does not have any predecessor, we can make rules for this car in such way that no other cars are involved. The trajectory only depends on the state of the traffic lights. We denote the position of vehicle i at time t on lane 1 by $p(t, i)$, and the speed by $v(t, i)$. Suppose that the sensor detects at t_1 a vehicle, and

the intersection has position p_2, then we have to calculate at which time t_2 the car passes the intersection. Assuming a constant speed $v(t_1, 1)$ we can calculate this t_2, namely

$$t_2 = \frac{p_2 - p(t_1, 1)}{v(t_1, 1)} + t_1 \tag{3.1}$$

Now, multiple situations can happen in the interval $[t_1, t_2]$. The first situation is that in the entire interval, the traffic light is green. This is the easiest case, since the driver sees green and does not have to think about decelerate. Therefore at time t_2, and position p_2, the car leaves the system with equal speed as when it entered the system.

A second situation is that at t_2 the traffic lights are red. One of the assumptions was that the vehicles never pass a red light. So there must be a moment that the car starts decelerating. But this decelerating needs a certain amount of distance, that depends on the speed of the vehicle. If we assume that the vehicles decelerate with rate $d_{\max}$, then we calculate the moment t_{brake}, which is the latest moment the vehicle can start braking, and can stop at position p_{stop}. So the trajectory becomes a constant speed with function s_1, and a parabola with s_2. For calculating the trajectory we have to use the following functions:

Function name: **stoppingtime**
Known variables: $t_1, p(t_1, 1), v(t_1, 1), p_{\text{stop}}, d_{\max}$.
To solve: $t_{\text{brake}}, p_{\text{brake}}, t_{\text{stop}}$

$$\begin{aligned} s_1(t) &= p(t_1, 1) + (t - t_1)v(t_1, 1) \\ s_2(t, t_{\text{brake}}, p_{\text{brake}}) &= \frac{1}{2} d_{\max}(t - t_{\text{brake}})^2 + v(t_1, 1)(t - t_{\text{brake}}) + p_{\text{brake}} \end{aligned} \tag{3.2}$$

These functions have to intersect, but also have the same tangent at the point of intersection. Secondly, the maximum of s_2 has to be p_{stop}. This is already included in the second equation. So from these two equations we want to calculate $t_{\text{brake}}, p_{\text{brake}}$ and t_{stop}. This results in the following set of equations:

Equations of **stoppingtime**

$$\begin{aligned} s_1(t_{\text{brake}}) &= s_2(t_{\text{brake}}, t_{\text{brake}}, p_{\text{brake}}) \\ s_2(t_{\text{stop}}, t_{\text{brake}}, p_{\text{brake}}) &= p_{\text{stop}} \\ \left.\frac{\partial s_2}{\partial t}\right|_{t=t_{\text{stop}}} &= 0 \end{aligned} \tag{3.3}$$

So given an arrival at time t_1, position $p(t_1, 1)$, and speed $v(t_1, 1)$, combined with the fact that the vehicle has to stop at position p_{stop}. We obtain the following results for $t_{\text{brake}}, p_{\text{brake}}$ and t_{stop}.

Solutions of **stoppingtime**

$$\begin{aligned} t_{\text{brake}} &= \frac{2d_{\max}(p_{\text{stop}} - p(t_1, 1) + t_1 v(t_1, 1)) + v(t_1, 1)^2}{2d_{\max} v(t_1, 1)} \\ p_{\text{brake}} &= p_{\text{stop}} + \frac{v(t_1, 1)^2}{d_{\max}} \\ t_{\text{stop}} &= \frac{2d_{\max}(p_{\text{stop}} - p(t_1, 1) + t_1 v(t_1, 1)) - v(t_1, 1)^2}{2d_{\max} v(t_1, 1)} \end{aligned} \tag{3.4}$$

In Figure 3.2 the situation is visually explained. Suppose that a car starts at time step 0 at position 100 with speed 13 m/s. This vehicle will drive at the same constant speed at the entire trajectory. Since this vehicle is the first vehicle, the position where it will stop p_{stop} is known. With the function stoppingtime, the time t_{brake} will be calculated as well as the position p_{brake}. This is the last moment that the car can brake before the intersection. If at that moment the traffic lights are green than it will drive at the same constant speed. If it is orange or red, then the car will start brake at a constant deceleration rate. In this example the dashed line is the line of the trajectory that the car would drive if it does not brake. In this case it would drive through

orange light. In reality this situation would be allowed, but in our model it will not happen since the decision is already made at the start of the braking.

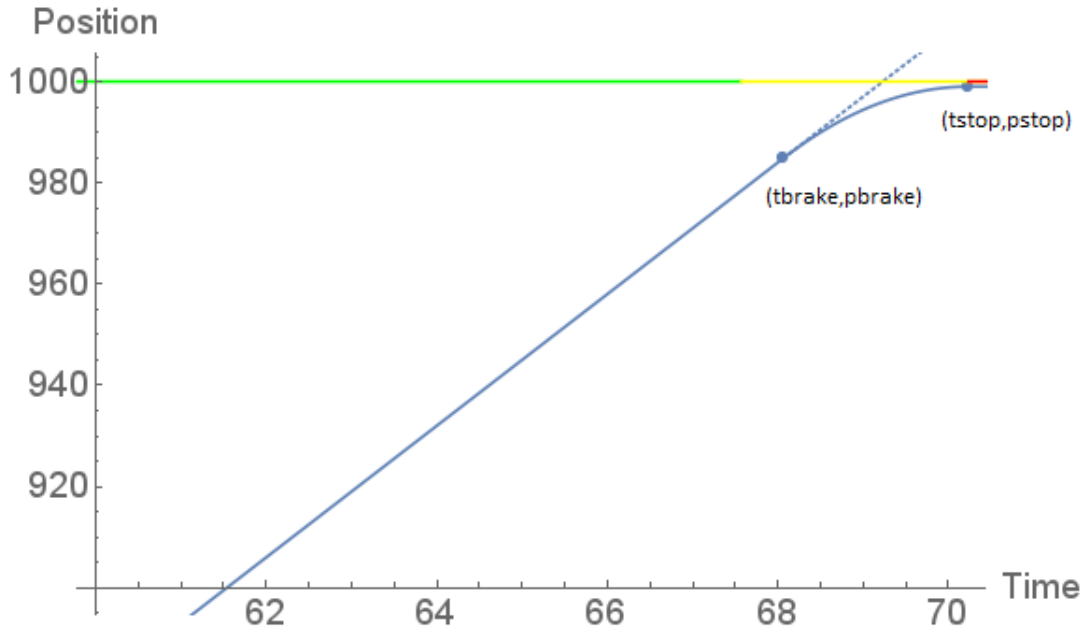


Figure 3.2: Visualisation of the stoppingtime function

So the trajectory of the first vehicle in the case that it must stop before the traffic lights will be the following.

Time	**Position**	**Speed**	**Type moving**
t_1	$p(t_1, 1)$	$v(t_1, 1)$	CONSTANT
t_{brake}	$p(t_{\text{brake}}, 1)$	$v(t_{\text{brake}}, 1)$	DECELERATE
t_{stop}	$p(t_{\text{stop}}, 1)$	$v(t_{\text{stop}}, 1)$	WAITING

From time t_{stop}, the car is waiting for the traffic lights to turn green. When the light actually becomes green at $t_{\text{nextgreen}}$, the cars need a certain reaction time τ_1 until it starts accelerating. So the accelerating part starts at time $t_{\text{nextgreen}} + \tau_1$. And it stops until the cars reaches the maximum speed. Since there are no cars in front of this car, it is possible for the car to accelerate at once to the maximum speed. Each car has its own maximum speed. Since we only use one data point, the starting point, we assume that the maximum speed for a car is $v(t_1, 1)$. This acceleration process we assume goes with constant acceleration with rate $a_{\max}$. The time that the car will be back to the maximum speed, and the position where this happen, can easily be calculated by:

$$\begin{aligned} t_{\text{maxspeed}} &= t_{\text{nextgreen}} + \tau_1 + \frac{v(t_1, 1)}{a_{\max}} \\ p_{\text{maxspeed}} &= p(t_{\text{stop}}, 1) + \frac{1}{2} a_{\max} (t_{\text{maxspeed}} - t_{\text{nextgreen}} - \tau_1)^2 \end{aligned} \tag{3.5}$$

So after the lights turn green we have the following trajectory:

Time	**Position**	**Speed**	**Type moving**
$t_{\text{nextgreen}} + \tau_1$	$p(t_{\text{stop}}, 1)$	0	ACCELERATE
t_{maxspeed}	p_{maxspeed}	$v(t_1, 1)$	CONSTANT

The trajectory is given 3.3. At the right of this picture, the end point is the point where the vehicle will drive at a constant speed again, which is equal to the starting speed. Note that the reaction time in this example is set to 0.

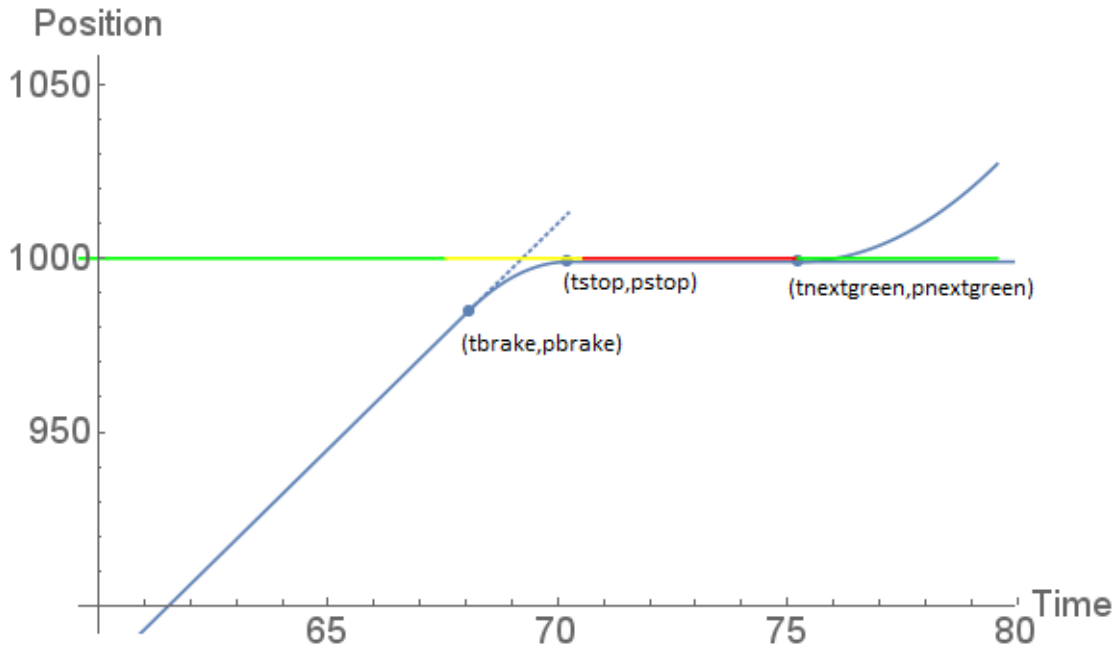


Figure 3.3: Total trajectory including one stop

The above trajectory is the trajectory that the car could drive when the lights turn to red. There is still another situation that can occur. This case is that at the latest moment of braking the lights are red, but before the car has stopped, the light turns green again. The logical thing to do for the driver is to go back to the maximum speed as soon as possible. In this case again a reaction time is included before it actually starts accelerating. Suppose there is a $t_{\text{green}} \in [t_{\text{brake}}, t_{\text{stop}}]$, which is the time that the traffic light turns to green, then the trajectory will be as follows:

Time	**Position**	**Speed**	**Type moving**
t_1	$p(t_1, 1)$	$v(t_1, 1)$	CONSTANT
t_{brake}	$p(t_{\text{brake}}, 1)$	$v(t_{\text{brake}}, 1)$	DECELERATE
$t_{\text{green}} + \tau_1$	$p(t_{\text{green}} + \tau_1, 1)$	$v(t_{\text{green}} + \tau_1, 1)$	ACCELERATE
t_{maxspeed}	$p(t_{\text{maxspeed}}, 1)$	v_1	CONSTANT

The quantities of the last two rows of the trajectory differ from the previous trajectory, the new quantities can be calculated by:

$$\begin{aligned}
p(t_{\text{green}} + \tau_1, 1) =& \frac{1}{2} d_{\max}(t_{\text{green}} + \tau_1 - t_{\text{brake}})^2 + v(t_1, 1)(t_{\text{green}} + \tau_1 - t_{\text{brake}}) + p_{\text{brake}} \\
v(t_{\text{green}} + \tau_1, 1) =& v(t_{\text{green}} + \tau_1, 1) + d_{\max}(t_{\text{green}} + \tau_1 - t_{\text{brake}}) \\
t_{\text{maxspeed}} =& t_{\text{green}} + \tau_1 + \frac{(v(t_1, 1) - v(t_{\text{green}} + \tau_1, 1))}{a_{\max}} \\
p(t_{\text{maxspeed}}, 1) =& \frac{1}{2} a_{\max}(t_{\text{maxspeed}} - t_{\text{green}} - \tau_1)^2 + v(t_{\text{green}} + \tau_1, 1)(t_{\text{maxspeed}} - t_{\text{green}} - \tau_1) \\
& + p(t_{\text{green}} + \tau_1, 1)
\end{aligned} \tag{3.6}$$

The situation is illustrated in 3.4. The position t_{brake} is the last time deceleration is possible, which must happen since the traffic lights are red. When the lights turn to green the cars accelerating again. Note that again the reaction time is set to 0 in this example. Otherwise the deceleration process would continue for τ seconds.

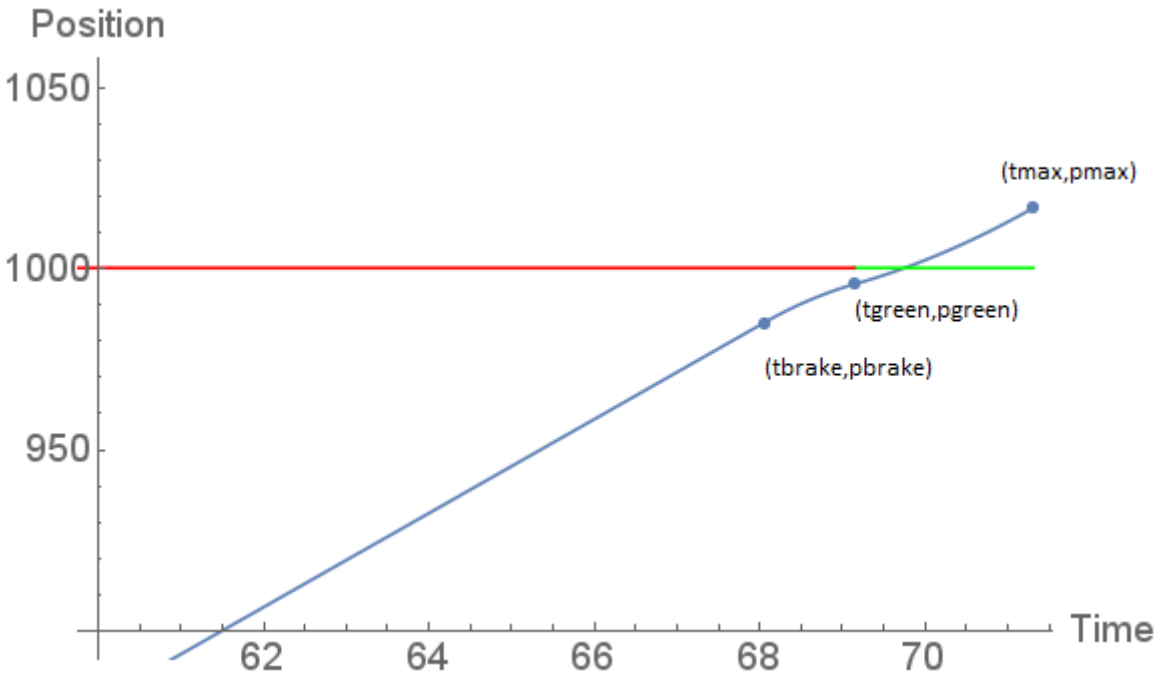


Figure 3.4: Stopping time from red to green

3.4 Trajectory of the following vehicles

The greatest advantage of the previous section was that there were no vehicles that have to take their predecessor into account. There was no risk of colliding. Only the traffic lights influenced the trajectory of the vehicle. For the following vehicles, it is necessary to know if the trajectories of the two vehicles cross each other, and the place that will happen. For this situation we do not track the cars any more until they accelerate to their maximum speed after the intersection. With crossing each other, we mean that the vehicles become closer to each other than a safe distance gap, g_{safe}, or time gap t_{safe}. The time gap is something that the entire model uses to make sure that the cars can drive safe. The gap in distance we will mainly use to determine the positions when cars are waiting behind each other, or to locate points which are harder to find by calculations. When cars are driving without many changes of speed, the model uses mostly time difference only.

The first problem can occur when two cars enter the sensor within a certain period that is shorter then the time gap. If this happens, then it instantly violates the assumption of the time gap and that there is no dependency at the start. But in a simulation this situation can happen. One way to solve this is to postpone the arrival of the second car, such that the time gap becomes t_{safe}. Since the speed is also an issue if the cars driving too close at each other, this should also be corrected. So if k is the leading car for car l, then if $(t_{1,l} - t_{1,k}) < t_{\text{safe}}$, then we change $t_{1,l}$ by $t_{1,k} + t_{\text{safe}}$. Also in this case, if the speed of vehicle l is higher than vehicle k, then vehicle l needs to decelerate. But when the time distance at the start is already higher than the safety gap, then vehicle l does not have the ability to decelerate to a safe distance. So in the the case that the time distance is exactly equal to t_{safe}, then vehicle l gets the same starting speed as vehicle k when the detector detects for vehicle l a higher speed than for vehicle k. This problem can also happen if the arrival times are more than t_{safe} in difference. How to determine that this problem is occurring will be explained later.

The first thing to do is a check whether vehicle l will have to react on vehicle k. A method to do this is to check whether any deceleration of vehicle l takes places before or after the intersection. When a vehicle passes the intersection, it first accelerates if needed, and then drives with the constant speed which is also detected at the sensor where the vehicle arrived. We distinguish three cases:

1. $v_l > v_k$
2. $v_l = v_k$
3. $v_l < v_k$

The first case is that the maximum speed of vehicle l is higher than the maximum speed of k. So the leading vehicle drives slower. What we would like to know is the point that vehicle l starts decelerating so that the

vehicles will not collide. For that we make use of a point $(t_m, p_m) = (t_{\text{maxspeed}}, p_{\text{maxspeed}})$, where the leading vehicle is driving at his maximum speed. With Equations 3.7 the point (t_3, p_3) can be calculated where the follower starts decelerating with deceleration d. The point (t_4, p_4) is the point where the speed of the follower is equal to the speed of the leader, and where the following vehicle has a time distance of t_{safe}. So s_1 describes the leading vehicle, and s_2, s_4 and s_3 respectively the following vehicle.

Function name: **DecreaseMoment**
Known variables: $t_1, t_m, t_{\text{safe}}, d, v_k, v_l, p_1, p_m$.
To solve: t_3, p_3, t_4, p_4

$$\begin{aligned} s_1(t) &= p_m + (t - t_m)v_k \\ s_2(t) &= p_1 + (t - t_1)v_l \\ s_3(t) &= p_m + (t - t_m - t_{\text{safe}})v_k \\ s_4(t, t_3, p_3) &= p_3 + (t - t_3)v_l + \frac{1}{2}d(t - t_3)^2 \end{aligned} \tag{3.7}$$

A situation can be found in Figure 3.5. Note that the point (t_m, p_m) can be put anywhere on s_1, as long as the point contains the maximum speed of the leader. In this case the traffic lights are at position 1000, and the time of the start of the brake is t_3 which is before the intersection. So the follower can not be stated as a new first vehicle.

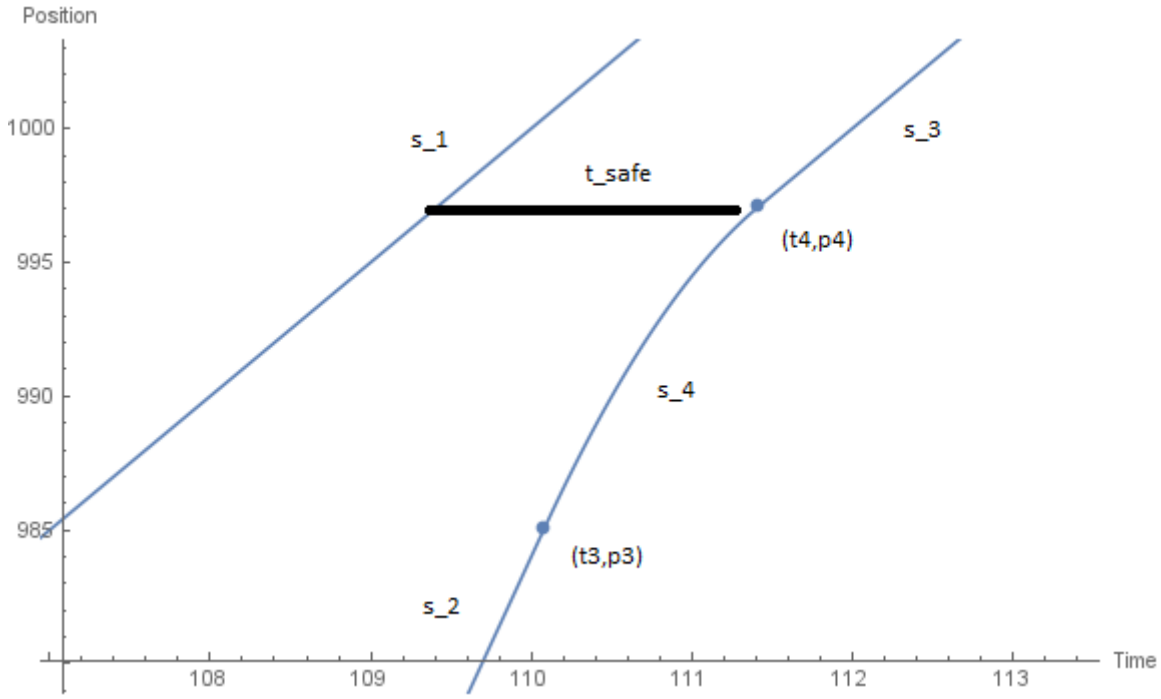


Figure 3.5: Trajectory to determine where the follower should brake to drive at a safe distance

In Equations 3.8 the equations are stated which should give the exact points for the trajectory showed above.

Equations of **DecreaseMoment**

$$\begin{aligned} s_2(t_3) &= s_4(t_3, t_3, p_3) \\ s_3(t_4) &= s_4(t_4, t_3, p_3) \\ s_3(t_4) &= p_4 \\ \left.\frac{\partial s_4}{\partial t}\right|_{t=t_4} &= s_3'(t_4) \end{aligned} \tag{3.8}$$

Lastly, in Equations 3.9 the solutions of this system are stated, which gives:

Solutions of **DecreaseMoment**

$$
\begin{aligned}
t_3 &= \frac{2d(p_m - p_1 - (t_m + t_{\text{safe}})v_k + t_1 v_l) + (v_k - v_l)^2}{2d(v_l - v_k)} \\
t_4 &= \frac{2d(p_1 - p_m + (t_m + t_{\text{safe}})v_k - t_1 v_l) + (v_k - v_l)^2}{2d(v_k - v_l)} \\
p_3 &= \frac{2d(p_m v_l - p_1 v_k + (t_1 - t_m - t_{\text{safe}})v_k v_l) + v_l(v_k - v_l)^2}{2d(v_l - v_k)} \\
p_4 &= \frac{2d(p_1 v_k - p_m v_l + (t_m - t_1 + t_{\text{safe}})v_k v_l) + v_k(v_k - v_l)^2}{2d(v_k - v_l)}
\end{aligned}
\tag{3.9}
$$

So for the following vehicle we get the following trajectory $s(t)$:

$$
s(t) = \begin{cases} s_2(t), & \text{if } t \in [t_1, t_3] \\ s_4(t, t_3, p_3), & \text{if } t \in [t_3, t_4] \\ s_3(t), & \text{if } t \in [t_4, \infty) \end{cases}
\tag{3.10}
$$

Since in our assumptions we stated that a trajectory can be calculated by only its predecessor, it is of importance to know what the point of decelerating is. If this point of breaking is after the intersection, then we can treat the follower as a first vehicle, since interaction does not take place before the intersection. This makes calculating the trajectory much easier, and we can use the rules in section 3.3. One of the main goals was to have a model that is not too complex, and these calculations are rather easy to compute. If the occupancy of the road is not too high, than it may be a good idea to do this check. However, if it is too crowded, than this step can be skipped, since it is obvious that a vehicle has too react on its leader. Another advantage of this method is that it can be used for different types of vehicles. By this we mean that there is a difference in maximum speed for each type. So now can be checked when a vehicle has to react on a tractor for example. Although the model will consider one type of vehicles, this part can always be used.

The second case is the case that the maximum speeds after the intersection are equal. So if (t_m, p_m) is a point where the leader drives his maximum constant speed v_l after the intersection, and (t_1, p_1) is the starting point of the follower, with the same speed, then the only thing to do is to check whether straight lines have a time distance of t_{safe} or more. Therefore the time t_2 that the follower arrives at p_m is $t_2 = \frac{p_m - p_1}{v_l}$. Then we find that the follower can be seen as a first car if $t_2 - t_m \geq t_{\text{safe}}$.

The third and last case is that the leader car has a higher speed than the follower. This is no problem when the leader has no delay before it arrives the traffic lights. But if the leader has to wait a certain time, then the follower might decelerate before the intersection or even has to stop. If the leader has no time period, where it was accelerating, then this means that this vehicle does not stop at the entire trajectory. It has a higher speed at the start already than the follower. This means obviously that there is no collision and the follower can be seen as a new first car which is just driving slower than his leader. If there are more parts of the trajectory that the vehicles are accelerating, it is enough to check the last part. If this part has a collision, then the follower has to react on the predecessor and is therefore not a first car. This case is different in calculation from the case of the follower with a higher speed, since it is wrong to just draw a straight line from the end to the beginning. In the case of a lower speed, it would give a collision at parts where there might not be collisions. This error occurs when the vehicle has to make a stop before the intersection. So the solution for this is to calculate the point where the leader is driving at the same speed as the follower v_2. This calculation is very easy since the model contains the starting points of each event. So accelerating is mostly based on parabolas, and the only thing we have to do is accelerating from v_1 to v_2, with acceleration d. An example is shown in figure 3.6

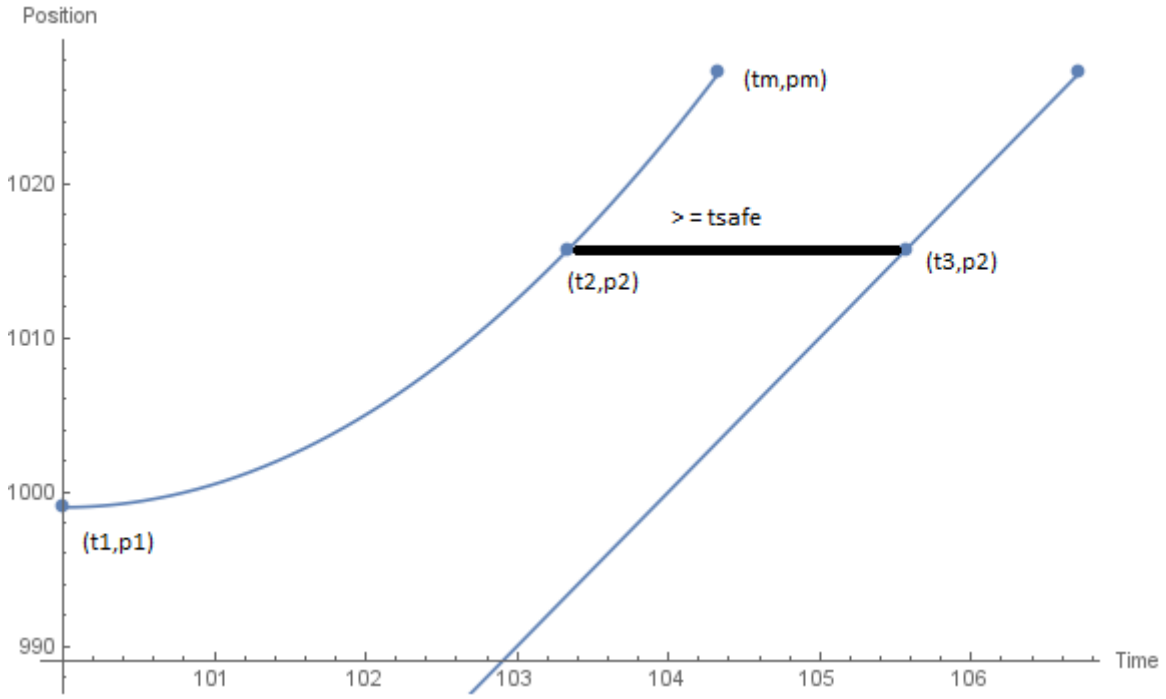


Figure 3.6: Check for first vehicle in case leader drives faster than follower

Although the calculations are very easy, it is nice to have a function that calculates all the values simultaneously. So knowing the speed v_2 of the follower, and the starting speed v_1 of the leader when it starts accelerating, we calculating the time and position where the speeds are equal. Knowing the position and the time, we can compare it with the expected time that the follower reaches this position. Based on the value of t_{safe}, the decision is made. The formulas, equations and solutions needed are described in equations 3.11, 3.12 and 3.13.

Function name: **ParabolaToV2**
Known variables: t_1, p_1, v_1, d, v_2.
To solve: t_2, p_2

$$s_1(t) = \frac{1}{2}d(t-t_1)^2 + v_1(t-t_1) + p_1 \tag{3.11}$$

Equations of **ParabolaToV2**

$$\begin{aligned} s_1(t_2) &= p_2 \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_2} &= v_2 \end{aligned} \tag{3.12}$$

Solutions of **ParabolaToV2**

$$\begin{aligned} t_2 &= t_1 + \frac{(v_2 - v_1)}{d} \\ p_2 &= p_1 + \frac{{v_2}^2 - {v_1}^2}{2d} \end{aligned} \tag{3.13}$$

So given the starting point of the acceleration (t_1, p_1) with speed v_1 and acceleration d, we find that the leader has speed v_2 at the point (t_2, p_2) that are stated in equation 3.13. To check whether we can define the follower as a first car, we have to calculate the time t_3 where the follower reaches the same position p_2. We assume that the follower is driving at constant speed v_2 at the entire trajectory. So this gives $t_3 = t_{3,l} = t_{1,l} + \frac{p_2 - p_1}{v_l}$. And if $t_3 - t_2 \geq t_{\text{safe}}$, then we have a new first car. Otherwise the leader influences the trajectory of the follower.

Note that when the occupancy of the road is high, or there are very long red times, then it is very unlikely to have many first cars. Many cars will have to react on the speed if the cars drive very close too each other. Also collisions will happen many times if the traffic lights are red. This is just vehicles joining the queue. So the

theorem of the first car may seem useless on crowded roads. But Smart Traffic is also not intended for roads that are almost over-saturated. When it is too crowded, then Smart Traffic does not perform better necessary than an intersection with cycle times. Many vehicles have to wait long, and Smart Traffic can not prevent that. It is useful when the there is average traffic. Then it can predict when the vehicles arrive the intersection and it will try to make the waiting times minimal. This is also the situation we stated in this model. We are looking at a starting point where there is free traffic, which is not over-saturated. And if that is that case then there might be several vehicles that turn out to be first cars, and then our theorem can be useful and save calculations.

Until now the trajectory of the first cars are explained. Next to that the method to determine whether a car is a first car is described. Now it is time to describe the trajectory of followers which have to react on its predecessor. The first thing we did was to correct the arrival time is case this arrival time is too close to its predecessor. Since we assumed a t_{safe}, then we have to make sure that it is not violated at any point. If these points are determined then we can go to the next step, which is checking the part of the trajectory where the collision takes places. To determine this part of the trajectory we make use of a function which calculates the collision. This may be the most important part of the model and also the most complicated part. In this part the most decisions are made. For the rest of the trajectory the vehicles will follow their leader, and somehow copy their trajectories. If the reaction on the collision is not good performed, it has consequences for the rest of the entire trajectory.

If the arrival of the follower is at $t_{1,l}$, then we need the set of events form the leading car such that the first event of this car is the event that happens at $t_{1,l}$. For each event in this list, we need to check whether the distance stays safe at the entire part of the trajectory. We know that there is a part where there is a collision, otherwise the vehicle was assigned to be a first car. In the model there are several types of events. These events are **arrival**, **constant speed**, **waiting**, **accelerating**, **decelerating**, and **departure**. Calculating whether there is collision depends on the type of event. We will discuss all possibilities.

The first possibility is that the event is arrival, constant speed or waiting. These arrivals all have the property that the speed is constant, in case of waiting it is zero. If the end point of this event is at least t_{safe} seconds before the next car arrives at this same end point. In these calculations we use differences in positions, which are linearly dependent on the speed of the vehicles. If the positions are large enough apart from each other, then there is no collision. The end point is equal to the starting point of the next event, so we use this time t_{next}, and calculate the positions of both vehicles. If the leading vehicle drives with v_1 and the follower with v_2, then the difference in position dif on t_{next} is given by:

$$dif = p_1 + (t_{\text{next}} - t_1)v_1 - p_2 - (t_{\text{next}} - t_2)v_2 \tag{3.14}$$

The pictures is as follows:

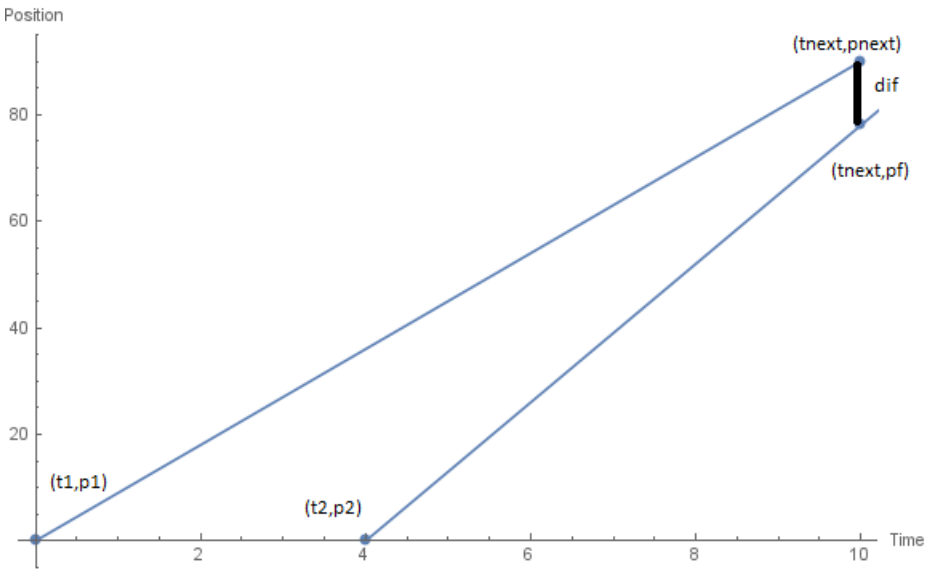


Figure 3.7: Illustration of collision during an arrival, constant or waiting event

Here (t_1, p_1) is the starting point of the current event of the leading car, driving with speed v_1. In case that the event of the leader is decelerating, then it is also enough to check the end point of the event, since that is

the moment where the car has the lowest speed. In this case the difference can be calculated by:

$$dif = \frac{1}{2}d(t_{\text{next}} - t_1)^2 + v_1(t_{\text{next}} - t_1) + p_1 - p_2 - (t_{\text{next}} - t_2)v_2 \tag{3.15}$$

Also here, the starting point of the event deceleration is (t_1, p_1) with speed v_1, and acceleration d, which must be negative in this situation.

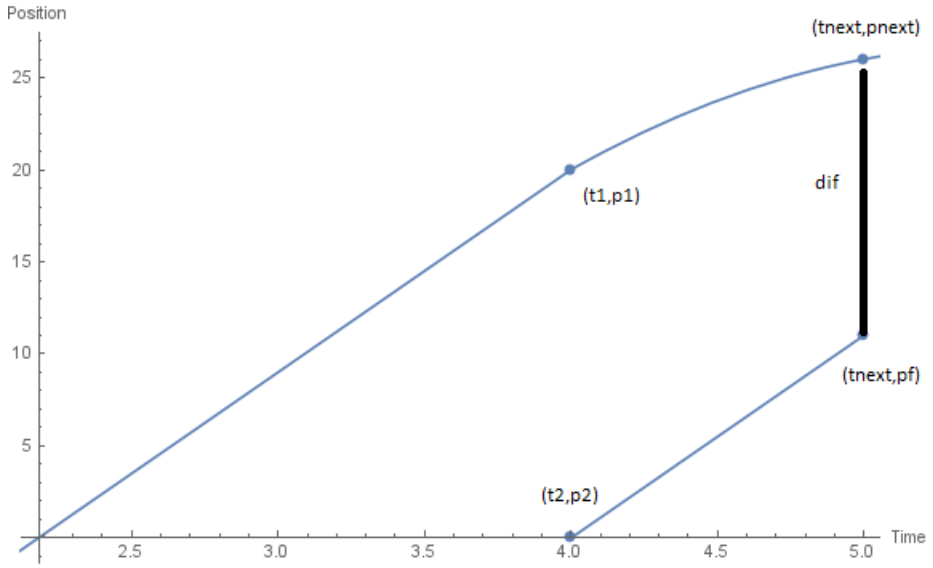


Figure 3.8: Illustration of collision during an decelerate event

The last case is a collision while the leader is accelerating. An example can be found in figure 3.9 This is the only case that it is not sufficient to look at the end point of the event, since it could be that the time distance is less than t_{safe} in the interval, but more than t_{safe} at the end point. So the difference can be calculated in this case by.

$$dif = \min_{t \in [t_1, t_{\text{tnext}}]} \left(\frac{1}{2}d(t - t_1)^2 + v_1(t - t_1) + p_1 - p_2 - (t - t_2)v_2 \right) \tag{3.16}$$

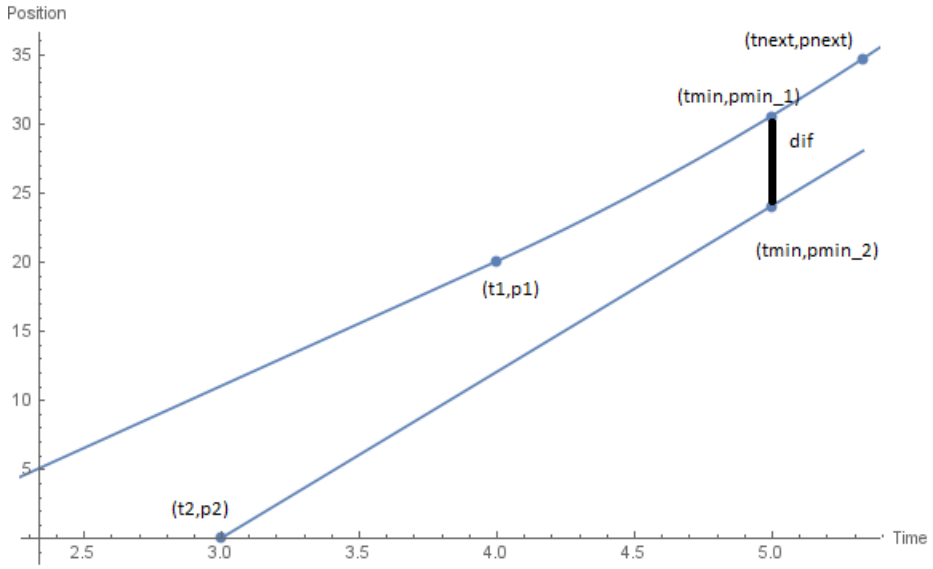


Figure 3.9: Illustration of collision during an accelerate event

Now all cases are described, and a decision has to be made. Since dif is a difference in position, we must translate it to time. This time depends on the speed. The follower should have at least t_{safe} seconds until he

reaches the same position as the leader. The minimal gap in position is therefore given by $dif_{\min} = v_2 t_{\text{safe}}$. If then $dif < dif_{\min}$ is found, then it means that there is indeed a collision and action should be taken based on this collision type.
So to conclude First for a car will be determined if it can be seen as a first car. If yes, then calculate the trajectory according to section 3.3. If not, calculate in combination with the leader trajectory what is the first event of the leader where the cars drive too close. If this event is found then action should be taken. In the next section we describe per collision event what should be done.

3.4.1 Arrival of a new car

Suppose a new car arrives at time t_1 and position p_1, and it is known that it is not a first car. The collision type is calculated. Then in this subsection we describe what actions a vehicle should take based on the type of collision.

Arrival or Constant
If the collision type is arrival or constant speed, then the following will be done. The vehicles collide on a constant speed, so this implies that the speed of the leader is lower than the speed of the follower. This situation we already described before when we calculated the moment of braking when the leader drives slower than the follower. When we were calculating whether a car should brake before or after the intersection we found the starting point of breaking, and the point that the follower has the same speed as the leader. This situation is exactly the same. The equations are 3.7, 3.8, and 3.9. So at time t_3 the vehicle starts decelerating at position p_3, and the deceleration stops at time t_4 and position p_4. However the situation that $t_3 < t_1$ can occur. So the deceleration should already have begun before the car even arrived. In this case we have to believe the data saying that the vehicle is starting there at that time and that speed. But it is impossible to make sure that the vehicle joins the leader by a single parabola with a given deceleration and with some constant speed parts.

The first option is to postpone the arrival, and change the speed of the following car. We know the car will join the rest very soon, so only the beginning of the trajectory will be different than the real trajectory. That is not a bad thing since the vehicle will join the leader very soon, and the main purpose is to determine the moment of passing the intersection. So the arrival of the follower will be t_{safe} later than the leader, but they both have the same speed. Thus we get a larger error in the beginning, but eventually the error becomes smaller, and it depends more on the leader.

If we want to make sure that the complete trajectory is a smooth curve with the given data, then another option would be to fit a third degree polynomial. This way it is possible to connect the start and end point with the correct speed. The disadvantage is that the deceleration is not constant at any part of the trajectory, which conflicts with the assumption that if accelerating / decelerating takes places, it is always at a constant rate. Both solutions will have a different effect. Since we fix the end point of the interval, only those will be compared in case of deceleration. If a collision is not found on this interval caused by this problem while it should be there, then the collision will be found on the next interval, and it will be corrected there. This is the advantage of changing the arrival. It gives very quick good results. The other option with the polynomial, gives decelerations that are not constant. Although it might not give that many problems, it is more desirable and also more realistic to use just one deceleration or acceleration.

Now we will discuss the case when the start of the deceleration happens after the arrival, which is the ideal case. However also if this occurs, we distinguish two cases. The difference has to do with the fact that at the point the follower has corrected its trajectory in such way that it is driving at the same constant speed as the leader, that in the mean time the leader could have changed from event. So if the leader for example accelerates, it is very unrealistic that the follower completes decelerating first and start accelerating many seconds later. So the way to control this is to first calculate the t_4 and p_4 from the solutions of 3.9. If this t_4 happens earlier than the end point of the collision interval, then there is no problem and the trajectory is given by the trajectory of 3.10.

In case that t_4 is after the end point of the collision interval, then this means that we have to adapt the deceleration process, since the leader is apparently doing something else already during the adapting process of the follower. For example if you are driving and there is a car that drives slower than you, then you must decelerate to avoid collision. If then the leader accelerates then you will also react on that acceleration. So it does not make sense to continue deceleration when you predecessor has begun to drive faster in the meantime. The strategy in this case, is that the follower joins the leader exactly at the end point of the collision interval. This can be done by first drive at constant speed, and then decelerate. However the deceleration is a variable we do not know in this case, since the maximum deceleration does not work. In this scenario the car will decelerate much earlier, and with a different rate. To calculate the point of deceleration and the deceleration rate, the

following will be done.

The functions of 3.17 describe that the vehicle first drives at constant speed s_1 beginning at (t_1, p_1) with speed v_1, and then decelerates s_2, such that at time t_2 it is at position p_2 with speed v_2.

Function name: **Condec**
Known variables: $t_1, p_1, v_1, t_2, p_2, v_2$.
To solve: t_3, p_3, v_3, d

$$\begin{aligned} s_1(t) &= p_1 + v_1(t - t_1) \\ s_2(t, d) &= \frac{1}{2}d(t - t_2)^2 + v_2(t - t_2) + p_2 \end{aligned} \tag{3.17}$$

In Equation 3.18 the equations needed to solve the desired variables are given:

Equations of **Condec**

$$\begin{aligned} s_1(t_3) &= s_2(t_3, d) \\ s_1(t_3) &= p_3 \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= \left.\frac{\partial s_2}{\partial t}\right|_{t=t_3} \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= v_3 \end{aligned} \tag{3.18}$$

The solutions of equations 3.18 are given by 3.19

Solutions of **Condec**

$$\begin{aligned} t_3 &= \frac{2(p_2 - p_1) - t_2(v_1 + v_2) + 2t_1 v_1}{v_1 - v_2} \\ p_3 &= \frac{-p_1(v_1 + v_2) + 2p_2 v_1 + (v_1 v_2 + {v_1}^2)(t_1 - t_2)}{v_1 - v_2} \\ v_3 &= v_1 \\ d &= -\frac{(v_1 - v_2)^2}{2(p_1 - p_2 + v_1(t_2 - t_1))} \end{aligned} \tag{3.19}$$

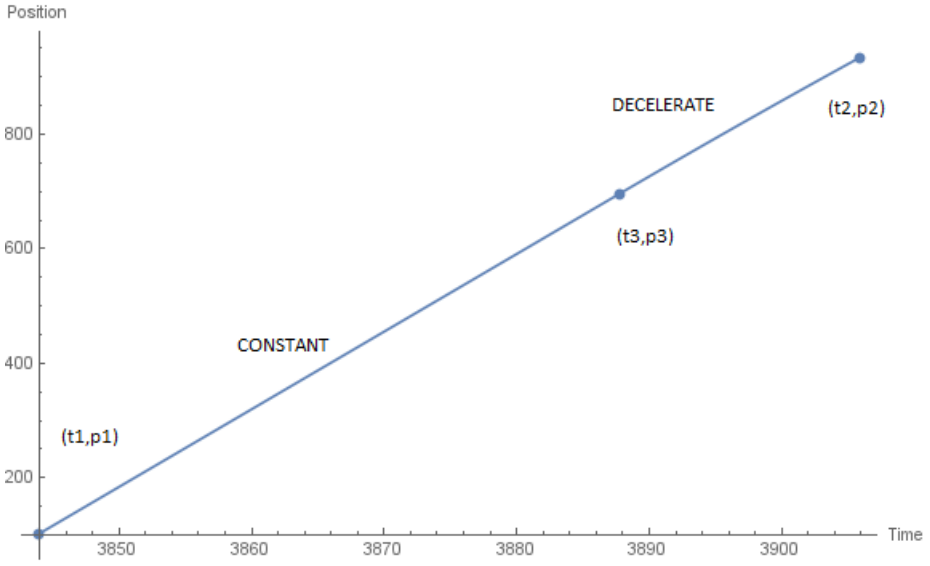


Figure 3.10: Reaction on constant speed collision when leader is accelerating during deceleration of the follower

Figure 3.10 shows an example of this situation. If this occurs then the deceleration rate is very low. So the vehicles are decelerating for a longer time, but the deceleration is slow. In the picture it almost seems that

there is only a constant speed, but to model it this way, the speed of the starting point is still correct, but also the end point of the model has the correct speed. From that moment one can model the trajectory in terms of the trajectory of the leader.

The last situation is a rare event and will therefore not occur often. But if it occurs then this problem should be taken into account. After the follower has the right speed and joined the previous car, all following events can be done t_{safe} seconds later than the leader, with a distance gap of $l_c + g_{\text{safe}}$ seconds.

Waiting
If the collision type is waiting then it means that the follower has to decelerate and thereafter has to stand still. This can be done by any deceleration rate, so for this one we choose the maximum rate. So the braking start as late as possible. The function we need for this is 3.2. The solutions are given by 3.4. In all these quantities, the number of the vehicle should be changed from 1 to the current ID. Also the position that the vehicle should stop is equal to the previous waiting position minus the length of the car and the safety gap.

Decelerating
If the collision point happens when the leader is decelerating, then the action that will be taken depends on the event that comes next for the leader. If the next event is waiting then a different action is needed, then when the leader starts accelerating after the deceleration.

The first case is that after the deceleration, the leader has to wait and therefore must make a stop. In this situation, we can pretend that the collision type is waiting instead of deceleration. Eventually the follower has to wait just like the leader, and its position is known. So the only thing to do is to make sure that the follower stops at the right position. To calculate the decelerate points, we can use the stoppingtime function again.

If the next event is constant speed, then we just copy the times and positions from the previous vehicle and adapt them with the safety gaps in time and positions. This happens from the start of the next event until the end. The problem that occurs here is that the decelerating part is not a parabola. If the end point of the leader is (t_2, p_2) with speed v_2, then a known point for the follower will then be $(t_2 + t_{\text{safe}}, p_2 - l_c - g_{\text{safe}})$, with speed v_2. Here, l_c is the length of the vehicle, and g_{safe} is the minimum gap that cars should have from the back of the leader to the front of the follower. But what exactly happens between those points is not described. From this end point on, the assumption is that the follower does exactly the same as the leader, but then t_{safe} time later, and at position $l_c + g_{\text{safe}}$ earlier. This process of copying the behaviour of the leader ends when the follower is interrupted and the leader not. This happens for example at the intersection if the leader has green, and the follower red. The other option to make it continuous is to fit a third-degree polynomial, but that is not necessarily needed for our purpose. We just need some points to that we can stick to for eventually estimating the time that the drivers cross the intersection line.

The last option is that after a deceleration comes an acceleration. Two decelerations directly in a row are not possible in our model. So this is the only option left. Since the follower sees the leader decelerate and then accelerate, it is likely that the follower will do something similar. In most cases, the follower will first decelerate. Then the follower sees that the leader is accelerating, so it stops decelerating and drives with a constant speed for a short period. When it is safe enough, the follower can start accelerating. For this process we build also a function that first drives at constant speed, then decelerates with rate d, constant speed, and lastly accelerates with rate a.

Function name: **Decconacc**
Known variables: $a, d, t_1, p_1, v_1, t_2, p_2, v_2, v_4$.
To solve: $t_3, p_3, v_3, t_4, p_4, t_5, p_5, v_5$

$$\begin{aligned} s_1(t) &= p_1 + v_1(t - t_1) \\ s_2(t, t_3, p_3) &= \frac{1}{2}d(t - t_3)^2 + v_1(t - t_3) + p_3 \\ s_3(t, t_4, p_4) &= p_4 + v_4(t - t_4) \\ s_4(t) &= \frac{1}{2}a(t - t_2)^2 + v_2(t - t_2) + p_2 \end{aligned} \tag{3.20}$$

In these equations we have to find t_3 which is the point where constant speed changes in decelerating, t_4 which is the point where decelerating changes back to constant speed v_4, and t_5 where constant changes in

accelerating. The point at (t_2, p_2) is chosen in the way such that the time and distance are safe. Also the speed v_2 is equal to the speed of the end point of the accelerating leader. The only choice that has to be made, is v_4. One choice could be the same speed as the speed of the leader in the turning point of decelerating into accelerating. Then the behaviour is the most similar as the leader, and no more information is needed since this information is saved by the model. Sometimes it is also possible to choose zero speed. Then you let the car stop and let him wait until it is safe to accelerate at the maximum acceleration rate to the correct speed. This option may not be that realistic, but for calculations it is easier and gives less problems. Also the event should then be named waiting instead of constant. An example is given in Figure 3.11.

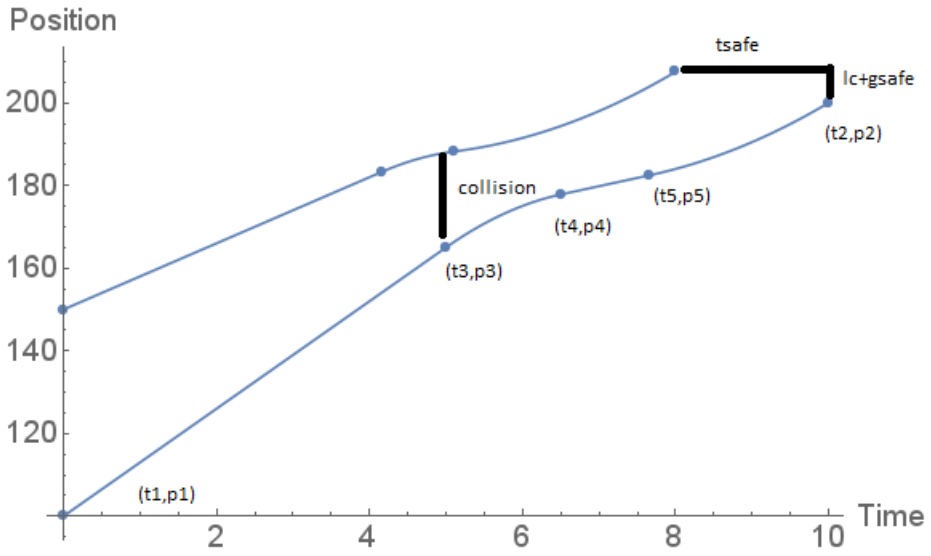


Figure 3.11: Collision during deceleration when the leader starts accelerating soon

Equations of **Decconacc**

$$
\begin{aligned}
s_1(t_3) &= s_2(t_3, t_3, p_3) \\
s_2(t_4, t_3, p_3) &= s_3(t_4, t_4, p_4) \\
s_3(t_5, t_4, p_4) &= s_4(t_5) \\
s_4(t_5) &= p_5 \\
\left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= v_3 \\
\left.\frac{\mathrm{d}s_4}{\mathrm{d}t}\right|_{t=t_5} &= v_5 \\
\left.\frac{\partial s_2}{\partial t}\right|_{t=t_4} &= \left.\frac{\partial s_3}{\partial t}\right|_{t=t_4} \\
\left.\frac{\partial s_3}{\partial t}\right|_{t=t_5} &= \left.\frac{\mathrm{d}s_4}{\mathrm{d}t}\right|_{t=t_5}
\end{aligned}
\tag{3.21}
$$

The solutions of this process are given in 3.22

Solutions of **Decconacc**

$$
\begin{aligned}
t_3 &= \frac{2ad(p_2 - p_1 + t_1 v_1 - t_2 v_4) + a({v_1}^2 - 2v_1 v_4) + d(2v_2 v_4 - {v_2}^2 - {v_4}^2)}{2ad(v_1 - v_4)} \\
p_3 &= p_1 + v_1(t_3 - t_1) \\
v_3 &= v_1 \\
t_4 &= \frac{2ad(p_2 - p_1 + t_1 v_1 - t_2 v_4) - a{v_1}^2 + d(2v_2 v_4 - {v_2}^2 - {v_4}^2)}{2ad(v_1 - v_4)} \\
p_4 &= p_3 - \frac{{v_1}^2}{2d(v_1 - v_4)} \\
t_5 &= t_2 + \frac{v_4 - v_2}{a} \\
p_5 &= p_2 + \frac{v_4^2 - v_2^2}{2a} \\
v_5 &= v_4
\end{aligned}
\tag{3.22}
$$

Unfortunately it is not always possible to make a trajectory with a constant part between decelerating and accelerating. Sometimes there is not enough time to do this. This can be checked by comparing the solutions in 3.22. It is fine if $t_1 \leq t_3 \leq t_4 \leq t_5 \leq t_2$ and $p_1 \leq p_3 \leq p_4 \leq p_5 \leq p_2$, and all speeds should be larger or equal to 0. If one of these rules does not apply, then we only have to use the decelerating and accelerating part. So starting at t_1, we decelerate at t_3, then accelerate at t_4 and are at the right position and speed t_2. The situation is shown in Figure 3.12. The corresponding formulas are given in the formulas of 3.23

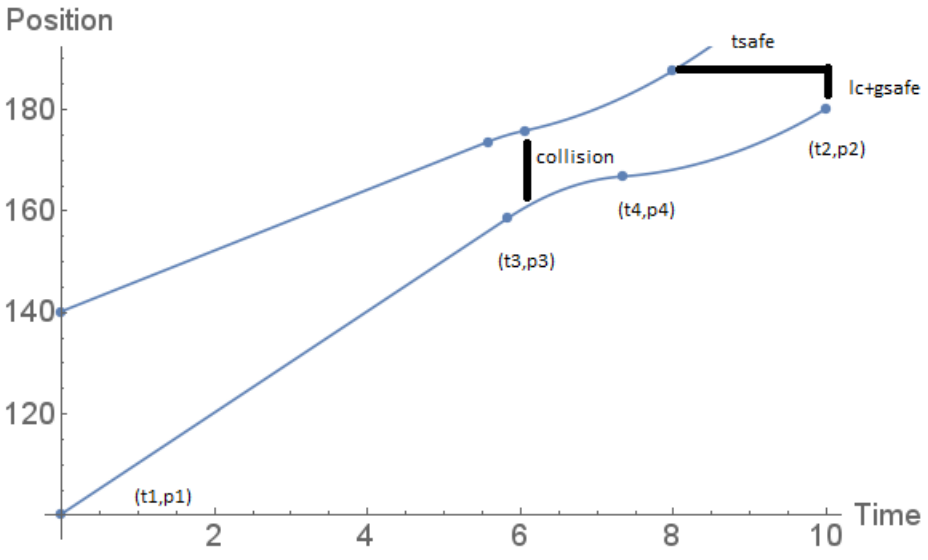


Figure 3.12: Collision during deceleration when a constant part is not possible for the follower

Function name: **Decacc**
Known variables: $a, d, t_1, p_1, v_1, t_2, p_2, v_2$.
To solve: $t_3, p_3, v_3, t_4, p_4, v_4$

$$
\begin{aligned}
s_1(t) &= p_1 + v_1(t - t_1) \\
s_2(t, t_3, p_3) &= \frac{1}{2}d(t - t_3)^2 + v_1(t - t_3) + p_3 \\
s_3(t) &= \frac{1}{2}a(t - t_2)^2 + v_2(t - t_2) + p_2
\end{aligned}
\tag{3.23}
$$

In 3.24 we have the equations. In this situation t_3 is the change from constant to decelerating, and t_4 is the change from decelerating to accelerating. t_2 is the point where the follower joins the trajectory of the leader.

Equations of **Decacc**

$$\begin{aligned} s_1(t_3) &= s_2(t_3, t_3, p_3) \\ s_2(t_4, t_3, p_3) &= s_3(t_4) \\ s_3(t_4) &= p_4 \\ \left.\frac{\partial s_2}{\partial t}\right|_{t=t_4} &= \left.\frac{\partial s_3}{\partial t}\right|_{t=t_4} \\ \left.\frac{\mathrm{d}s_3}{\mathrm{d}t}\right|_{t=t_4} &= v_4 \end{aligned} \tag{3.24}$$

The solution is given by 3.25

Solutions of **Decacc**

$$\begin{aligned} x &= (-2t_2(a^2 - ad) + 2(v_1 - v2)(d - a))^2 \\ y &= -4(a^2 - ad)\big((a^2 - ad){t_2}^2 + 2d(p_1 - p_2) + 2v_1(at_2 - dt_1) + 2v_2t_2(d - a) + (v_1 - v_2)^2\big) \\ t_3 &= -\frac{1}{d}\left(-at_2 + v_2 - v_1 + \frac{(a^3 - 2a^2d + ad^2)t_2 + (v_1 - v_2)(a - d)^2 + \frac{1}{2}(a - d)\sqrt{x + y}}{a^2 - ad}\right) \\ p_3 &= p_1 + v_1(t_3 - t_1) \\ t_4 &= t_2 + \frac{(v_1 - v_2)(a - d) + \frac{1}{2}\sqrt{x + y}}{a^2 - ad} \\ p_4 &= \frac{1}{2}d(t_4 - t_3)^2 + v_1(t_4 - t_3) + p_3 \\ v_4 &= d(t_4 - t_3) + v_1 \end{aligned} \tag{3.25}$$

Now we solved that problem, there could still be another problem. That is the problem that v_4 is negative, which is impossible since the cars are not driving backwards. But if that is the case, then there should be a feasible solution in the first option since when v_4 there is many space between begin and end point. So this problem does not occur and we should go back to the other situation "Decconacc" and use another value for v_4.

Accelerating
The last possible collision type is accelerating. Although this may seem different from the decelerating process. The calculations are similar. This has to do with the face that in case of decelerating we also checked the next event, and sometimes this is accelerating. If the collision is at an accelerating part, then this means that the follower first has to decelerate and then have to accelerate again. The decelerating part is clear since no one wants to have a collision, but accelerating is necessary to make sure the cars are driving in a flow. The only difference in this situation is that the rules are switched due to the fact that this process has a shorter time period needed than the previous case. So first we try a trajectory with only decelerating and accelerating. And if the speed of the transition point is negative, then we try a trajectory that includes a constant part between the decelerating and accelerating part. To calculate this we use the same functions, ("Decacc" and "Decconacc"), as we already used in the decelerating collision part.

3.4.2 After the arrival

Until now, we only discussed what happens after the vehicles arrive the system and what action they should take based on the events of the previous car. However the follower's trajectory also consists of events, and the follower should take actions based on the event that it is currently in. In the previous section, multiple events are calculated in the same time. Only when the vehicle has finished to do the last action, the model checks which state it currently is in, and executes the corresponding rules. Also in this situation we have to discuss the rules based on the different events, which are waiting, decelerating, accelerating, and constant speed.

Decelerating
The decelerating event occurs when the previous vehicle has to wait around this time, and the follower is decelerating from the free traffic. So this event can happen when the test with the collision type says that the follower should decelerate. The event of the leader is the waiting event that has already passed, or the first future coming waiting event in case it is not waiting yet. From this waiting event we get the waiting position, and with the stopping time function we calculate the times that the waiting process of the follower begins. From this point we schedule a new waiting event. The decelerate event is not the special event where the calculations

take place. The most calculations will be in the waiting event.

Waiting
When the current vehicle is waiting then there are two possible cases. The first option is the least complicated one. This is when the car is the first one in the row before the intersection line. The advantage is that there is no interaction with a vehicle in front of the follower. So as soon as the traffic lights turn to green, the vehicle can start accelerating and leave the road section.

The harder part is when there are cars in front of the current vehicle. If we are lucky we can just use the "stoppingtime" function and use the position of the leader minus the length of the car and safety gap. But issues can occur when the collision happens to be in the waiting period. We will therefore check this current action of the leading car. If the leader car is also waiting at the current time, then we have to check whether the car positions match, that means that the distance between the two fronts is equal to $l_c + g_{\text{safe}}$. The reason why we do this is that when a vehicle is waiting for the intersection, the position is determined by the queue-length. And in the mean time that we give the assignment that the follower has to wait, it could happen that the leader has moved forwards, and is waiting one or more positions more to the front. If the leader has moved, then we also have to move the follower. But since the leader has stopped again, it means that follower also has to stop. For this process we have the function "GoAndStop". First the car will accelerate with rate a and then decelerate with rate d. See for an example Figure 3.13. At t_1 the waiting process of the follower has started, but in the meantime the leader is driving and there will be a gap if the follower does not move any more. Therefore it starts the "GoAndStop" process.

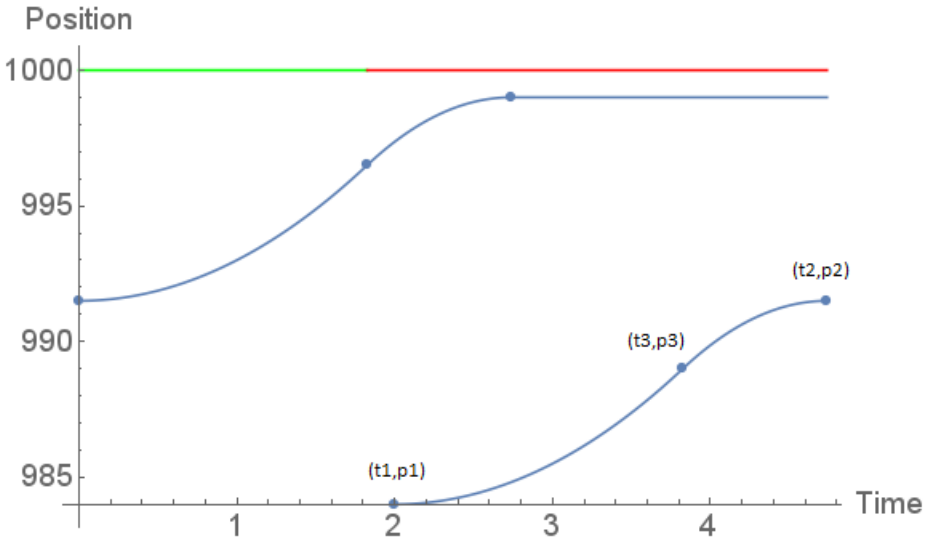


Figure 3.13: Movement of follower in case the leader has stopped waiting one position ahead of the follower

The formulas that describe the movement are stated in equation 3.26.

Function name: **GoAndStop**
Known variables: $a, d, t_1, p_1, v_1, p_2, v_2$.
To solve: t_2, t_3, p_3, v_3

$$s_1(t) = \frac{1}{2}a(t - t_1)^2 + p_1$$
$$s_2(t, t_2) = \frac{1}{2}d(t - t_2)^2 + p_2 \tag{3.26}$$

The equations that have to be solved are the following:

Equations of **GoAndStop**

$$\begin{aligned} s_1(t_3) &= p_3 \\ s_1(t_3) &= s_2(t_3, t_2) \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= v_3 \\ \left.\frac{\partial s_2}{\partial t}\right|_{t=t_3} &= \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} \end{aligned} \tag{3.27}$$

This gives the point (t_3, p_3) with speed v_3, where the accelerating process switches to decelerating. The time t_2 is the time where the vehicle reaches the new waiting position p_2.

Solutions of **GoAndStop**

$$\begin{aligned} t_3 &= t_1 + \frac{\sqrt{2ad(a-d)(p_1-p_2)}}{a^2-ad} \\ p_3 &= \frac{ap_1 - dp_2}{a-d} \\ v_3 &= a(t_3 - t_1) \\ t_2 &= -\frac{1}{d}\left(-at_1 + \frac{(a-d)\sqrt{2ad(a-d)(p_1-p_2)} + (a^3 - 2a^2d + ad^2)t_1}{a^2-ad}\right) \end{aligned} \tag{3.28}$$

If the positions are in such way that they only differ one position in the queue, then this means that from that moment on the actions will be the same as the leader, but only a time step t_{safe} later, and $l_c + g_{\text{safe}}$ earlier. Note that the trajectory ends always at a new event, and that it is a new event after the intersection. By copying all the events of the previous car it may happen that the last event starts before the intersection. If that is the case we remove the departure event of the follower, and use the last event as new input to make the trajectory complete.

It can also happen that at the moment the current vehicle starts with waiting, that the leader is not waiting any more. Then two things can happen. The vehicle accelerates and passes the intersection, or the vehicle has to make another stop. So if we find an event of the leader in the near future where it has to wait again, then another trajectory will find place than when a waiting part has not been found. If this waiting part has been found, then we make sure the follower accelerates and decelerates according to equations 3.26, 3.27 and 3.28. If the waiting event is not found, then the vehicle can do the same trajectory as the leader. Here we have to keep in mind that the vehicle can start accelerating t_{safe} after the leader did. So from that time we can copy the trajectory again.

Accelerating

This accelerating part is only the accelerating part when the vehicle is finished waiting. All the accelerating that may happen earlier, are scheduled together with the rest of the trajectory and is therefore never the end of a sequence of events. The accelerating event is to determine the period that the vehicle is accelerating based on the speed of the leader vehicle. So the first thing we have to know here is the speed that the leader can drive at maximum. After all the cars can not pass each other, and drive the speed that the slowest vehicle drives. So first will be determined at which position a vehicle is back at the speed that they eventually will drive. If this position is after the intersection, then a departure event is scheduled. But if the event happens to be before the intersection line, then a constant speed event is scheduled when the vehicle passes the green light. If acceleration is still going on at the moment of intersection, then the passing time is calculated with the following function. Since the vehicles are waiting, there is no starting speed. If acceleration stops before the intersection line, first a constant speed event is scheduled.

Function name: **ConstAccTime**
Known variables: a, t_1, p_1, p_2.
To solve: t_2, v_2

$$s_1(t) = \frac{1}{2}a(t - t_1)^2 + p_1 \tag{3.29}$$

Equations of **ConstAccTime**

$$s_1(t_2) = p_2$$
$$\left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_2} = v_2 \tag{3.30}$$

Solutions of **ConstAccTime**

$$t_2 = t_1 + \sqrt{\frac{2(p_2 - p_1)}{a}}$$
$$v_2 = \sqrt{2a(p_2 - p_1)} \tag{3.31}$$

In the worst case, the vehicle is the first one that would pass the intersection when the lights turned red again. Then the accelerating event will let the vehicle do the "GoAndStop" trajectory. However, the model checks the "GoAndStop" function as soon as the vehicle stops accelerating. Sometimes the accelerating part stops right before the intersection line. Then there is no possibility to decrease the speed to 0. In that case we have to remove some of the previous event, until it is possible to do the "GoAndStop" function.

Constant
The last possibility is that the event is constant speed. This constant speed can be after the intersection, where the cars passed the green light. If this happens then the vehicle is out of the system, and a departure event is scheduled. But if this event happens before the intersection line, then there are two options we have to distinguish. Since this event happens after a reaction of the predecessor car, it can happen that the predecessor is already accelerating in this time period again. The logical reaction of the follower is that if the leader accelerates, then the follower also accelerates. So if the event is constant speed the first thing to do is check for acceleration. If there is acceleration, then we also let the follower accelerate to the speed of the leader. The only exception is made when the leaders maximum speed is higher than the maximum speed of the follower. Then the acceleration will stop when the follower reaches its own maximum speed. If acceleration does not take place at this moment, then a new arrival is scheduled for the current car at the exact same location and time. From there on, a new collision will be calculated and this whole schedule starts over again.

3.4.3 Corrections

In general all possibilities are described in the rules above. But there is a very important part that we did not discuss. Many times the trajectories are calculated based on the collision type with the leader. Then the trajectory of the follower is adapted, and then we say that the follower does the same as the leader, but then with some delay and another position. But this can not always be true. When the leader passes the intersection in a green or yellow period, and the follower passes in a red period then we have a conflict. So after we calculated the trajectory entirely, we have to check whether this trajectory is possible. If it is not possible, then we have to remove some events in such way that it is possible for this car to make a stop at a given position.

The way to do this, is to consider the last event that a car does before the intersection, and calculate the time that the vehicle should pass the intersection according to the model and dependent on the type of event. If the calculations says that at the moment of passing the traffic light shows green, then the trajectory does not have to be adapted. But if it is red, then we use a "DetermineRemove" function to decide which events should be removed. The function "DetermineRemove" goes through the list of events that have happened before the intersection and it starts with the last one. If the last event was constant, then it will be investigated if decelerating is possible. If yes, then the new trajectory is calculated, and otherwise this event will be removed. If the event is removed, then we check the event that is then the last one. The event can not be waiting, since waiting only happens before the intersection and not at the intersection. In the rare case the event is decelerating. But if the decelerating is happening at the intersection, then it means that the deceleration started to late. So it is not useful to calculate if stopping is possible since we already know the answer. Decelerating will always be removed from the event list, since this problem can be solved by a previous event. The part where it mostly goes wrong is when the vehicle is accelerating. So if all other options do not work to find a correct trajectory, then we remove all events until the last acceleration. The first function we use here is the "Accdec" function. The situation is illustrated in Figure 3.14. In this situation the follower should have copied the trajectory of the leader, but then it drives to red. All the events are removed until there is an acceleration and from there on, a new trajectory is calculated so that the car stops right before the intersection at (t_2, p_2).

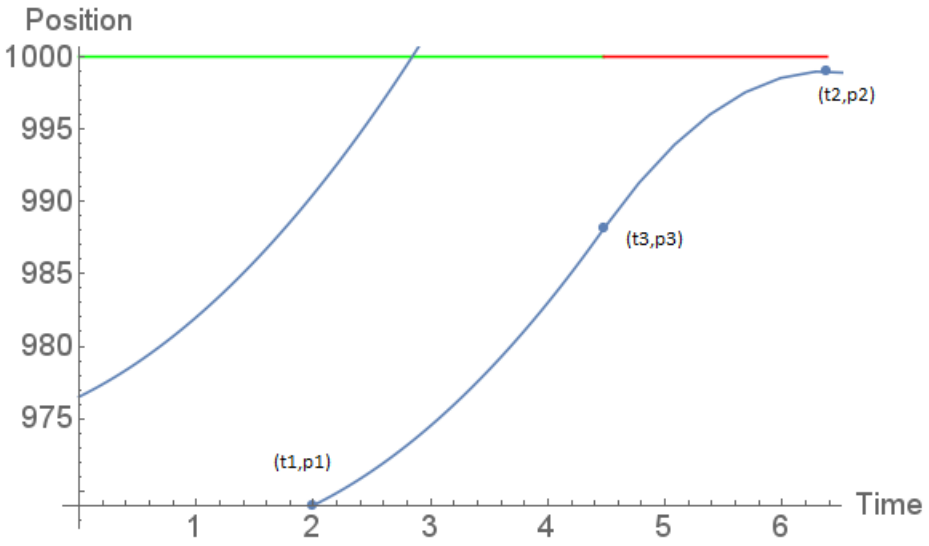


Figure 3.14: Followers that have to break due to the traffic light

The corresponding functions are:

Function name: **Accdec**
Known variables: $a, d, t_1, p_1, v_1, p_2, v_2$.
To solve: t_2, t_3, p_3, v_3

$$\begin{aligned} s_1(t) &= \frac{1}{2}a(t-t_1)^2 + v_1(t-t_1) + p_1 \\ s_2(t,t_2) &= \frac{1}{2}d(t-t_2)^2 + v_2(t-t_2) + p_2 \end{aligned} \tag{3.32}$$

Since the event that we start looking at is acceleration, we also have to use the trajectory of acceleration which is a parabola. The purpose of using these functions is to investigate whether it is possible that the vehicle still has an acceleration part, and then can stop at the given position p_2. The reason to consider the trajectory this way is that we would like to include the accelerating part since driving at a lower speed than necessary is unrealistic. The equations that have to be solved are:

Equations of **Accdec**

$$\begin{aligned} s_1(t_3) &= s_2(t_3, t_2) \\ s_1(t_3) &= p_3 \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= v_3 \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= \left.\frac{\partial s_2}{\partial t}\right|_{t=t_3} \end{aligned} \tag{3.33}$$

With these equations the transition point is calculated at time t_3, position p_3 and speed v_3. The time where the vehicle starts the waiting process is t_2.

Solutions of **Accdec**

$$
\begin{aligned}
x &= 2ad(d-a)(p_2-p_1) + (d^2-ad){v_1}^2 + (a^2-ad){v_2}^2 \\
t_2 &= -\frac{1}{d}\left(-at_1 + v_1 - v_2 + \frac{(at_1-v_1)(a-d)^2 + (a-d)\sqrt{x}}{a^2-ad}\right) \\
t_3 &= t_1 + \frac{(d-a)v_1 + \sqrt{x}}{a^2-ad} \\
p_3 &= \frac{2ap_1 - 2dp_2 + {v_2}^2 - v_1^2}{2(a-d)} \\
v_3 &= \frac{a\sqrt{x}}{a^2-ad}
\end{aligned}
\tag{3.34}
$$

But also here things can go wrong. The first thing is that t_3 has a lower value than t_1, and therefore no feasible solution exists. If this happens then it means that the accelerating part starts too late. Even if the accelerating part has been skipped, there is not enough time for the deceleration part. Then we have to go back further in time and use an earlier event. If there is any other acceleration event in the past we choose that one. If there are no other options mostly the first arrival is used to drive constant and let the vehicle stop before the intersection. On the other hand, it can also happen that v_3 is higher than the maximum speed, or the maximum speed that the current vehicle wants to drive. This is not permitted and if it does happen, then we have to include a constant speed event between the accelerating and decelerating. This constant speed is the maximum speed of the vehicle. For the situation of a too high v_3, we use the function "Acccondec". Figure 3.15 shows an example:

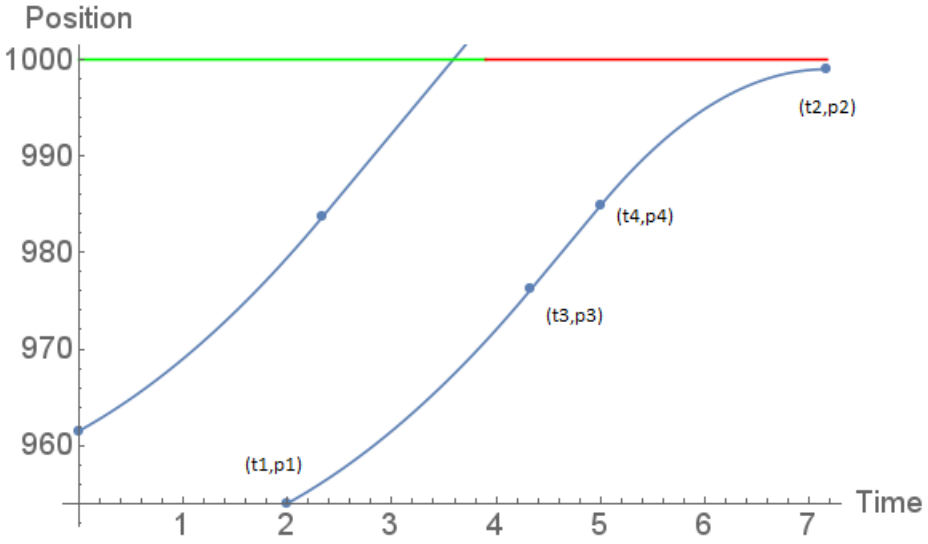


Figure 3.15: Correction in case the speed of v_3 would be to large in accdec example.

Function name: **Acccondec**
Known variables: $a, d, t_1, p_1, v_1, p_2, v_2, v_3$.
To solve: t_2, t_3, p_3, t_4, p_4

$$
\begin{aligned}
s_1(t) &= \frac{1}{2}a(t-t_1)^2 + v_1(t-t_1) + p_1 \\
s_2(t, t_3, p_3) &= p_3 + v_3(t-t_3) \\
s_3(t, t_4, p_4) &= \frac{1}{2}d(t-t_4)^2 + v_3(t-t_4) + p_4
\end{aligned}
\tag{3.35}
$$

Here v_3 is the maximum speed the vehicle will drive at the constant speed part. The equations to be solved are:

Equations of **Acccondec**

$$\begin{aligned} s_1(t_3) &= s_2(t_3, t_3, p_3) \\ s_2(t_4, t_3, p_3) &= s_3(t_4, t_4, p_4) \\ s_3(t_2, t_4, p_4) &= p_2 \\ \left.\frac{\mathrm{d}s_1}{\mathrm{d}t}\right|_{t=t_3} &= \left.\frac{\partial s_2}{\partial t}\right|_{t=t_3} \\ \left.\frac{\partial s_3}{\partial t}\right|_{t=t_2} &= v_2 \end{aligned} \tag{3.36}$$

The solution of the transition from accelerating to constant is given by t_3 and p_3. The transition point from constant to decelerating is given by t_4 and p_4, and the end point where there car will be waiting is at time t_2.

Solutions of **Acccondec**

$$\begin{aligned} t_2 &= t_1 + \frac{p_2 - p_1}{v_3} - \frac{v_1}{a} + \frac{v_2}{d} - \frac{v_3}{2d} + \frac{d{v_1}^2 - a{v_2}^2}{2adv_3} \\ t_3 &= t_1 - \frac{v_1}{a} \\ t_4 &= t_2 + \frac{v_3 - v_2}{d} \\ p_3 &= p_1 - \frac{{v_1}^2}{2a} \\ p_4 &= p_2 + \frac{{v_3}^2 - {v_2}^2}{2d} \end{aligned} \tag{3.37}$$

Now we have discussed all the issues that can happen during the trajectory when a car may have gone through a red light by following the leader. The purpose of this model was to describe the trajectories of the vehicle and to determine when they pass the intersection. The model that is described until now was based on the fact that we use only the data of one sensor at the beginning of a road section. These sensors can have errors in time but also in speed. Since the sensor lays in an area where there is free traffic, it is the best idea to use the measured speed, since in most cases it does not differ much from the maximum speed of the road. One can also choose to use the maximum speed of the road instead of the sensor data since in reality most vehicles are in fact driving around that speed. But use the measured speed is more relevant and interesting to test the model. Interactions occur more often if the speeds differ, and if we include this in the model it is more attractive to use than the restriction of only one speed. Exceptions are vehicle types which are not allowed to drive faster than a given speed. But this case we do not take into account in this model, since we need to vary the vehicle types.

Another problem is that one can choose a distance that is so large from the intersection that the model and reality become very different and do not match at all. It would therefore be nice to have more information of the trajectory than only one sensor. In reality it is mostly the case that there is a sensor that is far away from the intersection, and that there are one or multiple sensors in front of the traffic lights. These sensors are in the road for a reason and it would be a weird if we do not use this information. Therefore we would also like to use these sensors to give a more accurate time that vehicles will drive at the intersection. An extended model will be described in subsection 3.4.4.

3.4.4 Multiple sensors

By only using the first sensor, many trajectories will be different from the actual trajectory. In this subsection we will adapt the trajectories in such way that at certain points the model and the real trajectory will only differ t_{dif} time. In reality it is often the case that there are 3 sensors. One very far away, which is the sensor we used in the first model, the second one at a short distance from the intersection, and the last one at the intersection itself. The last sensor is for our purpose not interesting. Smart Traffic wants to predict the times that the vehicles arrive and pass the intersection line. If the traffic light is red and the sensor at the intersection is high, then it is already to late to prevent the car from stopping if the action to put the lights to green is done at the latest point. It is more interesting to look a little bit farther away, but in a reasonable distance from the intersection. So for this model we look at two sensors, the sensor at the start, and a sensor that is a few dozen meters away from the intersection.

The model starts the same as the first model. At first the entire trajectory will be drawn by the data given by the first sensor. Based on the events, the times can be calculated when the vehicle arrives at the second sensor according to the model. Then also the vehicles that arrive after the first car will get their entire trajectory by the first model. As soon as the first car reaches the second sensor, the sensor data will be compared with the model data. This comparison is in the first instance only based on time, and not on speed. If the difference between the model and the sensor data is smaller than t_{dif}, then it means that the model did a good prediction and it is not necessary to adapt the trajectory. So the predicted time that the vehicle will depart the system stays the same as calculated in the beginning. The other case is that the difference is larger than t_{dif}. There are multiple reasons that this can occur. For example wrong vehicle speed, or human behaviour can influence the times easily. The idea in this model is that the past of this vehicle is erased, and a new trajectory is calculated from the moment it hits the second sensor. Also the speed at the second sensor will be used for the rest of the trajectory. The past is not important anymore for the current vehicle. However, the trajectories of the next vehicles are based on the trajectories of the previous cars. And we can not pretend that the vehicles that are adapted are not there before the last sensor.

For calculating the trajectories of the following vehicles, it is enough of only using the last sensor point of which the leader has adapted its trajectory. Sometimes this is the beginning point when the model predicted all the next sensor positions right, but it can also be that the last sensor was high at another moment than the model. In this case the most accurate point is right before the intersection, and every part of the trajectory before this sensor is useless. For the model it is necessary to tell what is the last sensor point that has to been taken into account. The idea is to make a new arrival at the other sensors in case of adaptation, and then again search for collisions. Since the trajectory of the leader was apparently wrong it does not make sense to look for collisions on that part of the trajectory.

For some reasons it might be necessary to describe the trajectory between the sensor points. Now it feels like there is information missing if one wants to describe the total trajectory. In the plots it will cause gabs, since only the last part will be used. Although these parts will not be used for the following trajectories we discuss two easy ways, both with advantages and disadvantages to describe the parts between the sensors.

The first option is to look first whether the vehicle hits the second sensor earlier than expected or later. Depending on that, it means that during this period on average, the vehicle was accelerating or decelerating. Since the past does not matter, we could give the initial speed another value then the measurements gave. If we do this we can fit one simple curve between the two points, such that the speed at the second sensor is correct. The second point is the most important point since it is the actual traffic situation, and this should be the same as the sensor point which is given as input for the model. The new trajectory between the two sensors when the leader hits them, can be calculated as follows:

Function name: **ParabolaAccv2known**
Known variables: t_1, p_1, v_1, t_2, p_2.
To solve: d, v_1

$$s_1(t, d) = \frac{1}{2}d(t - t_2)^2 + v_2(t - t_2) + p_2 \tag{3.38}$$

The equations are:

Equations of **ParabolaAccv2known**

$$\begin{aligned} s_1(t_1, d) &= p_1 \\ \left.\frac{\partial s_1}{\partial t}\right|_{t=t_1} &= v_1 \end{aligned} \tag{3.39}$$

And the solutions are the following:

Solutions of **ParabolaAccv2known**

$$\begin{aligned} d &= -\frac{2(p_2 - p_1 + (t_1 - t_2)v_2)}{(t_1 - t_2)^2} \\ v_1 &= v_2 + d(t_1 - t_2) \end{aligned} \tag{3.40}$$

Now these equations are not complicated at all, but these calculations can give some consequences. The

speed v_1 that is calculated from these equations can be unrealistically high or low. That becomes the case when the difference between the model and the sensor data becomes large. If the vehicle comes to early at the sensor then there is a large acceleration, and the starting speed of the leader changes to a very low value. The result of this is that if we had used this as input that the following vehicles that arrive see a car with a very low speed in the beginning, and according to the model they have to react to that and start decelerating, while in fact there may be nothing that is driving in the way. In the worst case, the calculated speed is less than zero. The last solution to solve this problem would be to let the vehicle wait on the sensor and let it wait until it can start accelerating. Otherwise, it can also be that the model predicts an earlier arrival at the sensor than what actually happens. This means that the trajectory is decelerating and that the starting speed is higher than the actual measurement. It is possible that the starting speed is higher than the maximum speed, and therefore unrealistic. In contrary to the acceleration of the leader, the following vehicles would start decelerating later than expected when the leader is decelerating. So using this method would probably not be a good idea.

The second way of connecting the two sensor points is to use third degree polynomials instead of parabolas. Where it is not possible to connect one parabola between two points, and also have the right speeds at begin and end points, it can be done with a third degree polynomial. So given (t_1, p_1) with speed v_1, and (t_2, p_2) with speed v_2, we can fit exactly one curve. But unlike the parabolas, the third degree polynomials do not have the property of having constant acceleration over the trajectory. Although connecting the parts between the sensors is not necessary for every purpose, we will use the following method for similar purposes later in the trajectory where we can not describe a curve by a single parabola. This method is sort of a last resort if we want to connect two points of events by a curve. Therefore we will describe this method in more detail. Note that a and d are not acceleration and deceleration in this function.

Function name: **ThirdDegreeCurve**

$$s_1(t) = at^3 + bt^2 + ct + d \tag{3.41}$$

The equations to find the coëfficiënts are:

Equations of **ThirdDegreeCurve**

$$\begin{aligned} s_1(t_1) &= p_1 \\ s_1(t_2) &= p_2 \\ s_1'(t_1) &= v_1 \\ s_1'(t_2) &= v_2 \end{aligned} \tag{3.42}$$

Which gives as result:

Solutions of **ThirdDegreeCurve**

$$\begin{aligned} a &= -\frac{2(p_1 - p_2) + (t_2 - t_1)(v_1 + v_2)}{(t_1 - t_2)^3} \\ b &= -\frac{3(t_1 + t_2)(p_2 - p_1) + ({t_1}^2 + t_1t_2 - 2{t_2}^2)v_1 + (2{t_1}^2 - t_1t_2 - {t_2}^2)v_2}{(t_1 - t_2)^3} \\ c &= -\frac{6t_1t_2(p_1 - p_2) + (2{t_1}^2t_2 + t_1{t_2}^2 + {t_2}^3)v_1 + ({t_1}^3 - {t_1}^2t_2 + 2t_1t_2)v_2}{(t_1 - t_2)^3} \\ d &= -\frac{{t_1}^2p_2(3t_2 - t_1) + {t_2}^2p_1(t_2 - 3t_1) + (t_1^2{t_2}^2 - t_1{t_2}^3)v_1 + ({t_1}^3t_2 - {t_1}^2{t_2}^2)v_2}{(t_1 - t_2)^3} \end{aligned} \tag{3.43}$$

The acceleration function is the second derivative of s_1, and it is computed by.

$$a_1(t) = s_1''(t) \quad = 6at + 2b \tag{3.44}$$

The moment where the second derivative is zero determines the type of the trajectory. This happens at time $t_{a0} = -\frac{b}{3a}$. There are four options for this trajectory.

1. Only acceleration, if $(t_{a0} \leq t_1 \quad \wedge \quad a > 0) \quad \vee \quad (t_{a0} \geq t_2 \quad \wedge \quad a < 0) \quad \vee \quad (a = 0 \wedge b > 0)$

2. Only deceleration, if $(t_{a0} \leq t_1 \quad \wedge \quad a < 0) \quad \vee \quad (t_{a0} \geq t_2 \quad \wedge \quad a > 0) \quad \vee \quad (a = 0 \wedge b < 0)$
3. First acceleration, then deceleration, if $t_1 \leq t_{a0} \leq t_2 \quad \wedge \quad a < 0$
4. First deceleration, then acceleration, if $t_1 \leq t_{a0} \leq t_2 \quad \wedge \quad a > 0$

Note that if there only is acceleration or deceleration it is also allowed that $a = 0$. But then t_{a0} does not exist since the acceleration is constant and therefore can not be zero. If this rare situation occurs, then the trajectory is a simple parabola. But most of the cases a will be unequal to zero, and a third degree polynomial is used.

The next problem is that the events we described until now also received a constant acceleration or deceleration rate. All the methods we used for calculating next trajectories are based on these constant acceleration rates. In this situation there is no constant rate. It changes linearly over time. In our model we approach this rate by an average rate. We split acceleration from deceleration and for both we calculated what was the mean acceleration or deceleration over the interval. Other options would be to take the maximum deceleration rate, and the minimum acceleration rate, which will give the worst case result. The advantage with these rates is that there will be no collisions during the model, but the big disadvantage is that the following car has a trajectory that is much more delayed than the actual trajectory. So for now it seems better to use an average.

So the rates for the four options are:

1. $\overline{acc} = \dfrac{a_1(t_1) + a_1(t_2)}{2}$
2. $\overline{dec} = \dfrac{a_1(t_1) + a_1(t_2)}{2}$
3. $\overline{acc} = \dfrac{a_1(t_1) + a_1(t_{a0})}{2} \quad \text{and} \quad \overline{dec} = \dfrac{a_1(t_{a0}) + a_1(t_2)}{2}$
4. $\overline{dec} = \dfrac{a_1(t_1) + a_1(t_{a0})}{2} \quad \text{and} \quad \overline{acc} = \dfrac{a_1(t_{a0}) + a_1(t_2)}{2}$

The advantage of this method is that these functions connect indeed two points with correct speeds. There are several disadvantages. If option 3 is used, there can be a problem with the acceleration. We did not set a limit on this acceleration, and therefore there can occur speeds that are over the speed limit, or even unrealistically high. The same problem occurs when option 4 is used. In some cases, the speed can become negative, and this is also impossible unless cars drive backwards, what we do not assume.

Thus, option 3 could give wrong answers, but this is not necessarily bad for the model. During the accelerating part there will be no collisions this way. These will occur during the deceleration part. In the entire process, deceleration rates will not be very large. So for the follower it is possible to use a larger deceleration rate for example the maximum rate. The only requirement is that the time, the position and the speed match with the end point of the deceleration of the leader.

Option 4 is more a problem since followers can not react on negative speeds of the leader. This method is therefore not appropriate in this situation. The reason why this happens is that there is a lot of time to go from begin to end point. But these begin and end point have a distance that is too small for the large time period. The best option is too park the car somewhere between the begin and end point. For this situation we can go back to parabolas again.

3.5 Controlling the traffic lights

In Smart Traffic it is the purpose to optimize intersections and use waiting times and arriving cars to determine how long traffic lights have to be green. Until now we only used cycle times, but it would be nice to make something in such way that cycles are not necessarily needed. The trajectories that are described above can be used for this purpose. At any time t we would like to check for how long we should let the light be green if we would immediately turn it to green at that moment. This moment could be chosen in the future, but then there may be new arrivals that come in the future that can not be included since they are not there yet. So for this purpose, let's assume that we choose a time t and immediately turn the lights to green, and we have a sensor at the position of the intersection. Our model uses input sensor data and calculate the time that it leaves the system. So we can make a list of in and out flow of the vehicles. For this purpose we need to now how many

vehicles have a delay. This part uses ideas of the Fluid-Dynamic Model. The difference is that it does not use flow rates, but uses the actual situation on the road. The method for determine this number is the following.

1. Choose a time t when the lights should turn green.
2. Calculate the time t_0 of a vehicle that departs at t the system and had a constant speed of v_{max}.
3. Calculate n_a, the number of vehicles that entered the system in the interval $[0, t_0]$.
4. Calculate n_d the number of vehicles that departed the system in the interval $[0, t]$.
5. Calculate, $q = n_a - n_d$, the number of delayed vehicles at time t.
6. Determine $I_{\min}$ the ID of the first vehicle that has delay, which is $n_a + 1 - q$.

Now we have the number q, the number of vehicles that should have depart at time t, but are still in the system, and I_{min} the first vehicle in the row that already should have left. The trajectories of all vehicles have been calculated completely based on the cycle times. So at time t we can calculate the position and the speed at this time, since we know the event that this vehicle is doing. If at time t the lights turn green, we determine the position and the speed and calculate a new trajectory from that moment on. This will be done for all vehicles with an ID larger or equal than $I_{\min}$, until the last vehicle that entered before time t.

The vehicle with ID $I_{\min}$ is the first car in the row, and will therefore be treated as a first vehicle. The event that the vehicle was in at time t will be used as input, but then starting at time t, and new calculated position and speed. From there on every function that decides about green will tell that the moment of passing is green, and that is the main difference with the original model. In Figure 3.16 an example is stated.

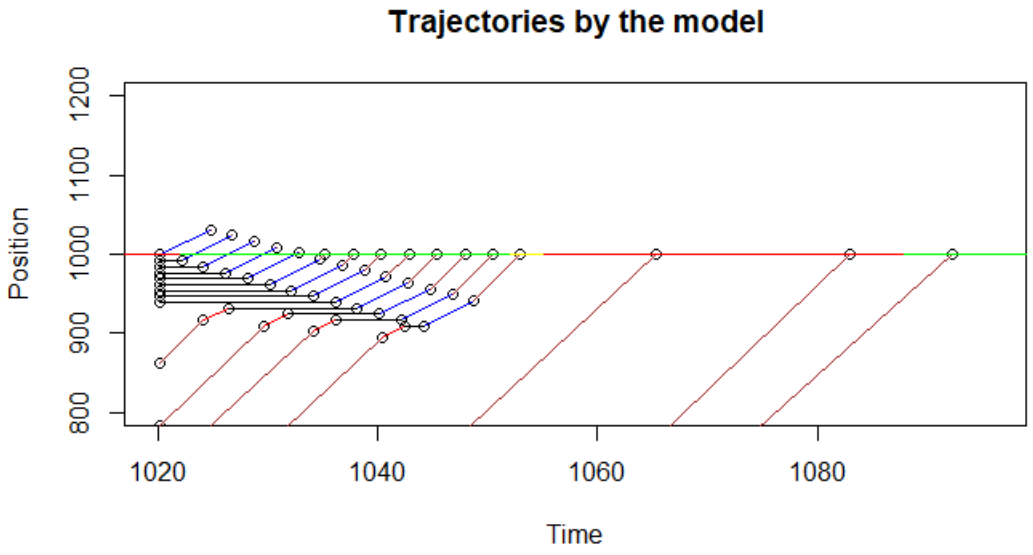


Figure 3.16: Switch everything to green and determine how long green is necessary

In Figure 3.16 the trajectories are shown when the decision is made to turn the lights to green at a moment t. At time $t = 1020$ in this case, all vehicles in the system have a dot in the picture, which means the location and speed has been calculated. The colors of the traffic light in this figure were the original states of the lights. If is visible that originally the light was red at $t = 1020$, but we would like to make it green. The first vehicle was waiting at that moment. Immediately it will accelerate to the maximum road speed. All the other vehicles in the row adapt their trajectories based on the first vehicle, and eventually the queue is dissolved. When the queue is dissolved there are still other vehicles that entered the system before time 1020. The last 3 do not have any disturbance of the other vehicles, and therefore the headways are larger than the vehicles that were in the queue. Smart Traffic optimizes the green times based on waiting times. They can decide to extend the green times to let vehicles pass that enter the intersection several seconds later, but if other conflicted directions have cars waiting it may be better to switch this traffic light to red earlier. The function to determine the traffic light states is now based on cycles, but other functions can be developed, to ask whether a light is green, or when the next green period starts. Every control system has a maximum red period so vehicles should get green at a certain time. Every time a vehicle enters the system, calculations can take place to determine the new

situation and to determine whether the lights should change. A minimum green period will be determined if the lights do change to green, and at the end of the period, a new calculation takes place to determine if the green period should be extended. All these optimizations we do not discuss in this paper, but we have now a method to estimate departure times of vehicles in the system. These departure times can be used by Smart Traffic to make decisions about switching the lights. Although we use the cycle times in the rest of the papers, this extension can also be used for Smart Traffic.

Chapter 4

Simulations

The simulations that are going to be done, are done in the program SUMO. This program is one of the programs the SWECO uses to simulate the projects that are done on the road. It can be used for entire cities, large networks, but also small networks like a single intersection. SUMO uses mainly microscopic models, since it uses interactions between individual vehicles, but macroscopic and mesoscopic models are also possible use. The microscopic model is also the type of model that will mainly be used during this paper, since the positions of each vehicle are what is of interest. So instead of real traffic data of the actual road, we use data from a simulation program. One of the reasons to use SUMO instead of real traffic data, is that real data can have inaccuracy in measurements, but there are also many differences in driver behaviour. Real sensors can not give exact times or speed so testing the model with this data gives many complicated side effects that can influence performances. Of course in real traffic there are many inaccuracies, but those should be learned from the traffic situation itself, and can then be used as model parameters, or changes in assumptions. For the use of Smart Traffic, driver imperfection has to be taken into account but it is not necessarily needed in the model now, since it is not a variable we are investigating. In Smart Traffic, there is input data needed to model the trajectories, and SUMO can also give this input data. Sensors can give a time that the vehicle enters a sensor, and when it leaves a sensor. Also the speed at that point is determined. To calculate the trajectories SUMO already uses a car-following model, but the calculations are updated after each time-step, which gives a lot of information that is not needed to determine the trajectory, and to determine when the car passes the intersection. When performing a simulation it gives a nice view when all cars are plotted each time-step on the right place on the road. But if one wants to use the data of each vehicle at each time-step, then the output of the data files become so large that the computer crashes when the simulation takes too much time, or the network is too big. Therefore it is better to only use a few time points that can also give a good indication of what the trajectory is. So in this simulation we only use the data that the sensor gives, instead of all data points of the vehicles. In section 4.1 we first describe how to build our network, and what input parameters can be included and in what files it should be put.

4.1 SUMO files

If we want to build a network in SUMO there are multiple ways to do that. Every part of the network will be saved in "XML" files, and will later be included into one single configuration file. So building the network can be done by typing everything in XML-files, or using a program so one can view the network immediately. In this case we use NETEDIT to visually build the network. The first thing to do is to create the nodes and the edges. All these parts can be found in a ".net.xml" file.

Nodes
The nodes have to get an ID, and the position of the node have to be defined. At these nodes, we can let vehicles into the system, or let them out. So at these points, we have to determine what we would like to give them as input.

Edges
The edges have also an ID, and the begin and end note of this edge have to be specified. The choice of reverse directions can be made, and the number of lanes in both directions can be included. Also the maximum speed of the road will be included and will also be used as input for the car-following model.

Connections
The connections between the edges have to be checked. Most of the connections are made when the nodes and

edges are created. But when the network becomes larger, and for example multiple lanes are involved, or when there are roads that certain directions may not go in, then new connections have to be made, and others have to be removed. These connections are specified by the edges, but also the lanes. This list is created in the XML-file. Also the direction is included in this file, for example a left turn, or straight ahead.

Traffic Lights
The states of the traffic lights have to be specified. This includes the type of the state each lane is in, and the duration of the phase. The connections are already made.

All these parts together form the "net-file" of the simulation. Below is a small example of what this looks like.

XML files
If one creates a network in NETEDIT, then the nodes are mostly translated to a junction in the XML file. The description of the junction can be read below. For example, the name of the node is N1, and the type is a dead end, since only traffic can enter the system here. The x and y give the coördinates of the location of the node. The part inclanes, gives the names of the lanes that go to this node, and intLanes includes the names of the lanes that go through the intersection. But in this case there are none, since it is not an intersection. The shape describes how the junction looks like, and in this case that is also not exciting since it is just a note without intersections.

```
<junction id="N1" type="dead_end" x="-1007.20" y="0.00" incLanes="E2_0"
        intLanes="" shape="-1007.20,0.00 -1007.20,3.20 -1007.20,0.00"/>
```

For the edges we have the following definition. The ID gives the name, and from and to describe the begin and the end note of this edge. Priority gives the importance of the road, but that is something we not use here. After the edge is defined, the lane is defined. If the index is zero, then this is the lane which is the most right. Speed is the maximum speed allowed on the road, and the length gives the total length of a lane on this edge. The shape gives the coördinates of the begin and end point of the lane.

```
<edge id="E1" from="N1" to="N5" priority="-1">
<lane id="E1_0" index="0" speed="13.89" length="1000.00"
        shape="-1007.20,-1.60 -7.20,-1.60"/>
</edge>
```

The connections will be included in the files as follows. There are connections between lanes outside the intersection, and connections with lanes that are in the intersection.

```
<connection from="E3" to="E2" fromLane="0" toLane="0" via=":N5_0_0" tl="N5"
        linkIndex="0" dir="r" state="O"/>
```

So first the connections between the edges are made, and then the specific lanes are mentioned. Since there is an intersection between the lanes, there is also the via part, which is the connections within the intersection. The index is just the number of the connection, and the direction tells that this is a right turn. The state says "O" which stands for controller of. These states are mostly use to determine when a certain lane in the intersection gets green, yellow or red.

At last we have the part of the traffic lights. In our case we use cycle periods. From the starting point, the system starts with the first phase for 42 seconds, and goes through the cycle with the given time. The states is a sequence in the order of the linkIndex in the connection part. In the example above, the index was 0, and that corresponds with the first index in de states below. The means that this connection has first green for 42 seconds, and then goes to yellow for 3 seconds, and it ends with red for 45 seconds.

```
<tlLogic id="N5" type="static" programID="0" offset="0">
<phase duration="42" state="GGgrrrGGgrrr"/>
<phase duration="3"  state="yyyrrryyyrrr"/>
<phase duration="42" state="rrrGGgrrrGGg"/>
<phase duration="3"  state="rrryyyrrryyy"/>
</tlLogic>
```

Next to cycle times, it is also possible to use other traffic light programs which are not cyclic. Some of these programs are based on traffic models, and is better comparable with smart traffic. These are eventually more interesting to use than cycle times. But since we are now more interested in estimating positions, it is much easier to use cycle times now. Other programs that are possible are for example switching the lights when the

gap between vehicles is bigger than a certain value. Another way is to use the time loss of a vehicle. If a vehicle has lost at least t time, than the traffic light will be green a little bit longer than it already was. For both models a minimum and maximum value is used for the time that a phase should take.

Vehicles and routes
After defining the network we have to define the vehicles and the routes that these vehicles will drive. This is the part that is the most interesting to change during the simulation. For example we can choose for multiple types of vehicles in the simulation, or adding more vehicles into the system. Also speed, acceleration and deceleration is something we can change in this file. The first thing we make is a vehicle type distribution. Here is an example:

```
<vTypeDistribution id="passenger">
<vType id="passenger#1" probability="1.0" width="1.9" length="5.0"
        maxSpeed="13.89" guiShape="passenger" tau="2" minGap="2.5" accel="3.0"
        decel="6.0"  sigma="0.5" height="1.73" color="1,1,0"/>
</vTypeDistribution>
```

In this example, there is only one type of vehicle in this distribution, since probability is set to 1.0. The id of this type "passenger#1". The width en length give the shape of the vehicle. Also the guiShape, the height and the colour are used to make it visible in SUMO. The maxSpeed gives the maximum speed that this type of vehicle will drive maximally in meters per second. The value of "tau" gives the desired time headway. The car-following model that we use, defines this as the time distance between the back of the leader and the front of the follower. The mingap is the minimum distance between the back of the leader and the front of the follower. The accel parameter give the maximum acceleration rate in m/s^2, and decel does the same for the deceleration. The sigma stands for driver imperfection, and in this car-following model, a value of 0.5 is the default value. Note that we can make multiple vehicle distributions. Each flow can have a new distribution.

Now that the vehicle distribution is described, we can describe the route distribution. An example can be find below.

```
<routes>
<routeDistribution id="routedist1">
<route id="route01" edges="E1 E6" probability="1.0"/>
</routeDistribution>
</routes>
```

Each routedistribution has its own ID and can consist of multiple routes. Also these routes have its own ID and probability. The route is given by a sequence of edges that the vehicles will drive. In this example it will start at the begin position of edge E1, and will leave at the end of E6. In the case that an edge has multiple lanes, then a lane-changing model is used. But this is something we do not take into account.

The last thing to do is to create the vehicles. This is a combination of the vehicle distribution and the route distribution. We define flows by the following:

```
<vehicles>
<flow id="flow"  begin="0" end= "5000" probability="0.1" route="routedist1"
        departSpeed="max" type="passenger"/>
</vehicles>
```

Here we defined one flow from time step 0 to 5000, which is the begin and end time of the simulation. At each time step there will be determined whether a new vehicle should enter the system based on the probability. The type and the properties of the new vehicle will be determined by the type, which is the vehicle type generated by the vehicle distribution "passenger" we described above. The route that the new vehicle will drive is generated by the route distribution "routedist1". The parameter departSpeed determines which at which speed it enters on the first lane. If it is set to "max" then it tries to go to the vehicles maximum speed as long as it is safe with respect to the leader.

The vehicle type distribution, the route distribution and the vehicles together form a ".rou.xml" file. After finishing this part we have to define all other parts of the network in additional files.

Additional files
In the additional files, there are many things that can be put there, for example several types of sensors, re-routers, variable speed signs and bus stops. In our simulation we are going to use Instantaneous Induction Loops Detectors. These may not be used in practice, but to get accurate data that we can work with, it is the most convenient type of detector. The advantage is that these detectors only write data to a file at the moment

that there is a vehicle entering the sensor. Also the current speed is given, and the moment of leaving. Other detectors work with time steps and occupancy. If we use them, then we can only say that a vehicle arrived between time 5 and 6 for example. And with the Instant Induction Loops Detector we get the exact time. The additional file looks as follows:

```
<additional xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/additional_file.xsd">
<instantInductionLoop id="e1_100" lane="E1_0" pos="100" file="e1test.xml"/>
<instantInductionLoop id="e1_900" lane="E1_0" pos="900" file="e1test.xml"/>
<instantInductionLoop id="e1_990" lane="E1_0" pos="990" file="e1test.xml"/>
<instantInductionLoop id="e1_998.5" lane="E1_0" pos="998.5" file="e1test.xml"/>
</additional>
```

For example we created here 4 detectors. Each has its own ID, and the lane is specified in combination with the position on that lane. The information from the detectors is the input for the simulation, and therefore we need to write them to a file. The easiest way, if the simulation is not too big, is to write everything to the same file. In our simulation we use a second additional file which writes the states of the traffic lights to another file. For each additional you make, you need to create another file.

Configuration file
The last file we need is a configuration file, which is given below.

```
<configuration>
<input>
<net-file value="test5.net.xml"/>
<route-files value="test5.rou.xml"/>
<additional-files value="test5.add.xml,test5_1.add.xml"/>
</input>
<time>
<begin value="0"/>
<end value="5000"/>
</time>
<output>
<netstate-dump value="rawpositionsdump.xml" />
<emission-output value="emissionoutput.xml" />
<full-output value="fulloutput.xml" />
<vtk-output value="vtkoutput/vtkoutput.xml" />
<fcd-output value="fcdoutput.xml" />
<amitran-output value="amitranoutput.xml" />
<lanechange-output value="lanechangeoutput.xml" />
<queue-output value="queueoutput.xml" />
</output>
</configuration>
```

In this file we included the net, the route, and the additional files as input. Then the duration of the simulation is set at 5000 time steps. The last thing to do generate files with the output. In the above example multiple files are created. We do not use these files for our simulation, since we already created the detector output in the additional files. But these files can help analysing the behaviour of the vehicles in a more detailed way. The disadvantage is that in some files for each time step the vehicle is present in the system, there is data. So some files become very large.

4.2 Performed simulations

In this section we describe the variables we are using in each simulation. The first thing that has to be clear is that each vehicle in the simulation has the same properties. The implementation of the model until now is not build to handle different types of vehicles. The implementation is now done in R, but other probably worked better to simulate these types of models. Java for example could be used for object oriented programming, and that is probably better to work with since one can store all properties of the vehicles. In the table below, the parameters of each simulation can be found. The road network for each simulation is the same.

Simulation Number	**1**	**2**	**3**	**4**	**5**
Duration simulation	5000 s	5000 s	5000 s	5000 s	5000 s
Probability arriving vehicle	0.1	0.1	0.15	0.1	0.1
Road length	1000 m	1000 m	1000 m	1000 m	1000 m
Sensor position	100 & 900m	100 & 900m	100 & 900m	100 & 900m	100 & 900m
Acceleration	3 m/s^2	1 m/s^2	3 m/s^2	3 m/s^2	3 m/s^2
Deceleration	6 m/s^2	2 m/s^2	6 m/s^2	6 m/s^2	6 m/s^2
Max speed	13.89 m/s	13.89 m/s	13.89 m/s	22.22 m/s	13.89 m/s
Minimum distance	2.5 m	2.5 m	2.5 m	2.5 m	2.5 m
Minimum time distance	2s	2s	2s	2s	2s
Length vehicles	5 m	5 m	5 m	5 m	5 m
Duration green	20 s	20 s	20 s	20 s	30 s
Duration yellow	3 s	3 s	3 s	3 s	3 s
Duration red	23 s	23 s	23 s	23 s	33 s
Correction threshold	1 s	1 s	1 s	1 s	1 s

Chapter 5

Results

In this chapter the results of the simulations performed with SUMO and R are described. These results are mostly performed by pictures, and some quantities calculated by the model compared to SUMO. The pictures of SUMO contain points for each time step of the simulation, and it shows with detail the movements that the car have done during the simulation. In the pictures of the developed model, there are only a few points drawn. These points are the transition points between different events. These points are connected by line segments. The colour of the line segments describe the type of event. The events arrival and constant are coloured brown, deceleration is coloured red, and acceleration is blue. If the vehicle is waiting then the segments is black. Due to the scale of the pictures, the first car in the queue has a red waiting segment, since it is too close to the line that describes the state of the traffic lights.

5.1 Results Simulation 1

One of the first problems that occurred was the yellow light. In all these simulations, we use cycle times that are given to SUMO and the model. The simulation starts at the beginning of a green or a red period. Now apparently SUMO has a problem with interpreting these times. If for example at time t the light switches to green, then SUMO already starts at time $t-1$ with accelerating if the car was waiting. The same holds for traffic lights that switch from green to yellow. In SUMO the vehicles do not seem to drive through the yellow light. But also in this situation the vehicle starts decelerating even though the light is still green, and it could easily pass the intersection in a green light. So an uncertain point of all these simulations will be switching the lights. It will happen that a car in the model will pass the intersection, while SUMO lets it stop.

An example of the problem is shown in the figures 5.1. In the figure of the model, the last car during the green period passes the intersection and does not stop and form a queue. But SUMO is already decreasing the speed in the green period. Knowing this fact we adapted the cycle a little bit. So instead of starting with for example a green period of t seconds, we make it $t-1$ seconds. All other periods will be t seconds long, and we just shift one second in time. That there is a shift in traffic states time, is confirmed in a file that SUMO prints with the traffic light states. In this file the states change one second too early. Even though we correct this time shift, there are still moments that SUMO starts too early with accelerating, or with decelerating. Since it is something what SUMO does and the cause is not known, we have to deal with it and not focus on it. It is necessary to take this into account when discussing results of simulations, but adapting the model is not something we should do.

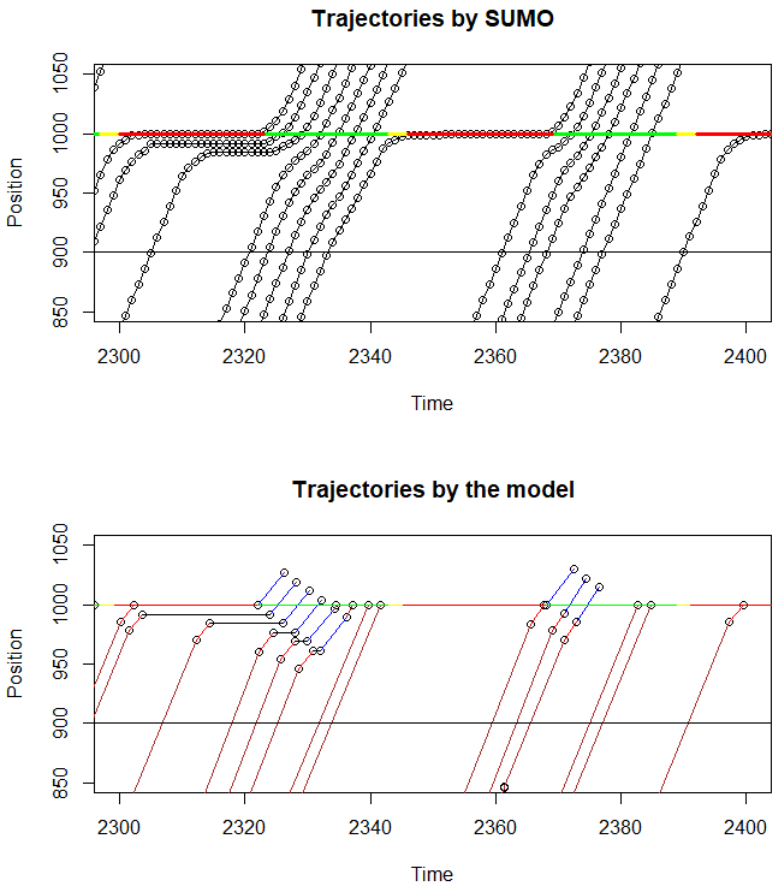


Figure 5.1: Problem of switching the lights

In figures 5.1 a part of the simulation is drawn. In total the simulation took 5000 seconds. In the part in the figure one can see that the moments of passing the intersection sometimes match the times that the vehicles pass in the SUMO model. Especially the vehicles that pass until the second red phase in the figures. From then the model predicts a car to pass while SUMO lets it wait. As a result, in the next green phase the vehicles pass earlier in the model than in SUMO and since the safe distance is two seconds, the results will tell that SUMO has a delay of 2 seconds for these cars. It is interesting to look at the differences of all cars. A density plot of the vehicles is given in figure 5.2

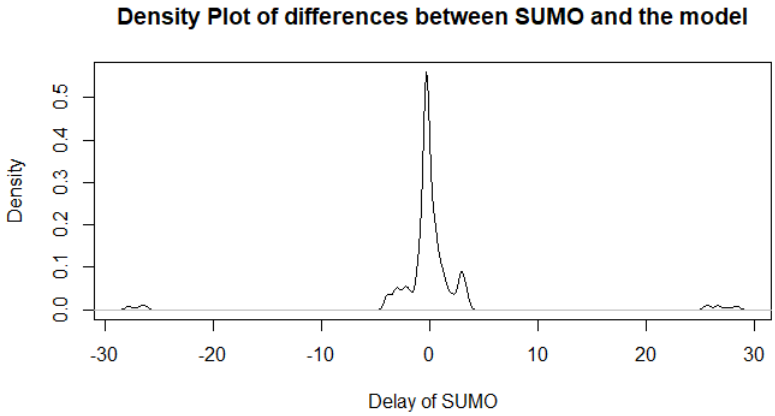


Figure 5.2: Density Plot of the differences between SUMO and the model

In figure 5.2 it is shown that the differences of most vehicles lay around the zero seconds. In fact the mean difference of all vehicles is -0.088223. So on average the model that is developed predicts passing the intersection a little bit earlier than SUMO does. But this simulation has a large standard deviation, which is 1.49861. This

deviation is too large to give good predictions. For example, if a margin is set on 1 second longer, then in many cases the vehicle still needs to stop. On the other hand, if the car is too early in the case Smart Traffic is used, then it has too make a stop, then do an acceleration, and it may cause extra delay for following cars. So the ideal situation would be that SUMO and the developed model only differ a certain threshold. A difference of 1 second is something that is acceptable. The frequencies can be found in

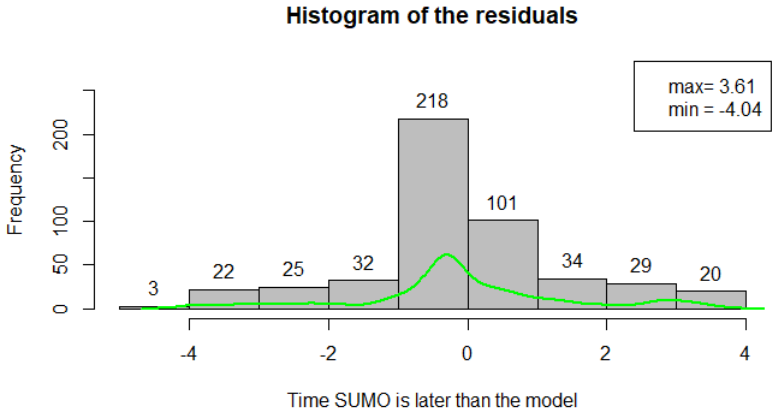


Figure 5.3: Frequencies of vehicles that depart in the same green period

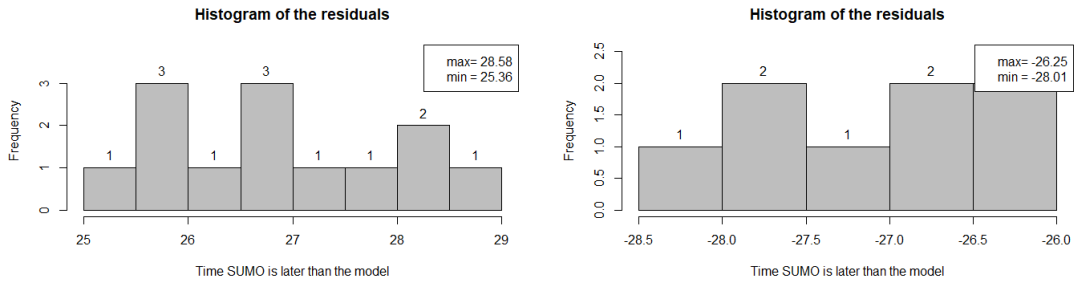


Figure 5.4: Frequencies of vehicles that depart a green period earlier or later

The residuals of the delay of the vehicles is dependent. So it is not a probability density function for which you can draw the delay with a certain probability. Therefore it is interesting to look at the series and to see if there are any patterns. A part of the entire simulation is plotted in figure 5.5.

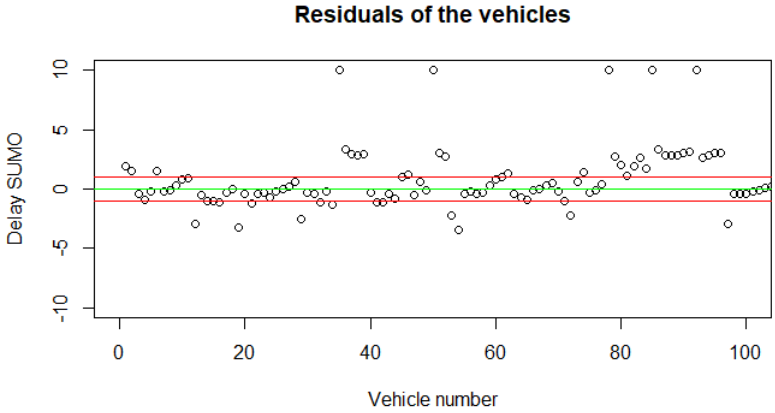


Figure 5.5: Time series of the residuals

First note that the highest value is not ten, but only plotted as ten to make the plot more clear. The actual values lay between 25 and 30 in positive and negative positions. The red horizontal lines are the critical value

of a 1 second difference. The first impression is that most values lay between these critical values except in the beginning there are three values that differ. The main thing that is remarkable is that when SUMO let a car wait while the model let it pass, that there are several cars following that have a higher delay than the critical value. As long as cars join the queue, they are all influenced by the delay of the first car in the queue. So between 80 and somewhere before 100 we see that it happens three times that a car has to wait an extra cycle. The last two have to do is as consequences of the first one. All the other cars that are driving in this period have about the same delay. So if the car around number 80 had not stopped, then all the other values would have been more around the critical value. This also causes the wide range of figure 5.3, and it causes the large standard deviation.

Another problem is that until now we only used the sensor in the beginning, and use that speed. But since the distance is large, anything could have happened between the starting point and the intersection. Therefore we now look at the model where we correct at a second point. This second point is the place where the second sensor is, at position 900. To show the difference, all the three plots are below each other.

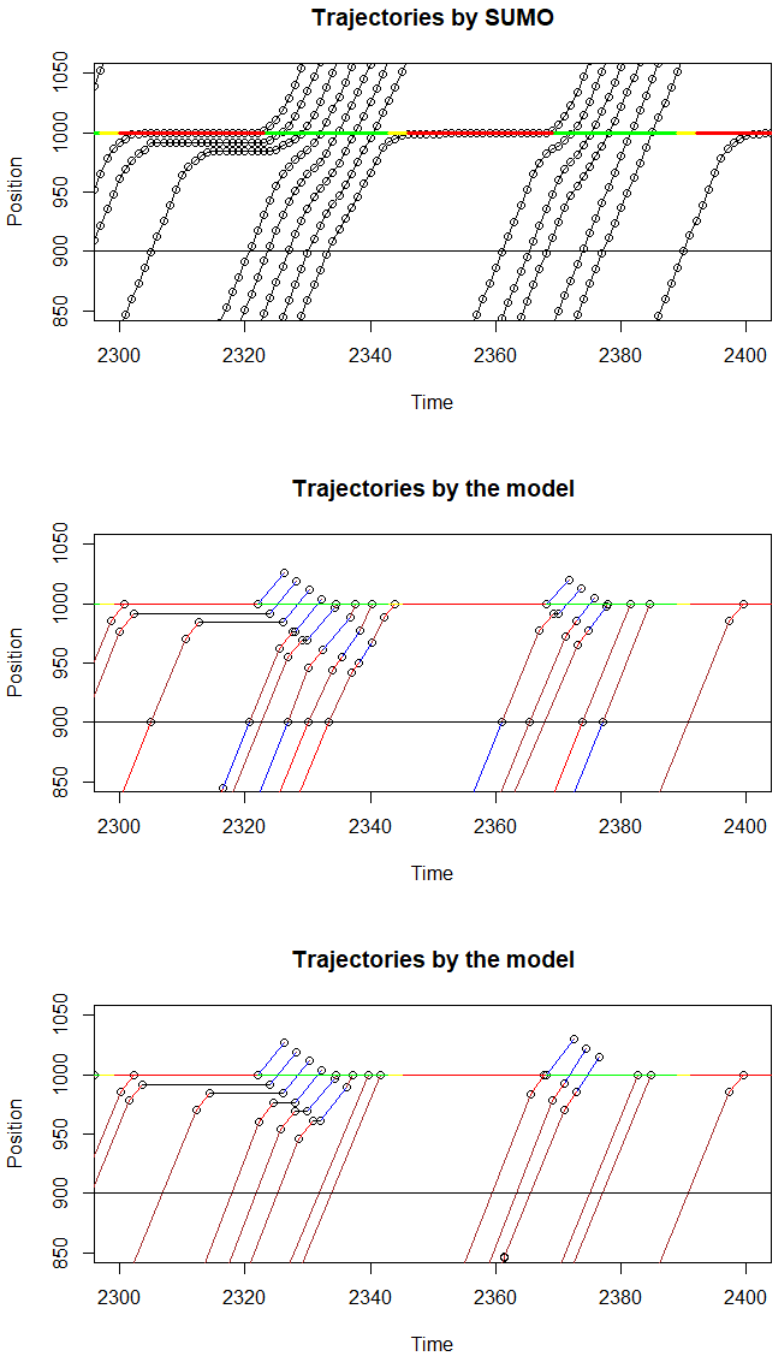
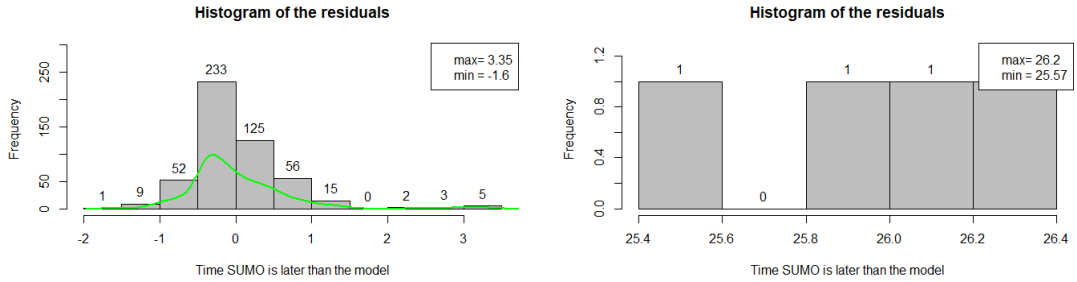


Figure 5.6: Problem of switching the lights, including the adapted model

In this part of simulation, a correction was made if the model predicted one second difference with the sensor data. The vehicles that are corrected have an extra dot in the picture. As soon a trajectory has changed, all following vehicles get also a correction. The main difference in this picture is that by the correction the vehicle does wait for the intersection just like the SUMO model. Apparently in the previous model we made a prediction in such way that the vehicle did not have to break and could drive with a faster speed. But in the corrected model it seems that the leaders arrive later than planned and that the followers have a delay because of this. In this situation we also look at the histograms of the residuals.



By using a correction 100 meters before the intersection, it was in this simulation never the case that SUMO let a vehicle pass a cycle earlier than the model. It happens 4 times that SUMO is a cycle later than the model. The mean of the difference between SUMO and the model is 0.0061, with a standard deviation of 0.646. So this means that SUMO is on average a little bit later than the model, although we included the outliers. But this standard deviation is already a lot smaller than the model without corrections. Next to that, the number of vehicles that differ more than 1,5 second decreased. The extremes in the first histogram are a result of the outliers in the second histogram. This is confirmed by the time series.

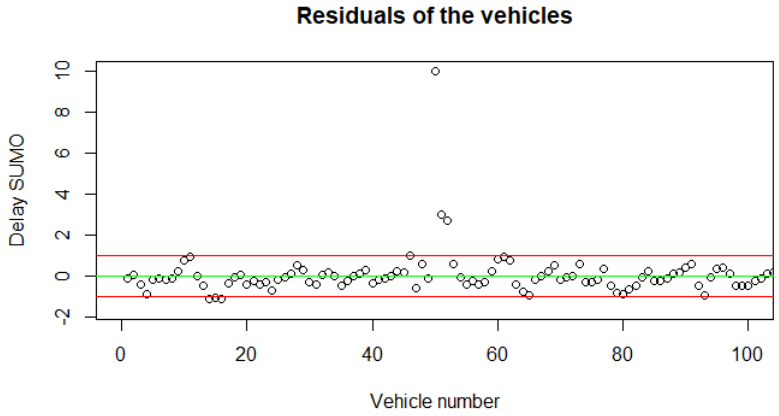


Figure 5.7: Time series of the residuals

As is seen in the figure 5.7 the outliers in the first histogram are caused by the vehicles in the second histogram. These values would have been between the critical values if SUMO had let pass the vehicles.

5.2 Results Simulation 2

In the first simulation, the corrected version of the model improved the original model. There were less cars that passed the intersection line in a different cycle if we compare the model to SUMO. Now in this simulation we kept the settings the same, but we changed the acceleration and deceleration rates. The acceleration an deceleration decreases in absolute value, making the trajectories less flexible. Again we compare the original model with the adapted model. A part of the simulation is showed in Figure 5.14.

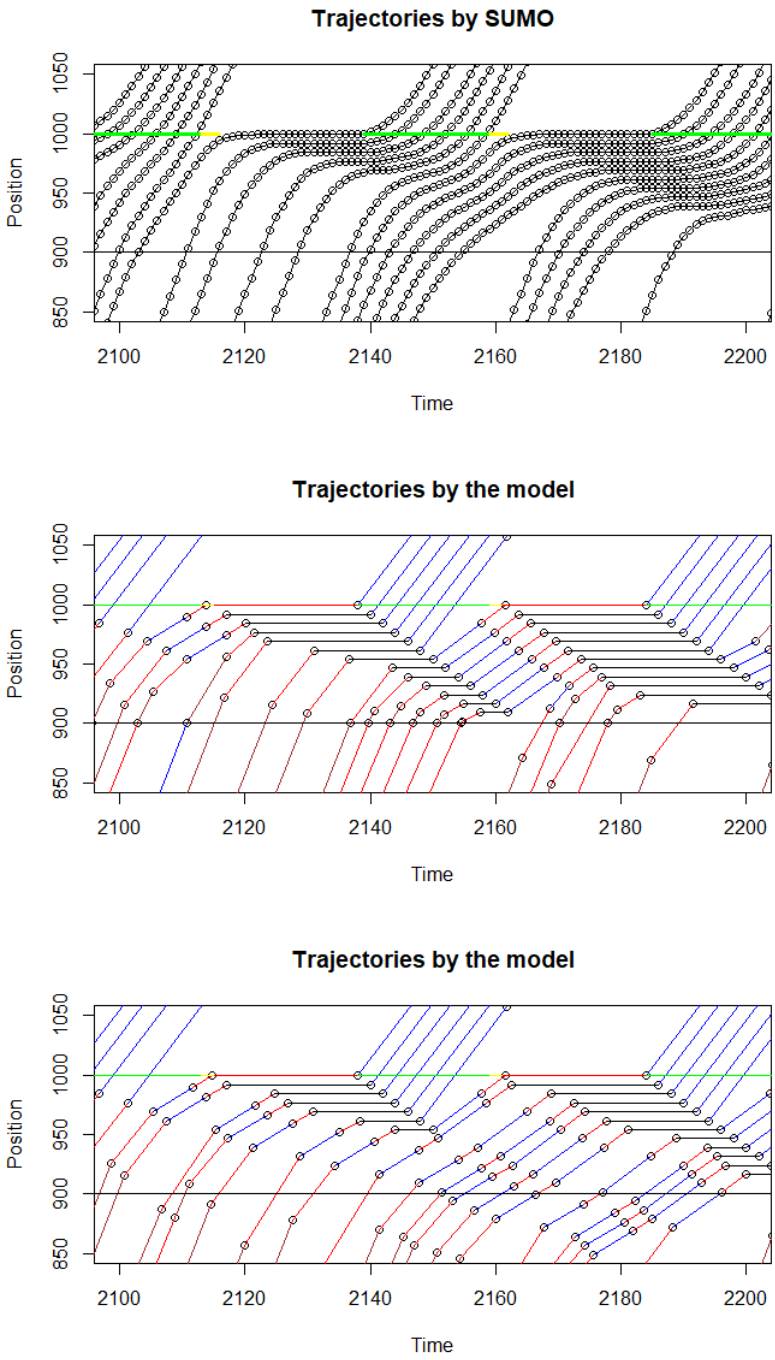


Figure 5.8: Changing acceleration and deceleration rate, 1) Sumo, 2) Adapted model, 3)

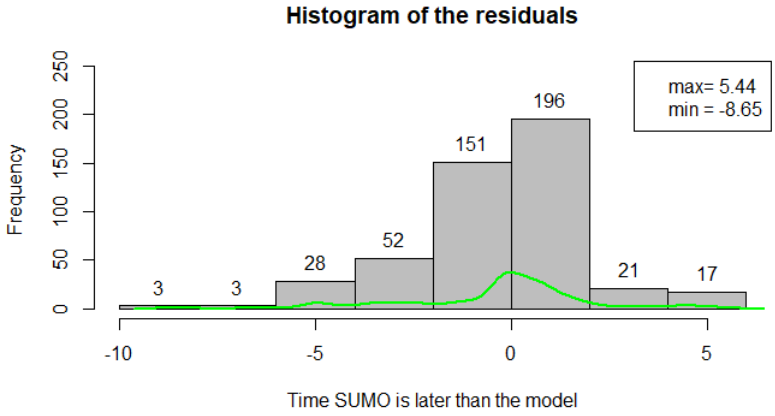


Figure 5.9: Frequencies of vehicles that depart in the same green period, basic model

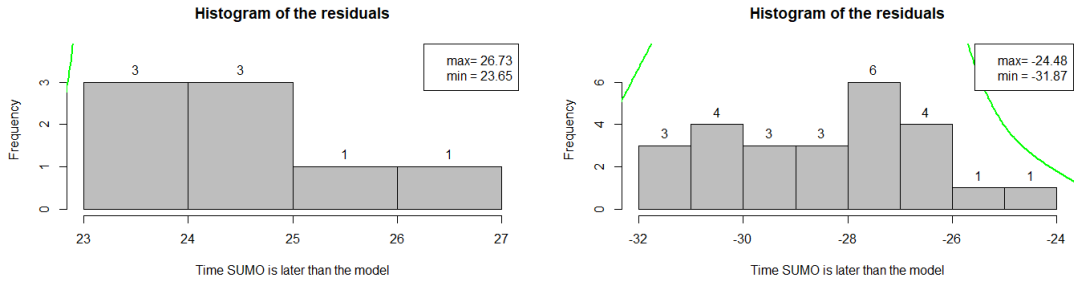


Figure 5.10: Frequencies of vehicles that depart a green period earlier or later, basic model

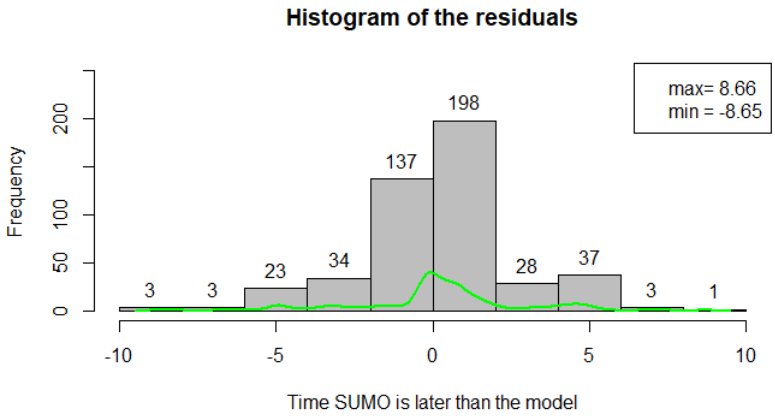


Figure 5.11: Frequencies of vehicles that depart in the same green period, adapted model

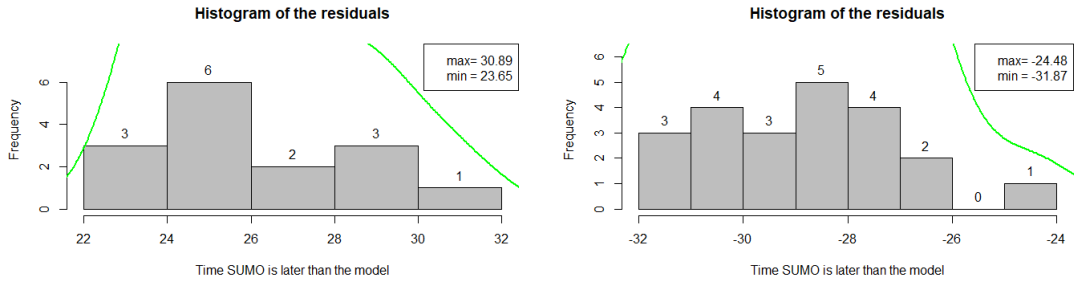


Figure 5.12: Frequencies of vehicles that depart a green period earlier or later, basic model

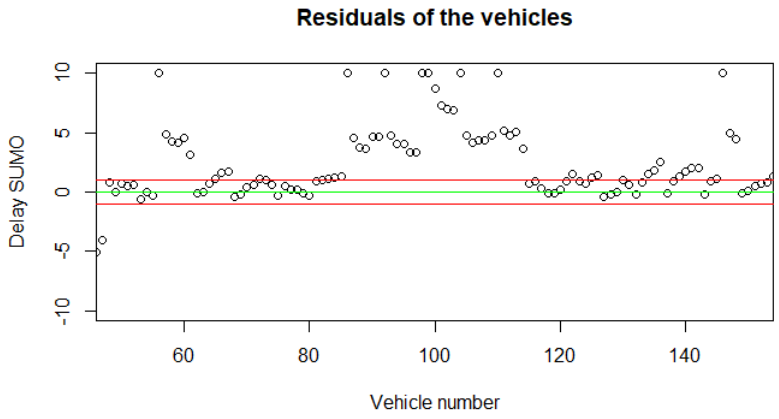


Figure 5.13: Time series of the residuals, adapted model

The mean of the basic model is -0.2694, and the standard deviation is 2.1817. For the adapted model the mean is 0.2342 and the standard deviation is 2.4018. The models do not show a lot of differences. The cause is the flexibility of the cars makes that SUMO and the model differ a lot. Again vehicles pass the intersection at different cycles. It occurs a lot that the vehicles pass the intersection in SUMO, but do not in the model. Compared to simulation 1, acceleration and deceleration closer to zero give worse results than higher values.

5.3 Results Simulation 3

In the third simulation we increased the number of vehicles. The probability of an arrival is increased from 0.1 to 0.15. Unfortunately there is a problem in the adapted model. The picture below shows a time period where the code does not get stuck. Although there is a big difference between the plots, it seems that this only is caused by the cars that have been passed earlier cycles. The model with only one sensor does not get stuck in the simulation, but the results are that the model is behind the SUMO model. The queues go far beyond the second sensor place, but the vehicles join the queue in a correct way, and they move with their leader. A problem of the adapted model is the speed at the second sensor. The cars drive ahead with that same speed, and the model is not good at determining if it should accelerate again. If there is a queue in front of this vehicle then there is no problem. But if all vehicles are moving again, then there are cases that the vehicle with the low speed keeps driving at that speed and causes a lot of delay for the vehicle itself and all the followers. The problem where the code does not run correctly is that at the moment sensor points are signalized, the model has other cars in front of this vehicle that has been signalized at the same position. This is caused by differences in passing the traffic lights. The adapted model says that these sensor points should be included in the model, but that means that vehicles passed their leaders which can not be the case in reality. Since the follower should still adapt the trajectory based on the leader, the trajectory goes backwards, or go back in time. Then the next vehicle gets problems with the collision function and gives errors.

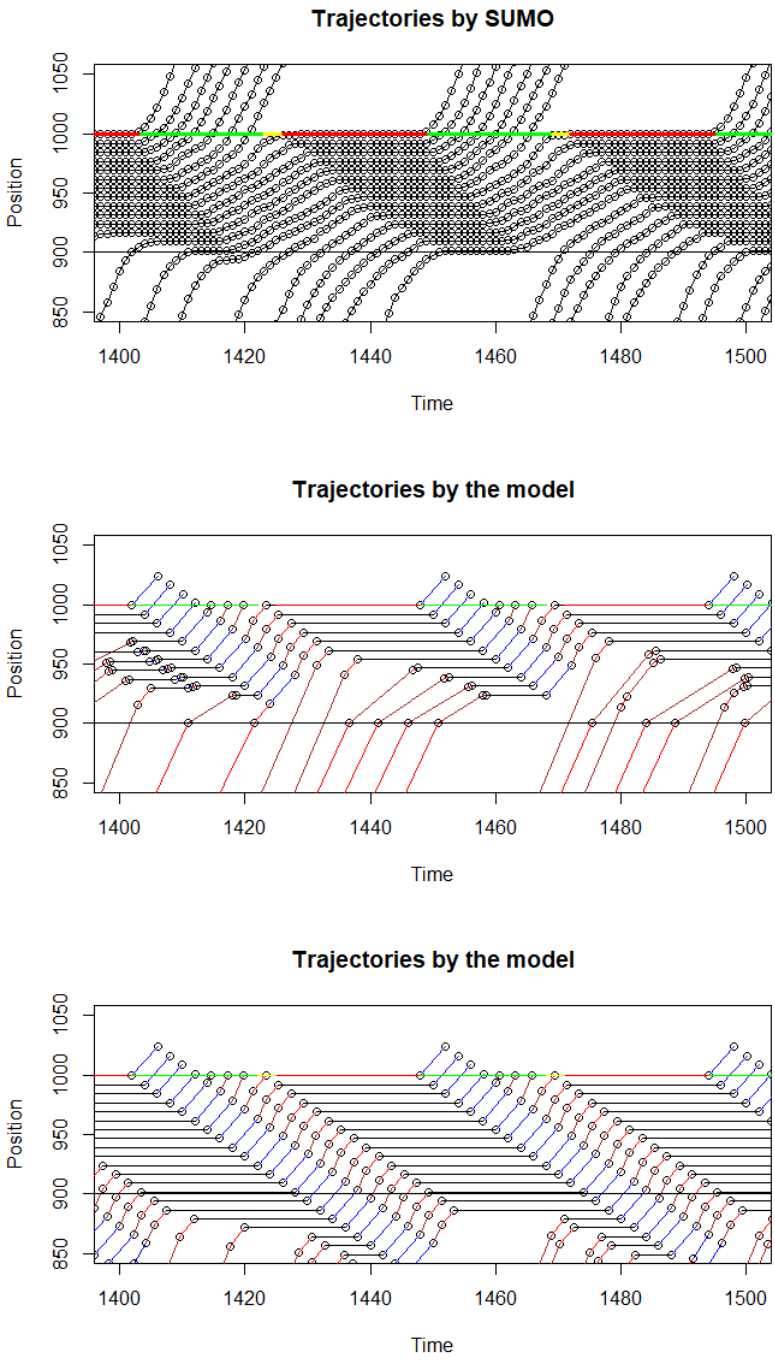


Figure 5.14: Increasing the number of vehicles, 1) Sumo, 2) Adapted model, 3) Basic model

Since the code does not run correctly, there are no results that make any sense to tell if the model performs well. Therefore only the results of the basic model is given. The mean of the residuals is -2.2138, and the standard deviation is 4.2.

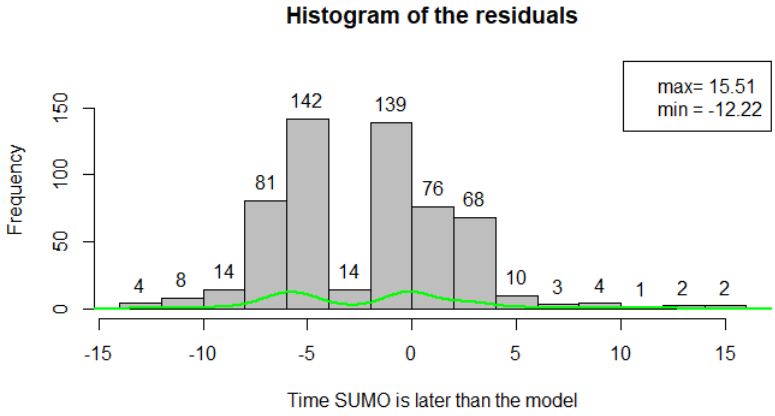


Figure 5.15: Frequencies of vehicles that depart in the same green period

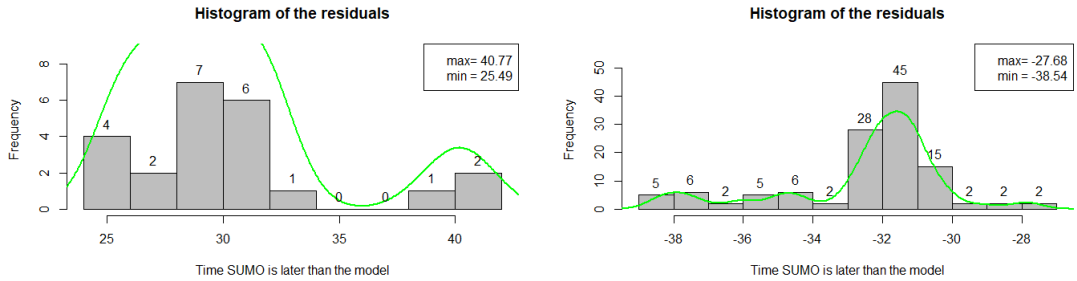


Figure 5.16: Frequencies of vehicles that depart a green period earlier or later

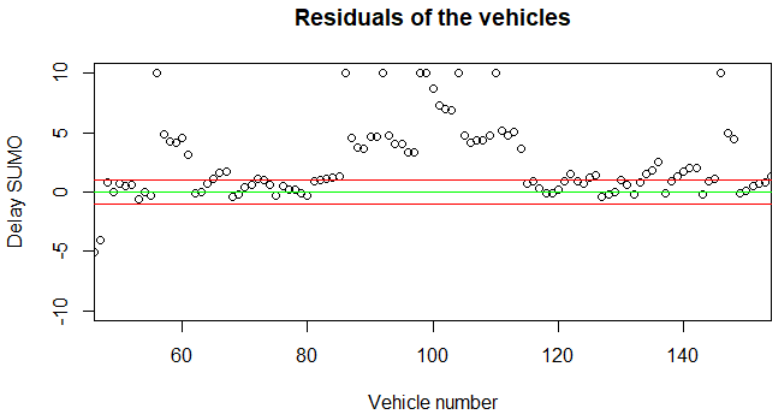


Figure 5.17: Time series of the residuals

The only difference with the first simulation is that the number of vehicles is increased. The probability increased from 0.1 to 0.15. This seems to be a small difference but for the model the results became bad, and for the adapted model even not possible. Their should be a lot of improvements done to make the model perform good even when vehicles are standing still at a sensor. Only the first model is not a good idea to determine the time vehicles should pass the intersection. A sensor at the intersection should be placed to determine if a vehicle did pass or stopped. Especially if the lights become orange it makes a big difference. That is the main results that many time differences are larger than 2 seconds between the model and SUMO.

5.4 Results Simulation 4

Another factor that could influence the model is the speed on the road. Until now each road had a maximum speed of 13.89 m/s, which is about 50 km/h. Now we will check what happens if the speed increases to 22.22 m/s (80 km/h).

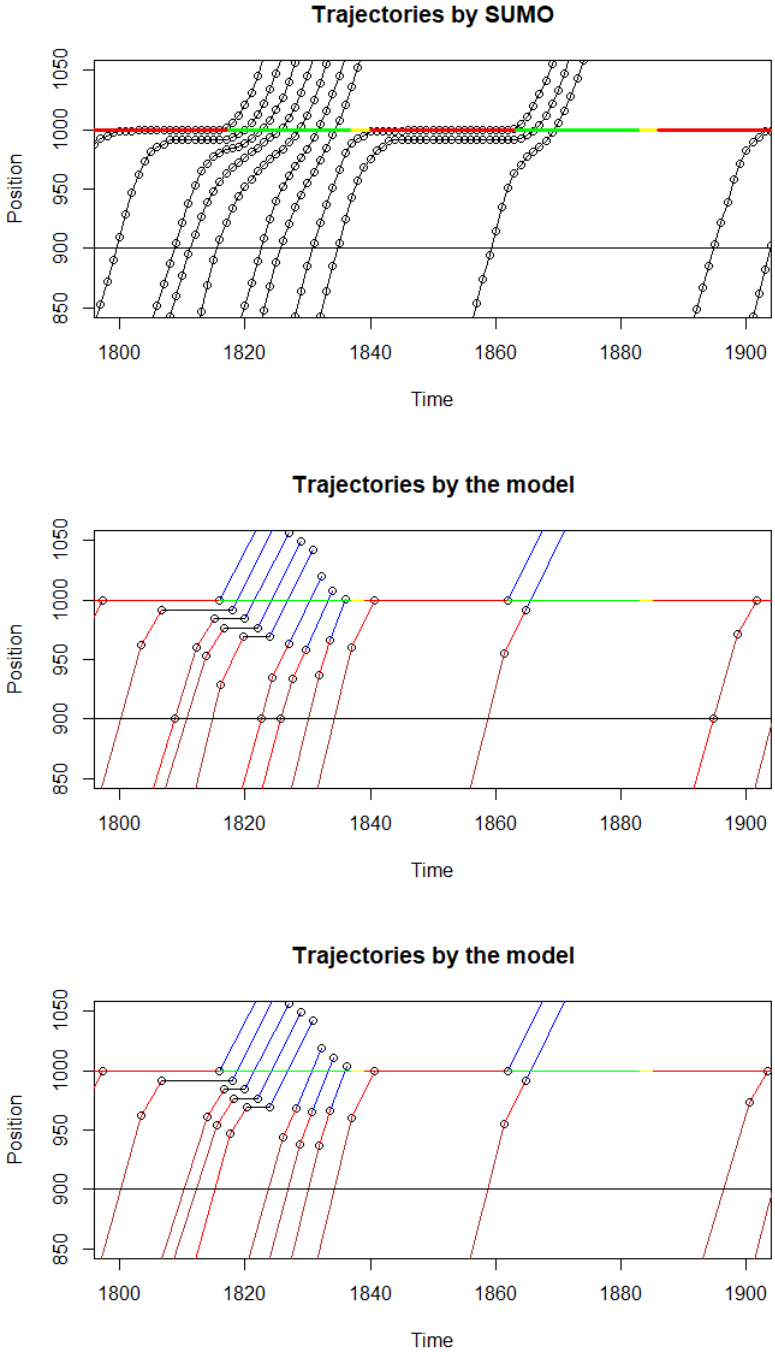


Figure 5.18: Increasing the number of vehicles, 1) Sumo, 2) Adapted model, 3) Basic model

The picture shows that the differences between the basic model and the adapted model are small. There are a few cars that have to be adapted, so the pictures look almost similar. If we look at the distribution of the residuals then the basic model has a mean difference of 0.0970, an standard deviation 1.2704. The adapted model has a mean difference of the adapted model is 0.3718, with standard deviation 1.0746. The results are shown in the pictures below.

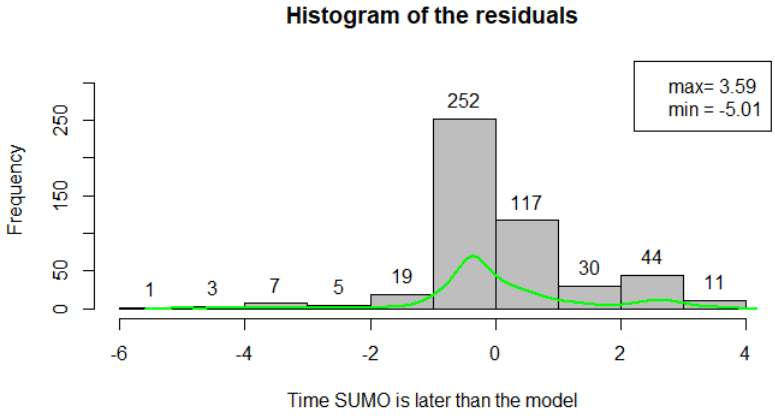


Figure 5.19: Frequencies of vehicles that depart in the same green period, basic model

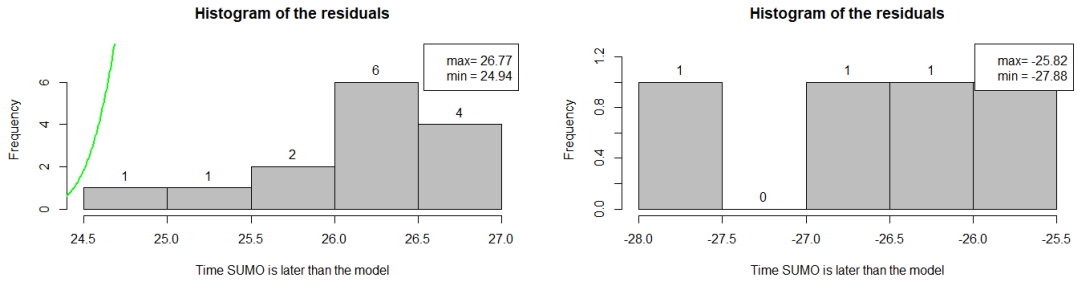


Figure 5.20: Frequencies of vehicles that depart a green period earlier or later, basic model

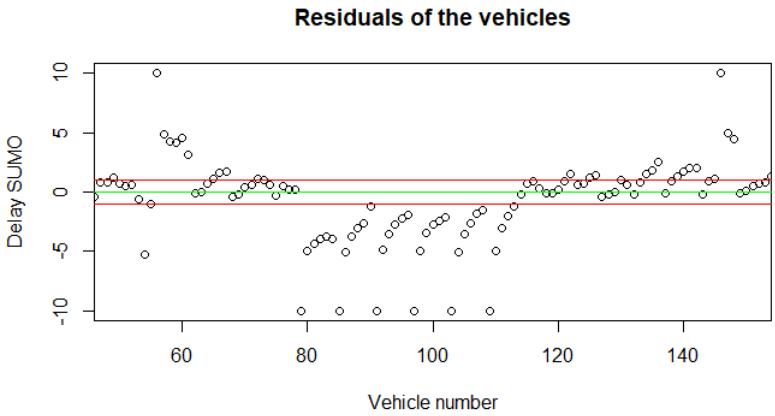


Figure 5.21: Time series of the residuals
9

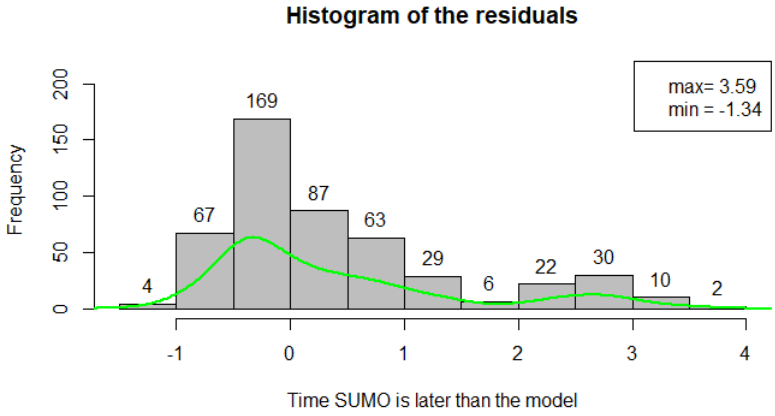


Figure 5.22: Frequencies of vehicles that depart in the same green period, adapted model

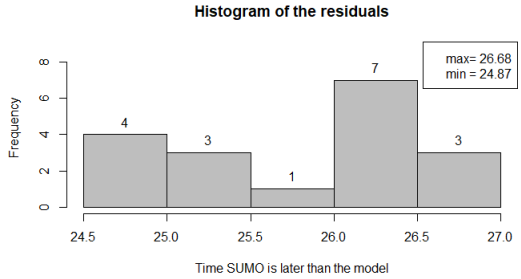


Figure 5.23: Frequencies of vehicles that depart a green period earlier or later, adapted model

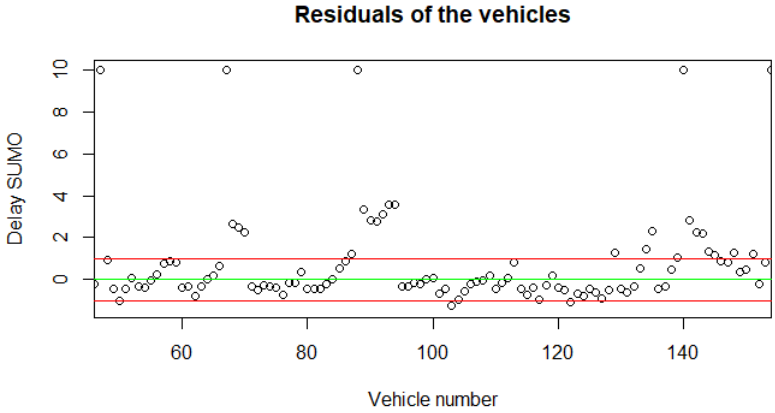


Figure 5.24: Time series of the residuals

In the case of high speed, it is not very clear what model works best. It is not necessarily the case that the adapted model improves the basic model a lot, but the results of both models are not very bad. Also here the main differences are caused by the switching of the traffic lights. If there was a correction then the results would be good since SUMO is mostly too late, which and small deviations of followers are caused by these.

5.5 Results Simulation 5

In this simulation, the only difference with simulation 1 is that we change the duration of the green periods. Vehicles may have to wait longer until the lights turn to green, but more vehicles can pass, and some of them do not have to break. In this simulation both models are implemented without errors. An example of the

differences are in figure 5.25. There it is visible that many corrections have been made, but it does not result in much change in passing the intersection. Of course this is only a part. The total results are found in the histograms and the time plot.

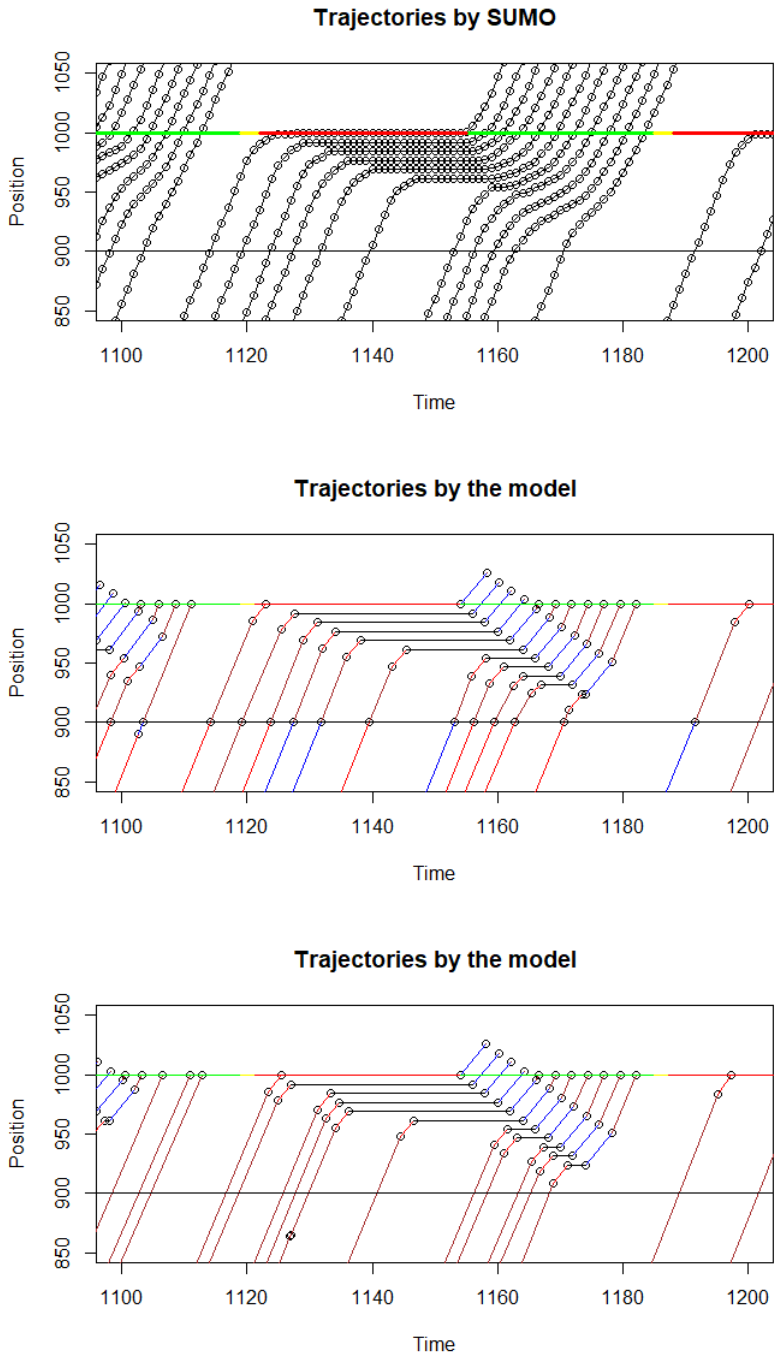


Figure 5.25: Increasing the number of vehicles, 1) Sumo, 2) Adapted model, 3) Basic model

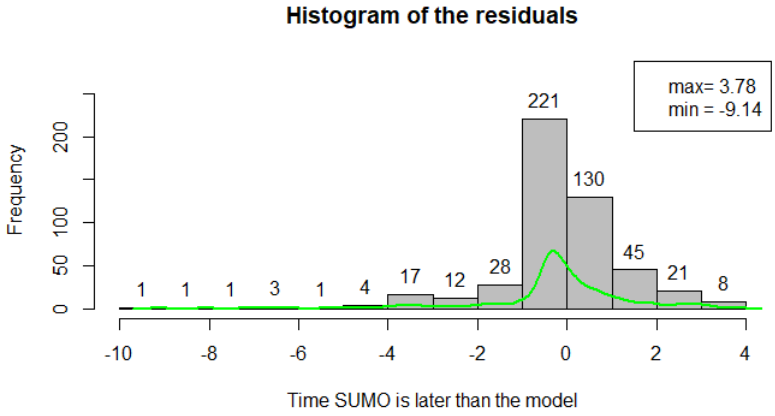


Figure 5.26: Frequencies of vehicles that depart in the same green period, basic model

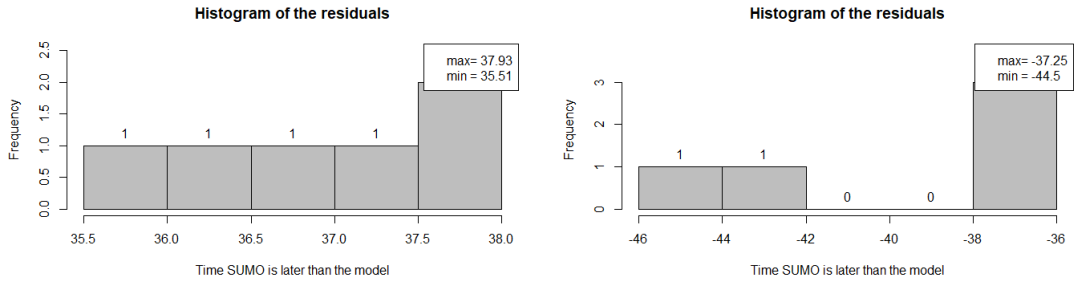


Figure 5.27: Frequencies of vehicles that depart a green period earlier or later, basic model

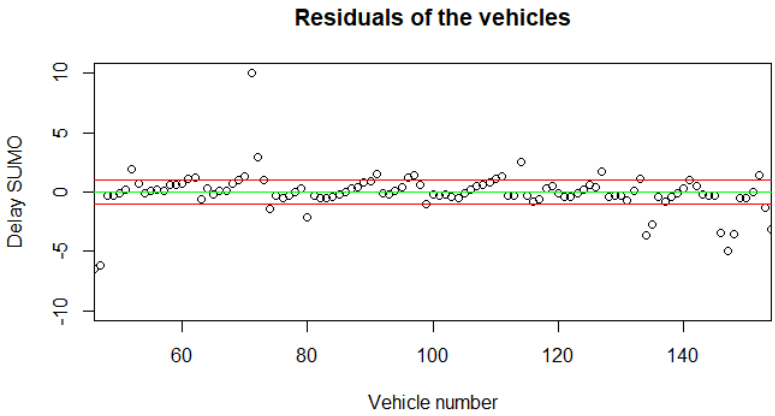


Figure 5.28: Time series of the residuals, basic model

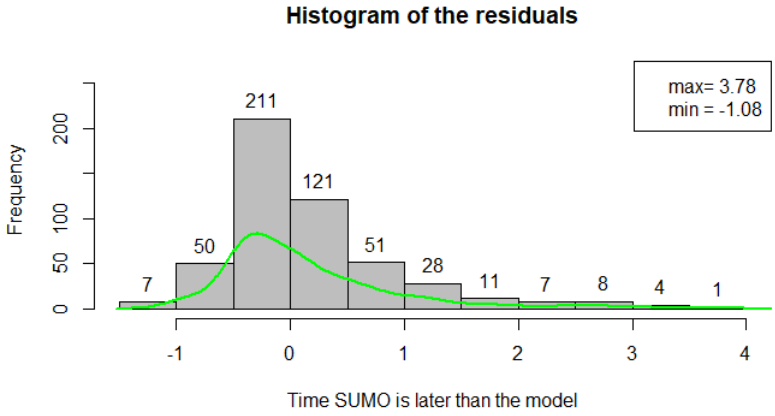


Figure 5.29: Frequencies of vehicles that depart in the same green period, adapted model

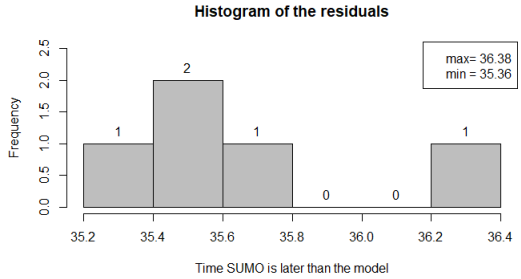


Figure 5.30: Frequencies of vehicles that depart a green period earlier or later, adapted model

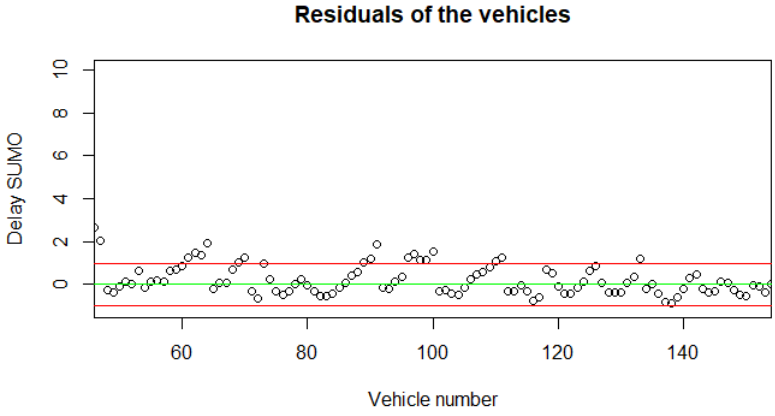


Figure 5.31: Time series of the residuals, adapted model

The basic model has a mean difference of -0.1762962 and a standard deviation of 1.5342. For the adapted model holds a mean of 0.1350, and standard deviation 0.7702. The mean changes only in sign, and the main difference between the models is that SUMO is not ahead of our model in the adapted case. There are less outliers with the adapted model, that is confirmed by the standard deviation, that is only the half of the basic model. In this case it never happened that the queue went behind the second sensor point. If that happens, results may be different. But as long as they have some speed at the second sensor point, The model performs well. Large differences are again caused by passing in the wrong cycle.

Chapter 6

Conclusion

In this paper the purpose was to estimate the trajectories of vehicles such that can be determined when the vehicles pass the intersection. The model has to be used real time, and should therefore be fast and not complex. The idea of building a model based on events should give only a few calculations since only transition points are taken into account.

Indeed this model is fast. When performing the simulations of the model, we simulated a time period of 5000 seconds and the results are available immediately. So if the trajectory of a new car has to be determined, then is also done quickly. But the complexity of the code of the model is something that does not make it attractive to make this type of model. The basic model is implementable for all situations. So with only one sensor it is possible to calculate a trajectory, starting at a far distance where the vehicles drive in free traffic. The problem with this model is the interaction between the vehicles. These interactions are based on a collision function, but when implementing the model there are many special situations that occur. The main problem with this model is that these problems come one by one when simulating an entire data set. The model is based on studying the behaviour of each individual car, and the actions are based on what is the most logic thing to do. So for the first data set we used, the code was rather structured, but when using a new data set, there are situations that were not seen in the first set, and for each situation you have to investigate where in the code in goes wrong, and what part of the model is not right. After many data sets, there is model that calculates all trajectories, but there can still be situations that we have not seen yet. And since these adaptations in the code each times, it is not very nice to use any more. It makes the code messy and it expands quickly. Another problem is that the definition of the event types is not clear enough. The arrival event was mainly used when a vehicle entered the system and a collision should be found. But when the adapted model was introduced, vehicles get many more arrival events each time we have to check for a new collision. During the process it became hard to decide when the event arrival should be scheduled, and when constant.

Also since we used cycle times during all simulations, the model is very sensitive to orange periods. At the moment that the lights switch to orange, the model lets some vehicles pass, while sumo does not and vice versa. It would have been desirable to change the trajectories when the prediction was different than reality, like for example driving one green period earlier of later than predicted. Unfortunately, that part of the model is not completely finished. The second sensor close to the intersection caused some problems when the traffic demand becomes higher. The main problem is that the model can not handle slow driving or stationary traffic on a sensor very well. The biggest problems occur when the model and reality have a different amount of cars waiting for the red light. The model says that the sensor data are the main guidelines, and from there on they follow the predecessor. But then conflicting situations can occur, where the car will drive backwards since the predecessor is still too close too the sensor. The other assumptions as time distance and space distance cause these problems. Consequently the collision function for the next vehicles do not work since cars are driving backwards with a negative speed. At this point the code gives errors, and no result is given for all following cars.

The behaviour of acceleration and deceleration also influenced the outcome of the model. In the most simulations we used parameter values in such way that vehicles accelerate and decelerate at a high value. But when we decrease these values, it makes the model less flexible. A difference between the developed model and the car-following model in SUMO is that the new model lets the vehicle stop more often than SUMO does. SUMO lets cars decrease speed earlier, but at a lower rate. The advantage is that the following principle is more smooth, but more complex the describe. In the new model the decreasing and accelerating parts are one curve, but often they had to start from 0. Since accelerating is not that quick, the problem with switching the lights plays a role more often, and as a consequence the number of vehicles differ during the entire process. The difference in departure times are therefore large and not fixable with a second sensor.

Changing the maximum speed on the road does not really influence the performance of the model. These results are quit similar and the adapted model gives a little improvement on the basic model. Also a change in the duration of a green period worked fine. But a change in duration of green periods influences the number of vehicles before the intersection. And in the case of the simulation it never happened that vehicles drive slow on the sensor or are waiting there.

To conclude, this paper tried to make a model that calculate the trajectories of vehicles towards traffic lights in a fast way. Although all the formulas are not too complicated to understand and to perform, it may not be the easiest model to implement. During the process too many different situations occur, where the assumptions could not entirely be satisfied. For example, the part where events have to be removed is some part that still has issues. It may be better to store more information than we now did to make more exact trajectories. Less information will be lost, but this means that the entire model should be revised. The problem that now occurred is that the model is build based on a starting data set, and during the process it expanded in an undesired way. It would be better to totally start over and build the model again. But for this period of the final project, there was not enough time to make an entirely new model.

Chapter 7

Future research

There are several suggestions for future research. One of the main problems is how to deal with multiple information points of a trajectory and how to adapt the trajectory based on these data points. If these points are necessary for the model, then it may not be a good idea to use an event based model, since there will be too many curves that have to be calculated between these points that makes it more complicated than it should be. The multiple sensors give many problems with assumptions that are made before. It is clear that assumptions are not always right, and they should be changed if they are unrealistic. But in this case the problem is not caused by bad assumptions. It is causes by the fact that some trajectories are based entirely on the model, and others are corrected by sensor data. At the point that this happens assumptions are violated many times. Maximum acceleration rates and speed can raise above their maximum in order to maintain a safe distance as long as possible. So the research should be done on how to combine a model with corrections during the process.

What the model certainly needs is a control for vehicles that passed the traffic lights, or did not pass the traffic lights. The model sometimes fails when there is a different amount of vehicles waiting in the model than there is in reality. One of the reasons that we did not include this is that in reality there are no sensors at the intersection itself. So in practice it is not possible to determine most of the times. It would be unrealistic to present a model that had the right number of vehicles, but in practically impossible. So to get optimal results it is needed to find a solution that is practically executable.

Furthermore it would be nice to test this model with practical data. In this paper we used simulated data that is also based on a model. Although these models are used for many practical situations, it may not always represent reality. If we could use actual data of such a road, we could better determine if this type of model is something that can be used to model trajectories. The main problem is that there is more randomness in the actual traffic, and it can give results that deviate a lot from the model. However, the car-following models could have the same problem.

Also other vehicle types we have not included in the model. The only difference that the vehicles had was their speed. And these speeds differed not much from each other. It becomes more interesting when there are for example tractors on the road. Including other vehicle types on the road, includes also different distant gaps to drive safe, and other acceleration and deceleration rates. All these components should actually be investigated. In this model we used only one type of safe distance, but that can also depend on the traffic situation. On a highway the distance in time may be larger than on a road where there is a very low maximum speed. How to use the right safety distance should therefore be investigated.

Bibliography

P S Bokare and A K Maurya. Acceleration-Deceleration Behaviour of Various Vehicle Types. *Transportation Research Procedia*, 25:4733–4749, 2017. ISSN 2352-1465. doi: 10.1016/j.trpro.2017.05.486. URL `http://dx.doi.org/10.1016/j.trpro.2017.05.486`.

Serge P Hoogendoorn. State-of-the-art of Vehicular Traffic Flow Modelling. Technical report, 2001.

Stefan Krauss. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. Technical report, 1998.

Stefan Lämmer and Dirk Helbing. flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, (4), 2008. doi: 10.1088/1742-5468/2008/04/P04019.

Harwin Saptoadi. Suitable Deceleration Rates for Environmental Friendly City Driving. *International Journal of Research in Chemical, Metallurgical and Civil Engineering*, 4(1):2–5, 2017.

SWOV. Headway times and road safety. Technical Report December, 2012.

Bas van der Bijl and Jeroen Brouwer. Smart Traffic : Intelligent Traffic Light Control. Technical report, SWECO.