ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

The influence of a speed limit reduction on a traffic jam

Martin Wermelinger & Lukas Stadelmann

Zurich
Dec 2012

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Martin Wermelinger Lukas Stadelmann

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

The influence of a speed limit reduction on a traffic jam

is original work which I alone have authored and which is written in my own words.*

Author(s)

Last name	First name
Wermelinger	Martin
Stadelmann	Lukas

Supervising lecturer

Last name	First name
Balietti	Stefano
Donnay	Karsten

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zurich, Dec 12
Place and date

Signature

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

Contents

1 Abstract

We modeled a section of a fictitious busy two lane highway with lane changing. The used driver model is the "intelligent driver model"[1] which we extended with our own lane changing model. We analyzed the influence of a speed limit reduction on the traffic flow in two cases. Once with and once without a specific event that worsens the traffic flow.

2 Individual contributions

During the programming phase of our model the work was not definitely distributed. We either programmed together or partially for one's own. But both were always informed about the status quo of the program. The writing of the report was split. Martin Wermelinger evaluated the simulation results and Lukas Stadelmann made the description of the model and the implementation.

3 Introduction and motivations

Traffic congestion is getting more and more a problem. The number of vehicles is increasing every day and the time wasted stuck in traffic is really annoying. There's a lot of effort to prevent traffic problems. One notice on swiss highways a speed limit reduction when the traffic is overloaded.

We want to model the effect of the speed limit reduction and see if this method improves the traffic flow. For this purpose we put the following questions:

Case 1 There's no specific event causing a traffic jam. We study the influence of the speed limit on self-induced stop-and-go traffic or even traffic jams that can suddenly appear on crowded highways. What's the ideal speed limit that allows a high traffic flow and at the same time prevents these phenomena?

Case 2 We cause a traffic congestion by a specific event. Once by a sudden temporary slow down of the cars due to the sun which blinds the drivers and once by an accident on a lane. We now reduce the speed limit at a certain point some kilometers ahead the disturbance. Is it possible to improve the traffic flow with this method or is the traffic congestion just displaced? Where is the ideal point for this speed intervention and which is the appropriate speed limit?

Because this method of reducing the speed limit is actually in use, we hope our model can prove a positive effect on the traffic flow.

Case 1 We expect that if there is a high traffic density, one can improve the traffic flow with a lower speed limit (but not too low!).

Case 2 We estimate there will be a specific place ahead the traffic congestion, where a speed reduction has a positive effect on preventing or clearing a traffic jam.

4 Description of the model

4.1 Intelligent Driver Model

We used the "intelligent driver model"[1] for the description of the vehicle behavior. This continuous and deterministic model calculates the accelerations of the vehicles dependent on the gap and on the velocity difference to the vehicle ahead. The acceleration of a vehicle α of a length l_α at position $x_\alpha(t)$ with velocity $v_\alpha(t)$ is determined as follows:

$$\dot{v}_\alpha = a \left[1 - \left(\frac{v_\alpha}{v_0} \right)^\delta - \left(\frac{s^*}{s_\alpha} \right)^2 \right]$$

with the maximum acceleration a, the desired velocity v_0, the gap $s_\alpha = [x_{\alpha-1}(t) - x_\alpha(t) - l_\alpha]$ and the velocity difference $\Delta v_\alpha(t) = [v_\alpha(t) - v_{\alpha-1}(t)]$ to the vehicle (α-1). The exponent δ describes the acceleration behavior and the effective desired distance s^* is calculated with the equation:

$$s^* = s_0 + \max \left(v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}, 0 \right)$$

with the safe time headway T and the jam distance s_0. The drivers keep the jam distance to a standing vehicle in a traffic jam. In the moving traffic they keep an additional safety distance $v_\alpha T$. This formula has an intelligent braking strategy included. In normal situations the drivers slow down with the deceleration b, but they brake harder in dangerous situations.

4.2 Extensions

4.2.1 Trucks and cars

Our model of the highway includes cars and trucks. The average desired velocity of the cars is set to the speed limit and the average desired velocity of the trucks is 80 km/h. The trucks are longer, have a smaller acceleration and a higher safe time headway.

4.2.2 Lane changing

Because we modeled a two lane highway with trucks and cars there will be a faster and a slower lane. So the dynamics are different whether the vehicles can change the lane or not. So we implemented a lane changing algorithm. The cars change the lane if several conditions are fulfilled. These conditions depend on the traffic situation on both lanes. The trucks must not change, they only drive on the right lane.

4.2.3 Disturbance and speed limit

In case 2 we wanted to cause a traffic congestion by a specific event. We modeled two events:

- The sun blinds the drivers during the sunset on a certain section of the highway, so the disturbance is induced by a reduction of the desired velocity on this section during the time the sun blinds.
- There's a hard braking of some vehicles due to an accident on a lane. The disturbance is induced by a intense reduction of the velocity in the section with the accident.

To be able to analyze the influence of a speed limit reduction, the model has the option to limit the desired velocity of the vehicles to the speed limit in a certain section of the highway.

5 Implementation

5.1 Introduction

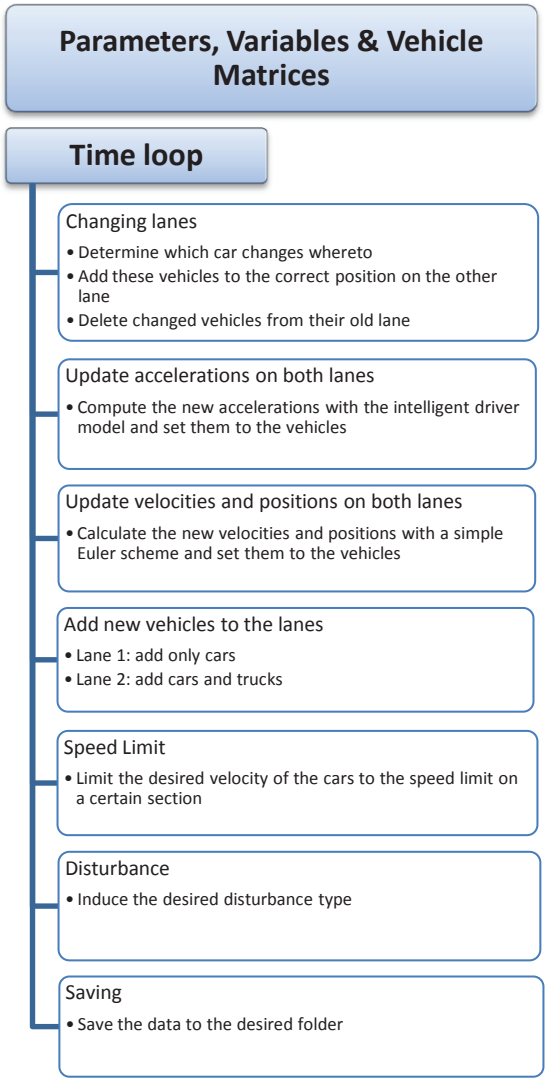

Figure 1: Basic program structure

Figure 1 shows the basic program structure of the function "`simulate_main.m`" (cf. appendix 9.1). At the beginning the parameters are defined and the variables initialized. The data of the vehicles is stored in two matrices. For each lane a matrix. The program is designed that this two matrices always stay well sorted, so it's easy to determine the vehicle ahead and behind of a specific vehicle. A vehicle

is deleted as soon as it leaves our section of the highway. So the matrices contain only the vehicles which we really need and the computing time is reduced. After the initialization comes the time loop which is build up of the following program components.

5.2 Time loop

5.2.1 Changing lanes

The changing lane component is split into two parts. One which changes the vehicles on the fast lane and one that changes the vehicles on the slower lane. Those parts are executed alternately so they will not disturb each other.

It firstly iterates over all vehicles and marks those which can change the lane by saving the positions whereto they will change. Only the cars are allowed to change, the trucks must not change the lane. There are plenty of asymmetric conditions which have to be fulfilled that a car is allowed to change the lane. They depend on the own desired velocity and on the gaps and velocities of the cars ahead and behind on both lanes. The most important conditions are mentioned in the following: A car changes the lane if his own desired velocity is higher than the velocity of the vehicle ahead on the same lane and if the velocity of the car ahead on the other lane is higher than the velocity of the car ahead on the same lane and if the gap to the vehicle ahead and behind on the other lane is larger than the effective desired distance s^*, which is identically calculated as in the "intelligent driver model"[1]. The cars also change from the fast lane to the slow lane if there's a huge gap on the slow lane even if their desired velocity is higher than the velocity of the vehicle ahead on the slower lane.

Afterwards the marked vehicles are one after another copied to the other vehicle matrix and then deleted from their old matrix.

5.2.2 Update accelerations on both lanes

The update of the accelerations is implemented with the "intelligent driver model"[1]. The following parameters were used:

	a	b	l	s_0	δ
Car	0.6 m/s^2	0.9 m/s^2	5 m	2 m	5
Truck	0.2 m/s^2	0.4 m/s^2	16 m	2 m	5

Table 1: Used parameters in the intelligent driver model[1]

This section of the program is executed only every tenth time step. We used a time discretization of Δt = 0.1 s. So the acceleration update takes place only once per second. In this way we modeled inattentions and the reaction time of the drivers. It now sometimes happens that the vehicles crash. Our model is in this case no longer accurate because the vehicle matrices are no longer well sorted which causes a inaccurate calculation of the accelerations. Therefore the program checks every time step whether the matrices are still well sorted or not. It displays a message and stops the execution if this problem appears.

5.2.3 Update velocities and positions on both lanes

These updates are implemented with a simple Euler scheme using a time discretization of Δt = 0.1 s.
The velocities are updated as follows:

$$v_\alpha(t + \Delta t) = v_\alpha(t) + a_\alpha(t)\Delta t$$

And the positions:

$$x_\alpha(t + \Delta t) = x_\alpha(t) + v_\alpha(t)\Delta t$$

5.2.4 Add new vehicles to the lanes

On the fast lane a new car is inserted if the gap to the vehicle ahead is larger than s^* and the time difference since the last car was inserted is greater than the so called "new car time". The new car has a normal distributed desired velocity around 120 km/h with variance 8 and a safe time headway around 2 s with variance 0.4. With these normal distributed values we model the different driving behavior of the drivers. The cars are added with the same velocity as their desired velocity.
On the slower lane are also trucks adjoined with a probability of 20%. The conditions which have to be fulfilled to add a new vehicle are the same as on the fast lane. The trucks have a normal distributed desired velocity around 80 km/h with variance 4 and a safe time headway around 3 s with variance 0.4. The trucks are added with the same velocity as their desired velocity. The cars on this lane have the same desired velocity as those on the fast lane but they are added with 80 km/h. The outcome of this is a better drive-in behavior.
We set the "new car time" of the fast lane to 2 s and of the other lane to 5 s, so we accomplished the highest utilization of the highway.

5.2.5 Speed limit

The speed limit component of the program enables the option to limit the desired velocity of the vehicles on a certain section of the highway during a desired time period. In the case of the sun blinds the drivers the speed limit is reduced several hundred meters ahead of the blinding section. This happens earlier than the sun begins to blind because this problem is predictable with the know-how about the weather-dependent behavior of the drivers on this section of the highway. In the unpredictable event of an accident the velocity is limited shortly after the crash occurs and several hundred meters ahead of the chrash.

5.2.6 Disturbance

This part of the program is able to induce our two disturbance types. This two cases are implemented as follows:

- If the sun blinds the drivers on a certain section of the highway they will slow down. We modeled this with a reduction of the desired velocity of the drivers in this section during the time the sun blinds.
- If the disturbance is caused by an accident the velocity of the involved vehicles is reduced very fast till they stand still. Not only the vehicles on the lane with the accident will slow down but also the vehicles on the other lane because they have to pass the accident very carefully. We modeled this by setting the desired velocity of the vehicles near the position of the accident to 1 m/s. When the accident is cleared the desired velocity of the vehicles is again set to the normal values.

5.2.7 Saving

All data is saved every second to a predefined folder. These data can be processed in the following functions.

5.3 Data processing functions

5.3.1 Plot graphics

Graphics of the simulation can be generated with the function "`plot_graphics.m`" (cf. appendix 9.2). This function calculates the traffic density (vehicles/km), the average velocity of the cars and trucks and counts the incoming and outgoing vehicles. Then it creates a traffic density plot and a velocity plot for each lane.

5.3.2 Plot simulation

The function "`plot_simulation.m`" (cf. appendix 9.3) visualizes the computed traffic flow on the highway. Therefore we extended the function "`draw_car.m`"[2] from the MSSSM course to the function "`draw_car_diff_colors.m`" (cf. appendix 9.4). The cars which were originally inserted in the fast lane (upper lane) are displayed red and the cars from the slow lane (lower lane) are green. Figure 2 shows a snapshot of the visualization while simulating a nondisturbed section of the highway without any speed limit.

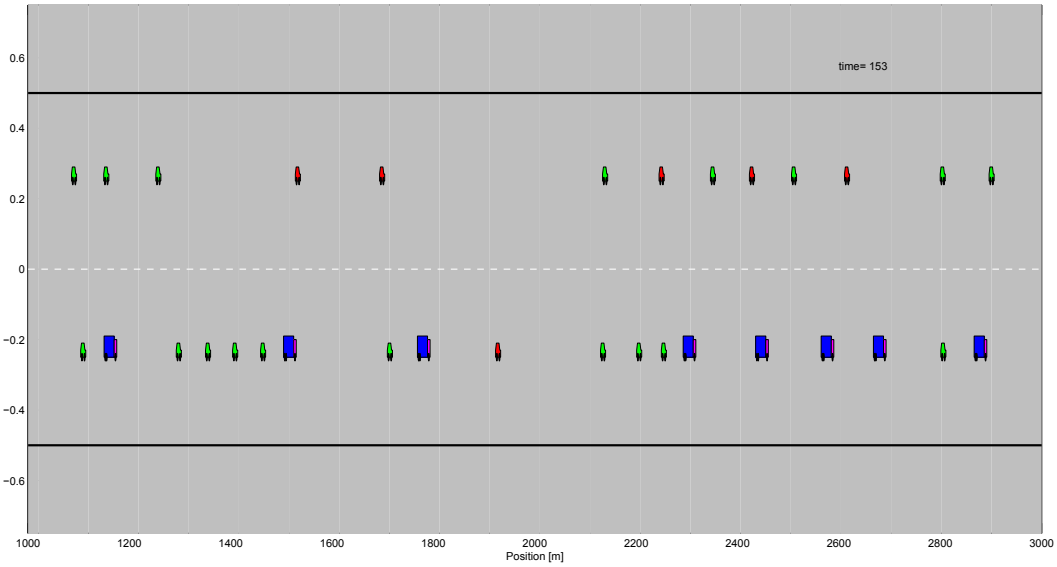

Figure 2: Example snapshot of the visualization of the highway

5.3.3 Computation and evaluation of multiple simulations

The execution of a whole bunch of simulations becomes possible by the function "`run_simulation.m`" (cf. appendix 9.5). It computes several simulations one after another and evaluates the data with the aid of the function "`evaluate_simulation.m`" (cf. appendix 9.6).

6 Simulation Results and Discussion

6.1 General Parameters and Settings

We analyzed two different cases. In the first case we investigated if our traffic model can induce a traffic jam just because of the implemented driver behavior and high traffic density. It means there is no specific event that should lead to a delaying of the traffic performance like an accident. But the question is if the implemented driver behavior with reaction time and lane changing could generate slackening of speed and finally stop-and-go traffic or even a jam? In the second case we inserted particular disturbance which should excite a slowdown and finally cause a traffic jam. We modeled two different disturbance types. The first type is a disturbance over a longer period of space and time, like the sun which is blinding at the evening. This disturbance forces the drivers to decrease their desired velocity by a certain factor (0.6) on a highway section of 1000 meters and for duration of 10 minutes. Therefore many drivers over a longer period are disturbed and forced to slow down which could generate a traffic jam. The second type of disturbance simulates an accident. The cars near to the accident on both lanes suddenly break very hard to simulate this accident. So the following cars have also to break hard what induces a traffic jam.

We needed a large scale of highway to survey the influence of disturbance and speed limitation over the traffic. However, to reduce the complexity and the calculating effort we wanted as small scales as possible. After some trials we decided to simulate the highway over a length of 10 kilometers and duration of 2000 seconds, because with this scale the boundary effects were almost negligible and the calculating effort is still feasible.

In the following we call the fast lane "lane 1" and the slower lane "lane 2". To evaluate the different simulations we needed some tools and characteristic factors. To monitor the traffic flow we decided to plot the density of vehicles per kilometer and the average velocity in a certain scope for each line separately in a graphic. As characteristic factors we chose the average time to transit the simulated highway, for both cars and trucks separately. And we determined also the incoming and outgoing traffic flow (vehicles/h). The incoming traffic flow stays constantly high. Every five seconds a new vehicle starts at the beginning of lane two if there is enough distance to the car ahead. On lane 1 a new car (there are no trucks on lane 1) starts if it has enough space to enter the simulated highway, but at most every two seconds. Therefore the incoming traffic is constant, unless a traffic jam blocks the inlet of the highway.

Because we implemented for each vehicle a random variance of the desired velocity v_0 and the desired safe time headway T, every simulation could run different and produce variant results. For this reason we simulated every case 20 times to make statistical statements possible.

The aim of this project is to observe the influence of a speed limit reduction on the traffic behavior. To reduce the calculating effort we decided just to simulate the common speed limit reductions from 120 km/h to 100 km/h or 80 km/h. These are the speed limit reductions that are already in use on swiss highways.

6.2 Without disturbance

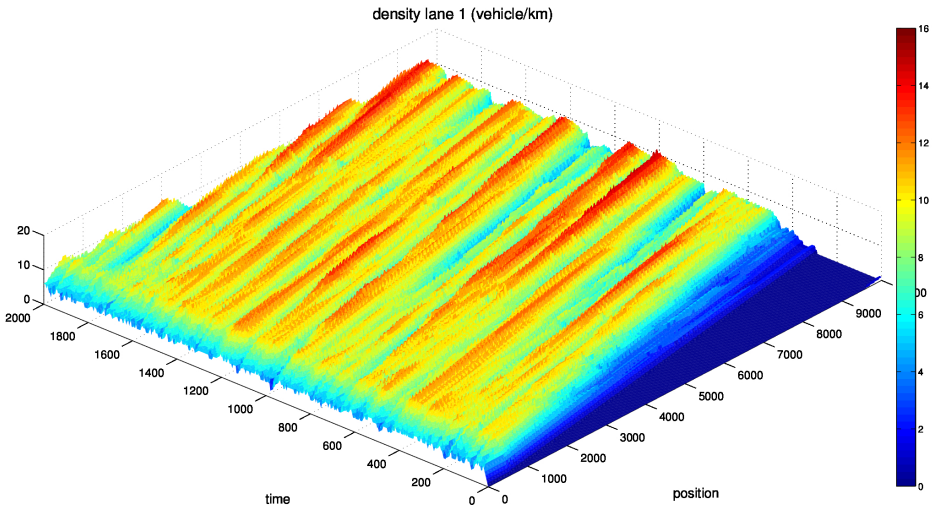

Figure 3: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 1, without any disturbance, without speed limit reduction

We expected with a reaction time of one second that self-induced stop-and-go traffic should occur. But with our model we couldn't generate such a traffic jam. The spatiotemporal density plot shows a waved contribution (see Fig. 3) because faster vehicles accumulate behind slower vehicles and moving groups develop. Between these groups there are gaps with a low density or even without cars. The blue corner at the beginning of the simulation is explained by the initial condition (empty highway). Therefore the first car reaches kilometer 10 around 300 seconds after the

beginning of the simulation. The spatiotemporal velocity plot of lane 2 shows some regions with a local slowdown up to 60 km/h but no real stop-and-go traffic (see Fig. 4). One can also notice the same blue corner at the beginning of the simulation caused by the same reasons as in the density plot. An average car needs 392 seconds to pass the 10 kilometers of the highway. Although the desired velocity is approximately 120 km/h the average speed of a car (92 km/h) is rather slow because of the high traffic density. The trucks need on average 472 seconds to cross the highway (average speed of 76 km/h). So they are able to drive almost their desired velocity of 80 km/h. Further simulations with a speed limit reduction are senseless because there is no breakdown of the traffic flow that could be improved with such a sanction.

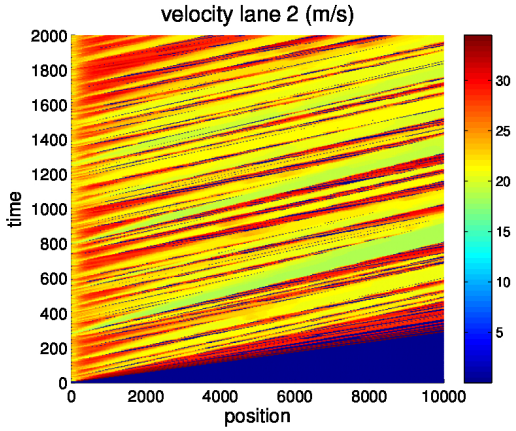

Figure 4: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, without any disturbance, without speed limit reduction

6.3 Disturbance type: sun blinding

The disturbance begins in the middle of the simulated highway at 5 kilometers and spans one kilometer. All drivers in this section are disturbed while 10 minutes. An example for such a disturbance could be sunset which affects all drivers who are driving towards the sun. In answer to the disturbance the drivers break and reduce their desired velocity with the disturbance factor (0.6).

We expected that the sudden appear of the disturbance could cause to a hard breaking behavior which leads to a traffic jam. To prevent such an immediate speed reduction we reduce the speed limit. The reduction begins 100 seconds before the disturbance because you can predict the date of the disturbance (like a sunset). The speed limit reduction begins two kilometers ahead of the disturbance and is

annihilated one kilometer behind, so we reduce the limit from kilometer three to kilometer seven.

6.3.1 Without speed limit reduction

Without speed limit reduction the vehicle density rises in the region of the disturbance and drops just after the disturbance at kilometer six (see Fig. 5). The wave structure of the density contribution is preserved. That means the vehicles group behind a slower vehicle which leads to a gap between two groups with a very low vehicle density. The spatiotemporal plot is very similar for both lanes. Because the

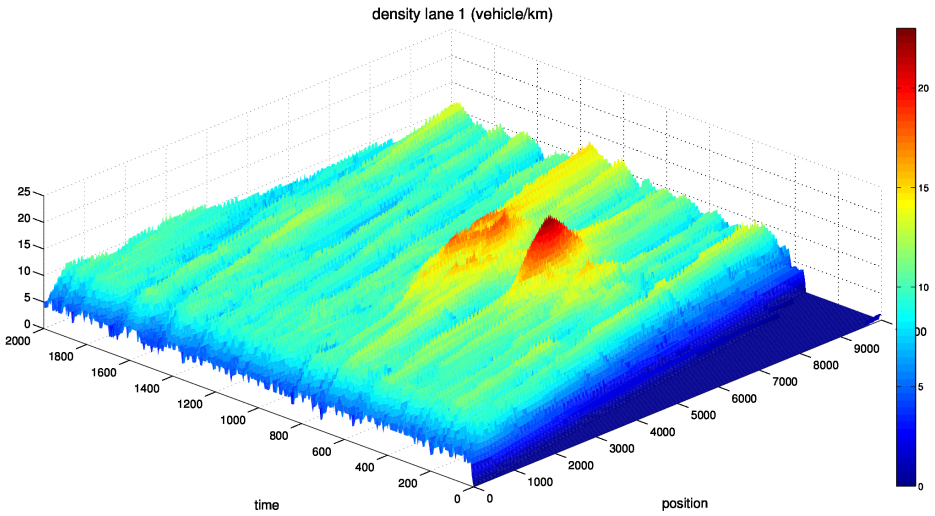

Figure 5: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 1, with disturbance type sun blinding, without speed limit reduction

disturbance appears suddenly the drivers have to break hard which can induce a traffic jam at the start of the disturbed zone particularly on lane 1 where the cars arrive with a higher velocity (see Fig. 6). The jam propagates in the direction of the highway origin with the time. This is the same effect as in reality, the jam moves against the driving direction. The traffic jam dissolves if not enough vehicles arrive at the beginning of the jam. The jam on lane 1 can also influence the vehicle movement on lane 2. So a jam occurs one kilometer ahead of the disturbance at kilometer four(see Fig. 7). The jam switches to the other lane.

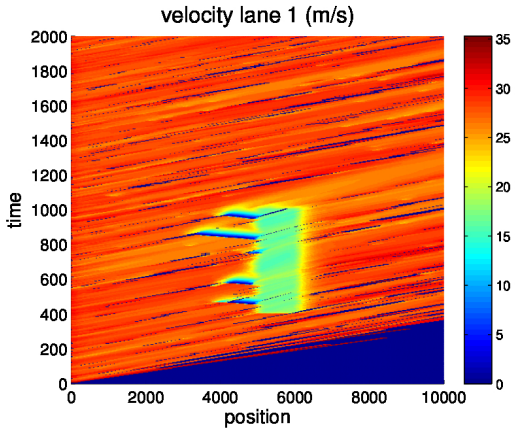

Figure 6: Spatiotemporal plot of the vehicle velocity [m/s] on lane 1, with disturbance type sun blinding, without speed limit reduction

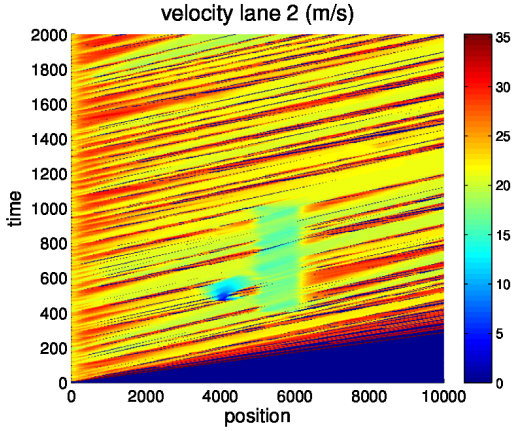

Figure 7: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type sun blinding, without speed limit reduction

6.3.2 Speed limit reduction: 100 km/h

At kilometer 3 and 100 seconds before the disturbance start the speed limit reduction begins. The desired velocity of the drivers is reduced to 100 km/h (plus a certain variance). During the disturbance the desired velocity is still the same as in the case without speed limit reduction. This sanction helps to prevent hard breaking at the beginning of the disturbance region. The most of the traffic jams can be avoided but it's still possible that a jam arises at the transition to the disturbance zone (see Fig. 10). Therefore the speed limit reduction is useful to prevent stop-and-go traffic because less traffic jams occur.

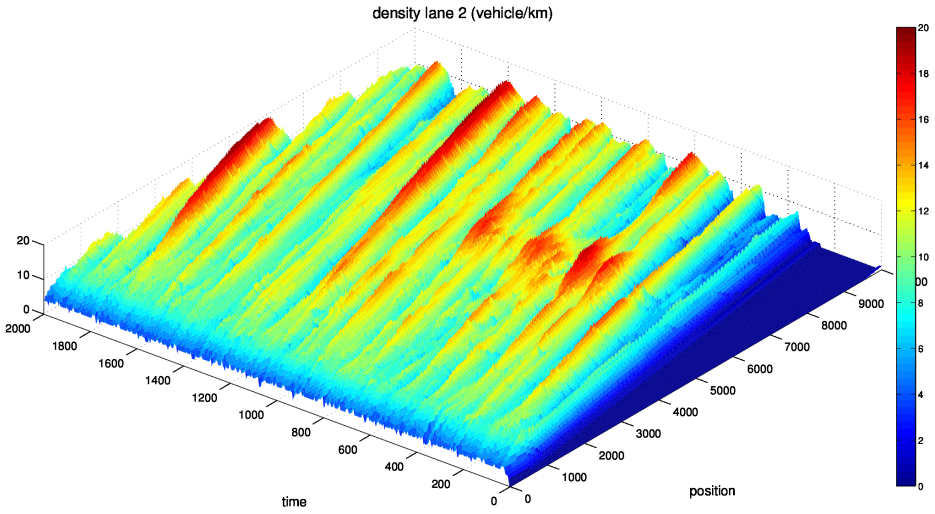

Figure 8: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 2, with disturbance type sun blinding, with speed limit reduction: 100 km/h

The density contribution stays still similar to the case without speed limit reduction (see Fig. 8).The high density hill in the disturbance region flattens somewhat. This means the peak is lowered from about 25 vehicles/km to about 20 vehicles/km. Therefore the shoulders of the density hill are less steep but more extended. Another interesting point is that one can discover after the disturbance time has elapsed still some zones with high density and low velocity develop but no more traffic jam.

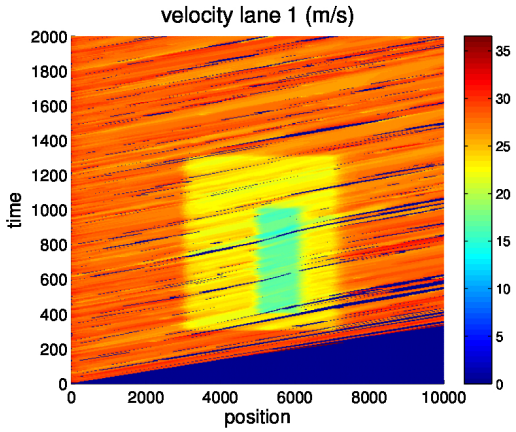

Figure 9: Spatiotemporal plot of the vehicle velocity [m/s] on lane 1, with disturbance type sun blinding, with speed limit reduction: 100 km/h

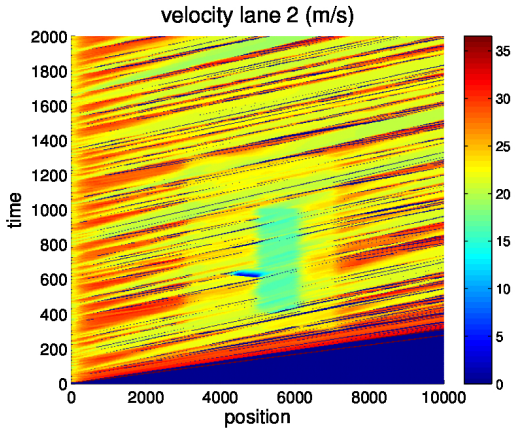

Figure 10: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type sun blinding, with speed limit reduction: 100 km/h

6.3.3 Speed limit reduction: 80 km/h

A speed limit reduction to 80 km/h doesn't improve the traffic flow. Basically the region where the traffic jam starts is displaced. There are some jams which start at the beginning of the speed limitation zone instead at the beginning of the disturbance zone. Because the speed limit 80 km/h is merely a little bit higher than the disturbed desired velocity the breaking behavior at the speed limit reduction (see Fig. 12) is very similar to the breaking behavior at the beginning of the disturbance without speed limit reduction (see Fig. 6). With such a restrictive speed limit reduction the high vehicle density regions locate within the speed limitation zone in the spatiotemporal plot (see Fig. 11) not only within the disturbance zone.

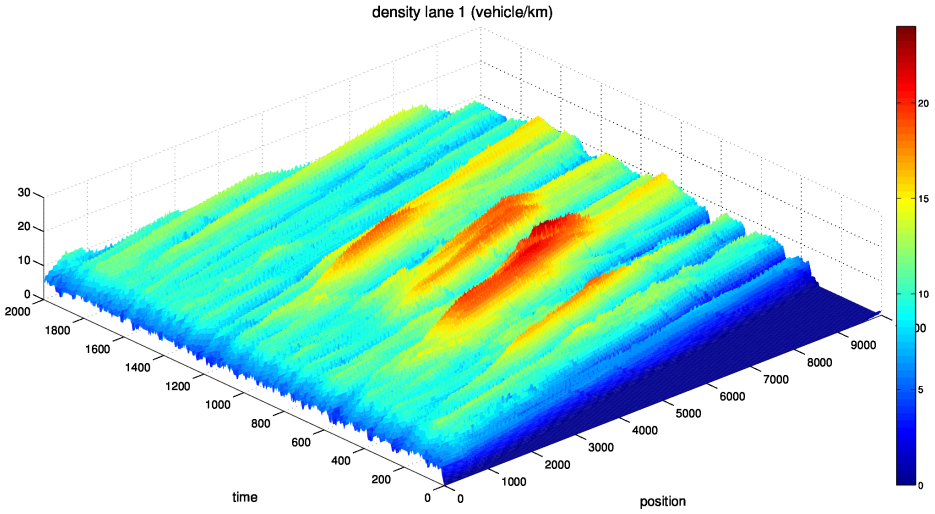

Figure 11: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 1, with disturbance type sun blinding, with speed limit reduction: 80 km/h

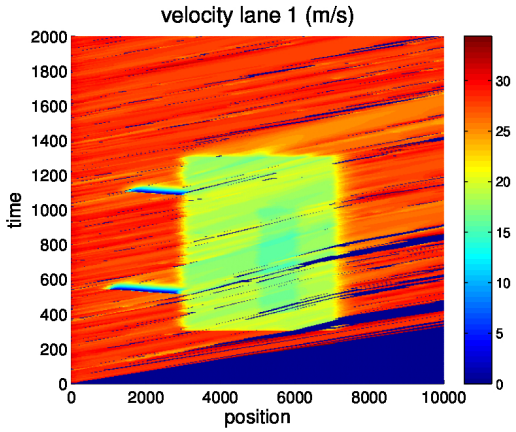

Figure 12: Spatiotemporal plot of the vehicle velocity [m/s] on lane 1, with disturbance type sun blinding, with speed limit reduction: 80 km/h

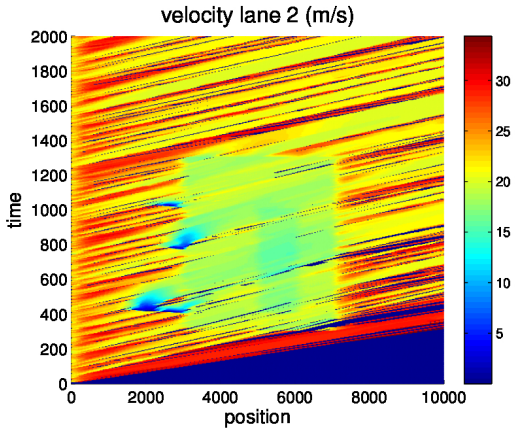

Figure 13: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type sun blinding, with speed limit reduction: 80 km/h

6.3.4 Characteristic factors

To compare how effective a speed limit reduction is, we calculated the average time a vehicle needs to pass the modeled highway. The average result and the deviation for a car is illustrated in Fig. 14. Obviously the average transit time increases from 405 to 410 seconds when a speed limit reduction to 100 km/h is inserted. So even there is less traffic jam the cars won't reach their destination earlier, because they have to slow down a little over a too long distance. If the speed limitation drops to 80 km/h the average time increases even more as the traffic jam appearance rises again. The cars need now 432 seconds to pass 10 kilometers. Interesting is the different performance of the trucks (see Fig. 15). With a speed limitation to 100 km/h they can decrease their average transit time from 487 to 479 seconds, because the limitation doesn't concern the trucks (they are just allowed to drive 80 km/h) and there are less jams to thwart them. Although the trucks don't need to slow down their desired velocity if the speed limitation drops to 80 km/h the average time raises as the development of jams increases again to 496 seconds.

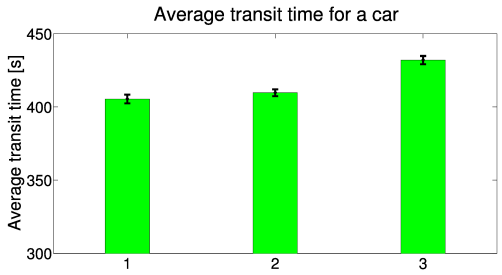

Figure 14: Average time to transit the highway for a car (with deviation), disturbance type sun blinding. Bar 1: no speed limit reduction; Bar 2: speed limit reduction to 100 km/h; Bar 3: speed limit reduction to 80 km/h

The incoming traffic flow stays constant for every case: approximately 1070 vehicles/h on lane 1 and 703 vehicles/h on lane 2. This means the highway is capable to absorb this amount of vehicles. Surprisingly the outgoing traffic flow doesn't drop. It also remains almost stable: approximately 745 vehicles/h on lane 1 and 675 vehicles/h on lane 2. The difference between the incoming and outgoing traffic flow is explainable by the cars which came in the highway but they haven't it crossed yet.

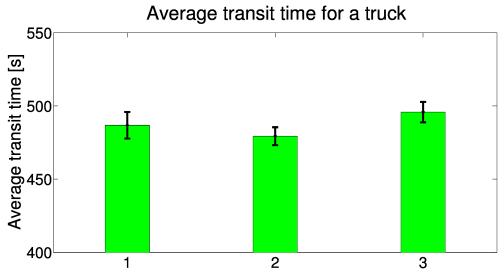

Figure 15: Average time to transit the highway for a truck (with deviation), disturbance type sun blinding. Bar 1: no speed limit reduction; Bar 2: speed limit reduction to 100 km/h; Bar 3: speed limit reduction to 80 km/h

6.4 Disturbance type: accident

Like a real accident this disturbance type appears suddenly. Therefore we implemented a fast slowdown. At a specific date (400 seconds) the drivers between kilometer 5 and kilometer 5.1 set their desired velocity very low (1 m/s) but not zero because this would excite a division by zero in our simulation. The accident influences both lanes, for this reason all vehicles have to slow down and the jam is growing quickly. To reduce the time necessary for computing the accident lasts just five minutes otherwise we would have to expand the time and space scales. In reality one can't predict the point of time when an accident happens. So the speed limit reduction begins after the start of the accident. 60 seconds after the accident the reduction locates in the same region as in the case sun blinding, from kilometer three to kilometer seven. The speed limit reduction is annihilated 10 minutes after the disturbance end.

6.4.1 Without speed limit reduction

The immediate slowdown leads to a massive development of traffic jam in the region of the disturbance. As only a few cars per minute can pass the disturbance there is no chance to solve the jam, the incoming traffic flow exceeds the outgoing. The density within the disturbance is three to four times higher than in the case sun blending (see Fig. 16). The length of the jam is expanding towards the beginning of the highway since more cars are arriving than leaving. After the cancellation of the accident the jam is beginning to solve at the front end but there are still vehicles arriving at the rear end. The congestion is moving against the direction of traffic such as it can be observed in reality (see Fig. 18). Finally around 700 seconds after the accident the traffic is dispersed into forward direction.

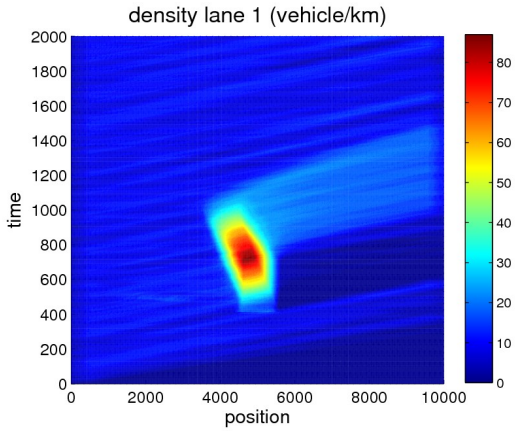

Figure 16: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 1, with disturbance type accident, without speed limit reduction

Another interesting point is the emergence of a second leg of the jam (see Fig. 17). This leg moves much faster towards the beginning of the highway and his behavior is very similar to the jam behavior in the case sun blinding without speed limitation (see Fig. 6). We suppose this is a characteristic of immediate slowdown. The drivers break harder that would be necessary due to the reaction time and the traffic comes to a standstill.

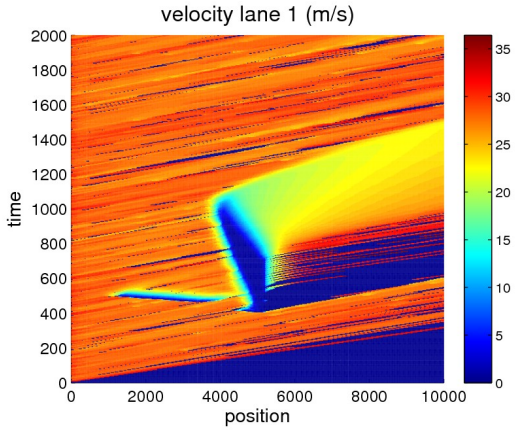

Figure 17: Spatiotemporal plot of the vehicle velocity [m/s] on lane 1, with disturbance type accident, without speed limit reduction

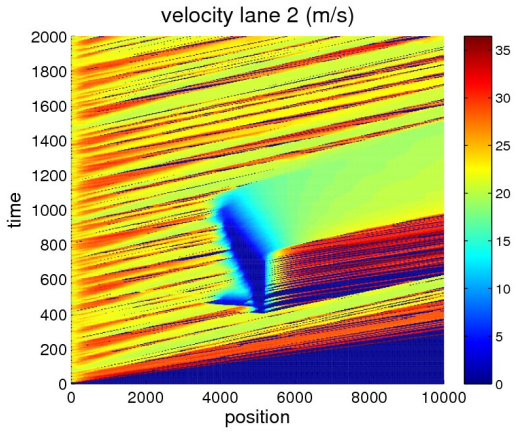

Figure 18: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type accident, without speed limit reduction

6.4.2 Speed limit reduction: 100 km/h

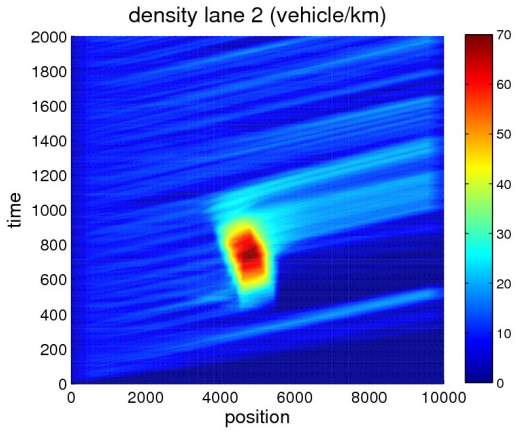

Figure 19: Spatiotemporal plot of the vehicle density [vehicle/km] on lane 2, with disturbance type accident, with speed limit reduction: 100 km/h

Though a speed limit reduction is set 2 kilometers ahead the disturbed zone the spatiotemporal plot of the velocity and density doesn't differ much to the case without speed limitation (see Fig. 19). The region with a slow velocity remains mainly the same. But the jam movement against the traffic direction is slightly reduced. Since the reduction starts 60 seconds after the accident it's not possible to

avoid the formation of a second leg of the jam because it builds up before. Otherwise the speed limit reduction helps to solve the established congestion (see Fig. 20).

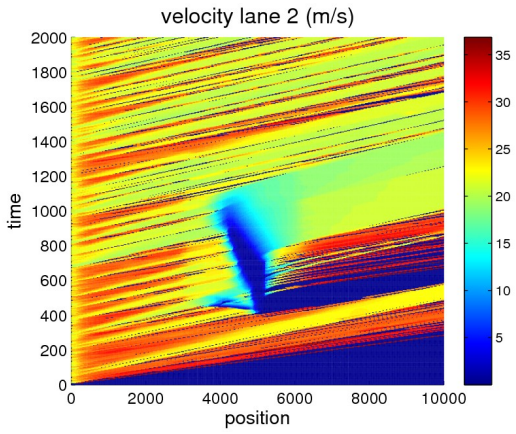

Figure 20: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type accident, with speed limit reduction: 100 km/h

6.4.3 Speed limit reduction: 80 km/h

Again a speed limit reduction to 80 km/h doesn't improve the traffic flow. The region with a very high density has nearly the same extent as in the case with a speed limit reduction to 100 km/h (see Fig. 19). Also similar to this case is the fact the speed limitation can't prevent the second leg of the jam (see Fig. 22), because it comes into action to late. Even worse there appears some additional jam caused by the speed limitation. This jam begins where the speed limitation intervenes, at kilometer 3 and develops the same as the second leg of the jam (see Fig. 21). The reasons for this emergence are the cars which enter the speed limitation zone and break too hard due to the reaction time. The main jam disperses into forward direction 600 seconds after the accident, 100 seconds earlier than in the case without speed limitation. This means the congestion can be relieved earlier, but the speed limitation is another 300 seconds in action.

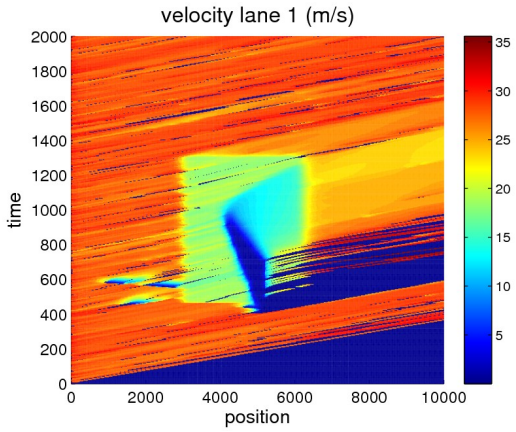

Figure 21: Spatiotemporal plot of the vehicle velocity [m/s] on lane 1, with disturbance type accident, with speed limit reduction: 80 km/h

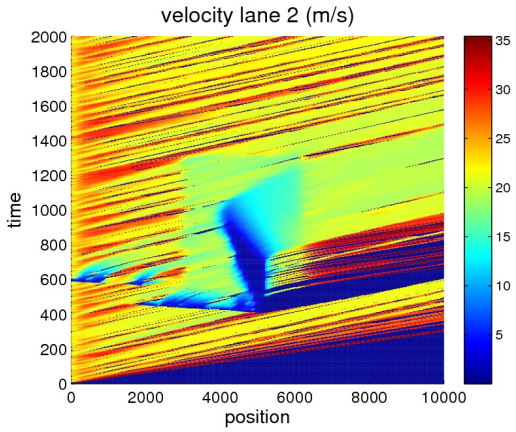

Figure 22: Spatiotemporal plot of the vehicle velocity [m/s] on lane 2, with disturbance type accident, with speed limit reduction: 80 km/h

6.4.4 Characteristic factors

The average time to pass the simulated highway for a car without speed limit reduction is 472 seconds. Like in the case disturbance type sun blinding, the average speed is increasing with a lower speed restriction. A speed limit of 100 km/h leads to an average transit time of 478 seconds, a speed limit of 80 km/h leads to 486 seconds (see Fig. 23). On a free highway with a length of four kilometers and without disturbance a car driver is losing 60 seconds if the speed limit drops from 120 km/h to 80 km/h. With disturbance type accident the speed limit reduction doesn't affect the average transit time in the same extent.

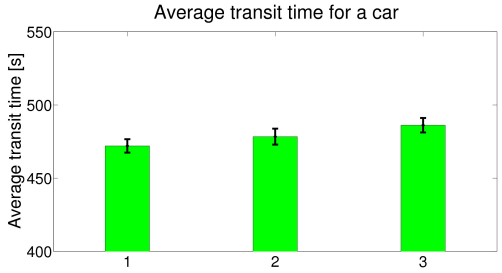

Figure 23: Average time to transit the highway for a car (with deviation), disturbance type accident. Bar 1: no speed limit reduction; Bar 2: speed limit reduction to 100 km/h; Bar 3: speed limit reduction to 80 km/h

Pretty much the same conclusion can be made for the trucks. The speed limit reduction from 120 km/h to 100 km/h increases the average transit time from 556 to 562 seconds, a reduction to 80 km/h raises the transit time furthermore to 568 seconds. So the trucks are even less influenced by speed limitation (see Fig. 24).

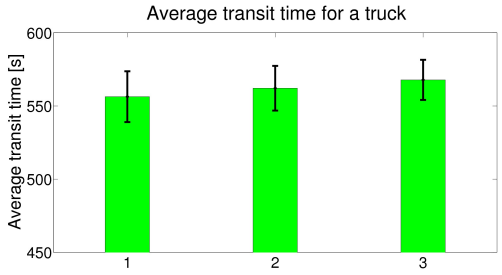

Figure 24: Average time to transit the highway for a truck (with deviation), disturbance type accident. Bar 1: no speed limit reduction; Bar 2: speed limit reduction to 100 km/h; Bar 3: speed limit reduction to 80 km/h

7 Summary and outlook

7.1 Without disturbance

Our model isn't able to produce self-induced traffic jam. Even with the implemented reaction time there was no chance to create a jam. We suppose the applied safety distances are still too large, the vehicles keep enough margins to react in time. The problem is we can't reduce the safety margins in our model, otherwise too much crashes appear and the simulation can't run until the end. We think in reality on highways self-induced traffic jam occurs because human drivers don't keep enough safety margins. Nevertheless they manage somehow to prevent almost all dangerous situations and crashes.

7.2 Disturbance type: sun blinding

If you just consider the average transit time as an optimization factor there is no big impact if one reduce the speed limit from 120 km/h to 100 km/h. The cars need a little bit more time, the trucks are somewhat faster. But other criteria should also be respected. Stop-and-go traffic is very annoying. Therefore a lot of drivers get stressed and furious and become a safety risk for themselves or other drivers. For this reason a uniform traffic flow is more desirable and helps making the road safer. A speed limit reduction to 100 km/h leads to less traffic jam and standstill. Hence a tempo reduction to 100 km/h should be taken into account. A reduction to 80 km/h doesn't show any clear advantage. The region where the traffic jam arises just relocates from the beginning of the disturbance zone to the beginning of the speed limit zone. Since there is traffic jam and speed limitation the average transit time increases, which is very unsatisfying. Generally one can say the jam is induced if the drivers have to reduce the desired velocity massively, therefore no large speed reduction step is requested.

7.3 Disturbance type: accident

The high incoming traffic flow and the immediate and heavy slowdown provoke inevitably a standstill and a rapid development of traffic jam. A speed limit reduction decreases the propagation of the separated second leg of the jam, but the main jam is hardly affected. The development of the average transit time shows the same effect as well. The vehicles need more time to pass the highway if the speed limit is reduced, but there is just a slight rise because the main jam length and duration stay similar in every case. Again a reduction to 100 km/h seems to be a good tradeoff. The vehicles already decrease their velocity ahead of the jam and it's safer to reach

the jam with a lower velocity. In addition the second leg of the jam which is moving much faster towards the beginning of the highway. However the vehicles need a few seconds more to cross the highway compared to the case without speed limit reduction.

7.4 Outlook

There are still a lot of outstanding issues we couldn't cover in our work. We set the parameters for the beginning and finish time and place of the speed limitation to some certain values which we generated empirical and seemed reasonable to us. So these parameters could probably be optimized. Where and when should the limitation start and when should it finish?

For the disturbance case accident an interesting approach would be to initialize the speed limitation dependent on the actual traffic flow, e.g. insert automatically a speed limit on a part of the highway if the traffic density exceeds a certain value. The same considerations are also valid for the annihilation of the speed limit reduction. If the density falls below a certain value, the reduction is annihilated.

The massive slowdown from 120 km/h to 80 km/h brought the problem of causing traffic jam. If the speed limit reduction is inserted step by step like on real highways this circumstances could be avoided. So could the traffic flow be improved if the speed limit reduction is distributed over a longer section of highway?

8 References

[1] M. Treiber and D. Helbing, "Explanation of observed features of self-organization in traffic flow," *arXiv preprint cond-mat/9901239*, 1999.

[2] S. Balietti and K. Donnay, "`draw_car.m`," *Lecture with Computer Exercises: Modelling and Simulating Social Systems with MATLAB*, 2012.

9 Appendix

9.1 simulate main

```
function time = simulate_main(simtime, l_highway, disturbance, speed_limit, dir)
%simulate_main(simtime, l_highway, disturbance, speed_limit, dir)
%Simulates the traffic flow of a highway with lenght l_highway for the duration of simtime
%disturbance(0:without; 1:lane2; 2:both; 3:accident)
%dir: data folder

%clean up
close all;
clc;
format long

%Indexes of the state matrix
ix = 1;             %Position
iv = 2;             %Velocity
iacc = 3;           %Acceleration
iv0 = 4;            %Desired velocity
iT = 5;             %Safe time headway
icp = 6;            %Changed position
itype = 7;          %Type 1:car; 2:truck
iol = 8;            %Original lane
idili = 9;          %Disturbance and speed limit (0:nothing, 1: only speed limit, 2:only disturbed, 3:both)
itime = 10;         %Car time elapsed on highway

%Parameters
dt = 0.1;           %Time Step [s] (needs to be 0.1!!)
simend = simtime;   %Simulation time [s]
nct_1 = 10;         %New Car Time lane 1 [s]
nct_2 = 3;          %New Car Time lane 2 [s]
T_d = 1;            %Dead time (has to be a multiple of 0.1)
dist_factor = 0.6; %Disturbance factor
pitnc_1 = 0;        %Point in time new car [s] lane1
pitnc_2 = 0;        %Point in time new car [s] lane2
l_highway;          %Length Highway [m]
n_1=2;              %First updated car lane 1
n_2=2;              %First updated car lane 2

%Initialize Variables
time = 0;           %Elapsed time [s]
ec1 = 0;            %Number of entering cars lane 1
lc1 = 0;            %Number of leaving cars lane 1
ec2 = 0;            %Number of entering cars lane 2
lc2 = 0;            %Number of leaving cars lane 2
type = 1;           %Initialize type
av_time_c = 0;      %Average transit time car
av_time_t = 0;      %Average transit time truck
lcc = 0;            %Leaving car counter
ltc = 0;            %Leaving truck counter
crash = 0;          %Crash Detection

%States of the Cars [x, v, acc, v0, T, changedposition, type(1:car; 2:truck), original lane, disturbance]
v0_car0 = 120/3.6;
T_car0 = 2;
state_1 = [l_highway+100 v0_car0 0 v0_car0 T_car0 0 1 1 0 0];  %State of lane 1
state_2 = [l_highway+100 v0_car0 0 v0_car0 T_car0 0 1 2 0 0];  %State of lane 2

%Intelligent Driver Model parameters
%Car
```

```
delta = 5;        %describes acceleration behavior (velocity difference)
epsilon = 3;      %describes acceleration behavior (gap)
a_c = 0.6;        %maximal acceleration [m/s^2]
b_c = 0.9;        %maximal decceleration [m/s^2]
l_c = 5;          %vehicle length [m]
s0 = 2;           %jam distance [m]

%Truck
a_t = 0.2;        %maximal acceleration [m/s^2]
b_t = 0.4;        %maximal decceleration [m/s^2]
l_t = 16;         %vehicle length [m]

%Save Parameters
save([dir '/Parameters'])

%Time loop
for time = 0:dt:simend

    %Increase elapsed time
    time_1 = ones(size(state_1,1),1)*dt;
    time_2 = ones(size(state_2,1),1)*dt;
    state_1(:,itime) = state_1(:,itime) + time_1;
    state_2(:,itime) = state_2(:,itime) + time_2;

    %Change vehicles from lane 1 to lane 2
    if(mod(time, 0.2) == 0)
        %Tag changing cars from lane 1 to lane 2
        state_1(:,icp) = zeros(size(state_1,1),1);
        if(size(state_1,1) > 2)
            %Delete car which has reached l_highway
            if(state_1(n_1,ix)>l_highway)
                av_time_c = av_time_c + state_1(1,itime);
                lcc = lcc + 1;
                state_1(1,:) = [];
                lc1 = lc1 + 1;
            end
            for vehicle_1 = n_1:size(state_1,1)-1

                %Determine vehicle on lane 2 ahead/ behind this vehicle
                for vehicle_2 = n_2:size(state_2,1)

                    if(state_2(vehicle_2, ix) < state_1(vehicle_1, ix))
                        vehicle_a = vehicle_2-1;   %Vehicle ahead
                        vehicle_b = vehicle_2;     %Vehicle behind

                        %Check lane change situation
                        v_1 = state_2(vehicle_a, iv);
                        v = state_1(vehicle_1, iv);
                        dv = v - v_1;
                        T = state_1(vehicle_1, iT);
                        a = a_c;
                        b = b_c;
                        l = l_c;
                        s_a = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0) + l;↙
%Desired gap to vehicle ahead

                        v_1 = state_1(vehicle_1, iv);
                        v = state_2(vehicle_b, iv);
                        dv = v - v_1;
                        T = state_2(vehicle_b, iT);
```

```
117                         switch state_2(vehicle_b, itype)
118                             case 1          %Car
119                                 a = a_c;
120                                 b = b_c;
121                                 l = l_c;
122                             case 2          %Truck
123                                 a = a_t;
124                                 b = b_t;
125                                 l = l_t;
126                         end
127                         s_b = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0) + l;↙
%Desired gap to vehicle behind
128
129                         c1 = state_2(vehicle_a, ix)-state_1(vehicle_1, ix) > 300;↙
%Gap to vehicle ahead_2 > 300
130                         c2 = state_1(vehicle_1, ix)-state_2(vehicle_b, ix) > s_b;↙
%Gap to vehicle behind_2 big enough
131                         c3 = state_1(vehicle_1, iv0) > state_1(vehicle_1-1, iv);↙
%v0 > v of vehicle ahead_1
132                         c4 = state_2(vehicle_a, iv) > state_1(vehicle_1-1, iv);↙
%v of vehicle ahead_2 > v of vehicle ahead_1
133                         c5 = state_2(vehicle_a, ix)-state_1(vehicle_1, ix) > s_a;↙
%Gap to vehicle ahead_2 big enough
134                         c6 = state_2(vehicle_a, iv) < state_1(vehicle_1, iv0);↙
%v of vehicle ahead_2 < v0
135                         c7 = state_1(vehicle_1-1, ix)-state_1(vehicle_1, ix) > 200;↙
%Gap to vehicle ahead_1 > 200
136
137                         if((c1 && c2) || (c3 && c4 && c5 && c2 && not(c6 && c7)))
138                             state_1(vehicle_1, icp) = vehicle_b;    %Set changed↙
position
139                         end
140
141                     end
142
143                 end
144
145             end
146
147         end
148
149         %Change the vehicles lane 1 to 2
150         ccc1 = 0; % Set changing car counter lane 1
151         for vehicle_1 = n_1:size(state_1,1)-1
152
153             if (state_1(vehicle_1, icp)>0)
154                 pos = state_1(vehicle_1, icp) + ccc1;
155                 state_2_temp_1 = state_2(1:pos-1,:);
156                 state_2_temp_2 = state_2(pos:size(state_2,1),:);
157                 state_2 = [state_2_temp_1; state_1(vehicle_1,:); state_2_temp_2];
158                 ccc1 = ccc1 + 1;
159                 state_2(pos, icp) = -1;
160             end
161
162         end
163
164         %Delete changed cars lane 1
165         for i=1:ccc1
166
167             for vehicle_1=n_1:size(state_1,1)-1
```

```
168
169                 if (state_1(vehicle_1, icp)>0)
170                 state_1(vehicle_1,:)=[];
171                 break
172                 end
173             end
174
175         end
176     end
177
178     %Change vehicles from lane 2 to lane 1
179     if(mod(time, 0.2) == 0.1)
180         %Tag changing cars from lane 2 to lane 1
181         state_2(:,icp) = zeros(size(state_2,1),1);
182         if(size(state_2,1) > 2)
183             %Delete car which has reached l_highway
184             if(state_2(n_2,ix)>l_highway)
185                 if(state_2(1,itype) == 1)
186                     av_time_c = av_time_c + state_2(1,itime);
187                     lcc = lcc + 1;
188                 else
189                     av_time_t = av_time_t + state_2(1,itime);
190                     ltc = ltc + 1;
191                 end
192                 state_2(1,:) = [];
193                 lc2 = lc2 + 1;
194             end
195             for vehicle_2 = n_2:size(state_2,1)-1
196
197                 if(state_2(vehicle_2, itype)==1)
198                     %Determine vehicle on lane 2 ahead/ behind this vehicle
199                     for vehicle_1 = n_1:size(state_1,1)
200
201                         if(state_1(vehicle_1, ix) < state_2(vehicle_2, ix))
202                             vehicle_a = vehicle_1-1;   %Vehicle ahead
203                             vehicle_b = vehicle_1;     %Vehicle behind
204
205                             %Check lane change situation
206                             v_1 = state_1(vehicle_a, iv);
207                             v = state_2(vehicle_2, iv);
208                             dv = v - v_1;
209                             T = state_2(vehicle_2, iT);
210                             switch state_2(vehicle_2, itype)
211                                 case 1          %Car
212                                     a = a_c;
213                                     b = b_c;
214                                     l = l_c;
215                                 case 2          %Truck
216                                     a = a_t;
217                                     b = b_t;
218                                     l = l_t;
219                             end
220                             s_a = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0) + l;↙
%Desired gap to vehicle ahead
221
222                             v_1 = state_2(vehicle_2, iv);
223                             v = state_1(vehicle_b, iv);
224                             dv = v - v_1;
225                             T = state_1(vehicle_b, iT);
226                             a = a_c;
```

```
227                             b = b_c;
228                             l = l_c;
229                             s_b = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0) + l;↙
%Desired gap to vehicle behind
230
231                             c1 = state_1(vehicle_a, ix)-state_2(vehicle_2, ix) >↙
300;    %Gap to vehicle ahead_1 > 300
232                             c2 = state_2(vehicle_2-1, ix)-state_2(vehicle_2, ix) <↙
200;  %Gap to vehicle ahead_2 < 200
233                             c3 = state_2(vehicle_2-1, iv) < state_2(vehicle_2,↙
iv0);     %v of vehicle ahead_2 < v0
234                             c4 = state_2(vehicle_2, ix)-state_1(vehicle_b, ix) >↙
s_b;    %Gap to to vehicle behind_1 big enough
235                             c5 = state_1(vehicle_a, iv) > state_2(vehicle_2-1, iv);↙
%v of vehicle ahead_1 > v of vehicle ahead_2
236                             c6 = state_1(vehicle_a, ix)-state_2(vehicle_2, ix) >↙
s_a;    %Gap to vehicle ahead_1 big enough
237
238
239                             if((c1 && c2 && c3 && c4) || (c2 && c3 && c5 && c6 &&↙
c4))
240                                 state_2(vehicle_2, icp) = vehicle_b;    %Set↙
changed position
241                             end
242
243                         end
244
245                     end
246                 end
247             end
248
249         end
250
251         %Change the vehicles lane 2 to 1
252         ccc2 = 0;   % Set changing car counter lane 2
253         for vehicle_2 = n_2:size(state_2,1)-1
254
255             if (state_2(vehicle_2, icp)>0)
256                 pos = state_2(vehicle_2, icp) + ccc2;
257                 state_1_temp_1 = state_1(1:pos-1,:);
258                 state_1_temp_2 = state_1(pos:size(state_1,1),:);
259                 state_1 = [state_1_temp_1; state_2(vehicle_2,:); state_1_temp_2];
260                 ccc2 = ccc2 + 1;
261                 state_1(pos, icp) = -1;
262             end
263
264         end
265
266         % Delete changed cars lane 2
267         for i=1:ccc2
268
269             for vehicle_2=n_2:size(state_2,1)-1
270                 if (state_2(vehicle_2, icp)>0)
271                 state_2(vehicle_2,:)=[];
272                 break
273                 end
274             end
275
276         end
277     end
```

```
278
279     %Update accelerations of the vehicles from lane 1
280     if(mod(time,T_d) == 0)
281         for vehicle = n_1:size(state_1,1)
282
283             switch state_1(vehicle, itype)  %Vehicle type?
284                 case 1          %Car
285                     a = a_c;
286                     b = b_c;
287                     l = l_c;
288                 case 2          %Truck
289                     a = a_t;
290                     b = b_t;
291                     l = l_t;
292             end
293
294             x = state_1(vehicle,ix);       %Position of this vehicle
295             xm1 = state_1(vehicle-1,ix);   %Position of vehicle in front
296             v = state_1(vehicle,iv);       %Velocity of this vehicle
297             vm1 = state_1(vehicle-1,iv);   %Velocity of vehicle in front
298             v0 = state_1(vehicle,iv0);     %Desired velocity of this vehicle
299             T = state_1(vehicle,iT);       %Safe time headway of this vehicle
300
301             ds = xm1 - x - l;       %gap to the vehicle in front
302             dv = v - vm1;           %velocity difference to the vehicle in front
303             ss = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0);       %effective desired↙
distance
304
305             state_1(vehicle,iacc) = a*(1 - (v/v0)^delta - (ss/ds)^epsilon);    %new↙
acceleration
306
307         end
308
309     end
310
311     %Update accelerations of the vehicles from lane 2
312     if(mod(time,T_d) == 0)
313         for vehicle = n_2:size(state_2,1)
314
315             switch state_2(vehicle, itype)  %Vehicle type?
316                 case 1          %Car
317                     a = a_c;
318                     b = b_c;
319                     l = l_c;
320                 case 2          %Truck
321                     a = a_t;
322                     b = b_t;
323                     l = l_t;
324             end
325
326             x = state_2(vehicle,ix);       %Position of this vehicle
327             xm1 = state_2(vehicle-1,ix);   %Position of vehicle in front
328             v = state_2(vehicle,iv);       %Velocity of this vehicle
329             vm1 = state_2(vehicle-1,iv);   %Velocity of vehicle in front
330             v0 = state_2(vehicle,iv0);     %Desired velocity of this vehicle
331             T = state_2(vehicle,iT);       %Safe time headway of this vehicle
332
333             ds = xm1 - x - l;       %gap to the vehicle in front
334             dv = v - vm1;           %velocity difference to the vehicle in front
335             ss = s0 + max(v*T + (v*dv)/(2*sqrt(a*b)), 0);       %effective desired↙
```

```
distance
336
337             state_2(vehicle,iacc) = a*(1 - (v/v0)^delta - (ss/ds)^epsilon);    %new↙
acceleration
338
339         end
340
341     end
342
343     %Update velocities and positions of the cars from lane 1
344     state_1(n_1-1,ix) = state_1(n_1-1,ix) + state_1(n_1-1,iv)*dt;
345     for vehicle=n_1:size(state_1,1)
346
347         x = state_1(vehicle,ix);       %Position of this vehicle
348         v = state_1(vehicle,iv);       %Velocity of this vehicle
349         acc = state_1(vehicle,iacc);   %Acceleration of this vehicle
350
351         state_1(vehicle,iv) = v + acc*dt;    %Update velocity
352         if(state_1(vehicle,iv)<0)
353             state_1(vehicle,iv) = 0;
354         end
355         state_1(vehicle,ix) = x + v*dt;      %Update position
356
357     end
358
359     %Update velocities and positions of the cars from lane 2
360     state_2(n_2-1,ix) = state_2(n_2-1,ix) + state_2(n_2-1,iv)*dt;
361     for vehicle=n_2:size(state_2,1)
362
363         x = state_2(vehicle,ix);       %Position of this vehicle
364         v = state_2(vehicle,iv);       %Velocity of this vehicle
365         acc = state_2(vehicle,iacc);   %Acceleration of this vehicle
366
367         state_2(vehicle,iv) = v + acc*dt;    %Update velocity
368         if(state_2(vehicle,iv)<0)
369             state_2(vehicle,iv) = 0;
370         end
371         state_2(vehicle,ix) = x + v*dt;      %Update position
372
373     end
374
375     %Add new vehicle Lane 1
376     v0_new = (120+(randn*8))/3.6;       %Desired velocity: 120 km/h with normal↙
distribution with variance 8
377     T_new = 2+(randn*0.4);              %Safe time headway: 2 s with normal↙
distribution with variance 0.4
378     dv = v0_new - state_1(size(state_1,1), iv);
379     ss = s0 + max(v0_new*T_new + (v0_new*dv)/(2*sqrt(a_c*b_c)), 0);
380
381     if(state_1(size(state_1,1),ix)>ss && (time-pitnc_1)>nct_1)
382             new_vehicle = [0 v0_new 0 v0_new T_new 0 1 1 0 0];       %Create new↙
vehicle [x, v, acc, v0, T, changedposition, type(1:car; 2:truck), original lane,↙
disturbance]
383
384             state_1 = [state_1; new_vehicle];    %Add new vehicle
385
386             pitnc_1 = time;
387             ec1 = ec1 + 1;
388     end
389
```

```
390     %Add new vehicle Lane 2
391     switch type
392         case 1
393             v0_new = (120+(randn*8))/3.6;       %Desired velocity: 120 km/h with↙
normal distribution with variance 8
394             v_new = (80+(randn*6))/3.6;         %Actual incoming velocity: 80 km/h↙
with normal distribution with variance 6
395             T_new = 2+(randn*0.4);              %Safe time headway: 2 s with normal↙
distribution with variance 0.4
396             dv = v_new - state_2(size(state_2,1), iv);
397             ss = s0 + max(v_new*T_new + (v_new*dv)/(2*sqrt(a_c*b_c)), 0);
398
399             if(state_2(size(state_2,1),ix)>ss && (time-pitnc_2)>nct_2)
400                 new_vehicle = [0 v_new 0 v0_new T_new 0 1 2 0 0];        %Create↙
new vehicle [x, v, acc, v0, T, changedposition, type(1:car; 2:truck), original lane,↙
disturbance]
401
402                 state_2 = [state_2; new_vehicle];    %Add new vehicle
403
404                 pitnc_2 = time;
405                 ec2 = ec2 + 1;
406
407                 %Determine type of next new vehicle
408                 if(rand>0.2)
409                         type=1;
410                     else
411                         type=2;
412                 end
413
414             end
415
416         case 2
417             v0_new = (80+(randn*4))/3.6;       %Desired velocity: 80 km/h with↙
normal distribution with variance 4
418             T_new = 3+(randn*0.4);             %Safe time headway: 2 s with normal↙
distribution with variance 0.4
419             dv = v0_new - state_2(size(state_2,1), iv);
420             ss = s0 + max(v0_new*T_new + (v0_new*dv)/(2*sqrt(a_t*b_t)), 0);
421
422             if(state_2(size(state_2,1),ix)>ss && (time-pitnc_2)>nct_2)
423                 new_vehicle = [0 v0_new 0 v0_new T_new 0 2 2 0 0];        %Create↙
new vehicle [x, v, acc, v0, T, changedposition, type(1:car; 2:truck), original lane,↙
disturbance]
424
425                 state_2 = [state_2; new_vehicle];    %Add new vehicle
426
427                 pitnc_2 = time;
428                 ec2 = ec2 + 1;
429
430                 %Determine type of next new vehicle
431                 if(rand>0.2)
432                         type=1;
433                     else
434                         type=2;
435                 end
436
437             end
438
439     end
440
```

```
441     %Disturbance and Speed Limit
442     pos_dist_start = l_highway * 0.5;         %Starting point of disturbance
443     pos_dist_end = pos_dist_start + 100;      %End point of disturbance
444     time_dist_start = simend * 0.2;           %Start time of disturbance
445     time_dist_end = time_dist_start + 300;    %End time of disturbance
446
447     pos_limit_start = l_highway * 0.3;        %Starting point of speed limit
448     pos_limit_end = pos_dist_end + 1000;      %End point of speed limit
449     time_limit_start = time_dist_start + 60;  %Start time of speed limit
450     time_limit_end = time_dist_end + 600;     %End time of speed limit
451
452     limit_factor = speed_limit/120;           %Speed limit factor
453     dist_factor_l = dist_factor/limit_factor; %Disturbance factor with speed limit
454
455     %Speed limit
456     if(speed_limit > 0)
457         %Lane1
458         for vehicle_1 = n_1:size(state_1,1)
459
460             if(time > time_limit_start && time < time_limit_end && state_1↙
(vehicle_1, idili) == 0 && state_1(vehicle_1, ix) > pos_limit_start && state_1↙
(vehicle_1, ix) < pos_limit_end)
461                 state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) * limit_factor;↙
%Decrease velocity
462                 state_1(vehicle_1, idili) = 1;
463             end
464
465             if(state_1(vehicle_1, idili) == 1 && (not(time > time_limit_start &&↙
time < time_limit_end) || not(state_1(vehicle_1, ix) > pos_limit_start && state_1↙
(vehicle_1, ix) < pos_limit_end)))
466                 state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
(1/limit_factor);    %Increase velocity
467                 state_1(vehicle_1, idili) = 0;
468             end
469         end
470
471         %Lane2
472         for vehicle_2 = n_2:size(state_2,1)
473
474             if(time > time_limit_start && time < time_limit_end && state_2↙
(vehicle_2, itype) == 1 && state_2(vehicle_2, idili) == 0 && state_2(vehicle_2, ix) >↙
pos_limit_start && state_2(vehicle_2, ix) < pos_limit_end)
475                 state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) * limit_factor;↙
%Decrease velocity
476                 state_2(vehicle_2, idili) = 1;
477             end
478
479             if(state_2(vehicle_2, itype) == 1 && state_2(vehicle_2, idili) == 1 &&↙
(not(time > time_limit_start && time < time_limit_end) || not(state_2(vehicle_2, ix) >↙
pos_limit_start && state_2(vehicle_2, ix) < pos_limit_end)))
480                 state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
(1/limit_factor);    %Increase velocity
481                 state_2(vehicle_2, idili) = 0;
482             end
483         end
484     end
485
486     %Disturbance
487     switch disturbance
488         case 1
```

```
489             %Lane1
490             for vehicle_1 = n_1:size(state_1,1)
491
492                  if(state_1(vehicle_1, idili) == 2)
493                      state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
(1/dist_factor);       %Increase velocity without speed limit
494                      state_1(vehicle_1, idili) = 0;
495                  end
496                  if(state_1(vehicle_1, idili) == 3)
497                      state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
(1/dist_factor_l);     %Increase velocity with speed limit
498                      state_1(vehicle_1, idili) = 1;
499                  end
500
501             end
502
503             %Lane2
504             for vehicle_2 = n_2:size(state_2,1)
505
506                 if(time > time_dist_start && time < time_dist_end && state_2↙
(vehicle_2, itype) == 1 && state_2(vehicle_2, ix) > pos_dist_start && state_2↙
(vehicle_2, ix) < pos_dist_end)
507                     if(state_2(vehicle_2, idili) == 0)
508                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
dist_factor;          %Decrease velocity without speed limit
509                         state_2(vehicle_2, idili) = 2;
510                     end
511                     if(state_2(vehicle_2, idili) == 1)
512                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
dist_factor_l;        %Decrease velocity with speed limit
513                         state_2(vehicle_2, idili) = 3;
514                     end
515                 end
516
517                 if(state_2(vehicle_2, itype) == 1 && (not(time > time_dist_start &&↙
time < time_dist_end) || not(state_2(vehicle_2, ix) > pos_dist_start && state_2↙
(vehicle_2, ix) < pos_dist_end)))
518                      if(state_2(vehicle_2, idili) == 2)
519                          state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
(1/dist_factor);     %Increase velocity without speed limit
520                          state_2(vehicle_2, idili) = 0;
521                      end
522                      if(state_2(vehicle_2, idili) == 3)
523                          state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
(1/dist_factor_l);   %Increase velocity with speed limit
524                          state_2(vehicle_2, idili) = 1;
525                      end
526                 end
527             end
528
529         case 2
530             %Lane1
531             for vehicle_1 = n_1:size(state_1,1)
532
533                 if(time > time_dist_start && time < time_dist_end && state_1↙
(vehicle_1, itype) == 1 && state_1(vehicle_1, ix) > pos_dist_start && state_1↙
(vehicle_1, ix) < pos_dist_end)
534                     if(state_1(vehicle_1, idili) == 0)
535                         state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
dist_factor;          %Decrease velocity without speed limit
```

```
536                         state_1(vehicle_1, idili) = 2;
537                     end
538                     if(state_1(vehicle_1, idili) == 1)
539                         state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
dist_factor_l;        %Decrease velocity with speed limit
540                         state_1(vehicle_1, idili) = 3;
541                     end
542                 end
543
544                 if(state_1(vehicle_1, itype) == 1 && (not(time > time_dist_start &&↙
time < time_dist_end) || not(state_1(vehicle_1, ix) > pos_dist_start && state_1↙
(vehicle_1, ix) < pos_dist_end)))
545                     if(state_1(vehicle_1, idili) == 2)
546                         state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
(1/dist_factor);     %Increase velocity without speed limit
547                         state_1(vehicle_1, idili) = 0;
548                     end
549                     if(state_1(vehicle_1, idili) == 3)
550                         state_1(vehicle_1, iv0) = state_1(vehicle_1, iv0) *↙
(1/dist_factor_l);   %Increase velocity with speed limit
551                         state_1(vehicle_1, idili) = 1;
552                     end
553                 end
554             end
555
556             %Lane2
557             for vehicle_2 = n_2:size(state_2,1)
558
559                 if(time > time_dist_start && time < time_dist_end && state_2↙
(vehicle_2, itype) == 1 && state_2(vehicle_2, ix) > pos_dist_start && state_2↙
(vehicle_2, ix) < pos_dist_end)
560                     if(state_2(vehicle_2, idili) == 0)
561                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
dist_factor;          %Decrease velocity without speed limit
562                         state_2(vehicle_2, idili) = 2;
563                     end
564                     if(state_2(vehicle_2, idili) == 1)
565                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
dist_factor_l;        %Decrease velocity with speed limit
566                         state_2(vehicle_2, idili) = 3;
567                     end
568                 end
569
570                 if(state_2(vehicle_2, itype) == 1 && (not(time > time_dist_start &&↙
time < time_dist_end) || not(state_2(vehicle_2, ix) > pos_dist_start && state_2↙
(vehicle_2, ix) < pos_dist_end)))
571                     if(state_2(vehicle_2, idili) == 2)
572                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
(1/dist_factor);     %Increase velocity without speed limit
573                         state_2(vehicle_2, idili) = 0;
574                     end
575                     if(state_2(vehicle_2, idili) == 3)
576                         state_2(vehicle_2, iv0) = state_2(vehicle_2, iv0) *↙
(1/dist_factor_l);   %Increase velocity with speed limit
577                         state_2(vehicle_2, idili) = 1;
578                     end
579                 end
580             end
581
582         case 3
```

```
583             %Lane 1
584             for vehicle_1 = n_1:size(state_1,1)
585
586                 if(time > time_dist_start && time < time_dist_end && state_1↙
(vehicle_1, ix) > pos_dist_start && state_1(vehicle_1, ix) < pos_dist_end)
587                     if(state_1(vehicle_1, idili) == 0)
588                         state_1(vehicle_1, iv0) = 1;          %Decrease velocity↙
(~= 1 m/s)
589                         state_1(vehicle_1, idili) = 2;
590                     end
591                     if(state_1(vehicle_1, idili) == 1)
592                         state_1(vehicle_1, iv0) = 1;          %Decrease velocity↙
(~= 1 m/s)
593                         state_1(vehicle_1, idili) = 3;
594                     end
595                 end
596
597                 if(not(time > time_dist_start && time < time_dist_end) || state_1↙
(vehicle_1, ix) > pos_dist_end)
598                     if(state_1(vehicle_1, idili) == 2)
599                         state_1(vehicle_1, iv0) = (120+(randn*8))/3.6;     %↙
Increase velocity
600                         state_1(vehicle_1, idili) = 0;
601                     end
602                     if(state_1(vehicle_1, idili) == 3)
603                         state_1(vehicle_1, iv0) = (120+(randn*8))/3.6 *↙
limit_factor;     %Increase velocity
604                         state_1(vehicle_1, idili) = 1;
605                     end
606                 end
607             end
608
609             %Lane 2
610             for vehicle_2 = n_2:size(state_2,1)
611
612                 if(time > time_dist_start && time < time_dist_end && state_2↙
(vehicle_2, ix) > pos_dist_start && state_2(vehicle_2, ix) < pos_dist_end)
613                     if(state_2(vehicle_2, idili) == 0)
614                         state_2(vehicle_2, iv0) = 1;          %Decrease velocity↙
(~= 1 m/s)
615                         state_2(vehicle_2, idili) = 2;
616                     end
617                     if(state_2(vehicle_2, idili) == 1)
618                         state_2(vehicle_2, iv0) = 1;          %Decrease velocity↙
(~= 1 m/s)
619                         state_2(vehicle_2, idili) = 3;
620                     end
621                 end
622
623                 if(not(time > time_dist_start && time < time_dist_end) || state_2↙
(vehicle_2, ix) > pos_dist_end)
624                     if(state_2(vehicle_2, idili) == 2)
625                         if(state_2(vehicle_2, itype) == 1)
626                            state_2(vehicle_2, iv0) = (120+(randn*8))/3.6;      %↙
Increase velocity of a car
627                         else
628                            state_2(vehicle_2, iv0) = (80+(randn*4))/3.6;       %↙
Increase velocity of a truck
629                         end
630                         state_2(vehicle_2, idili) = 0;
```

```
631                     end
632                     if(state_2(vehicle_2, idili) == 3)
633                         if(state_2(vehicle_2, itype) == 1)
634                            state_2(vehicle_2, iv0) = (120+(randn*8))/3.6 *↙
limit_factor;      %Increase velocity of a car
635                         else
636                            state_2(vehicle_2, iv0) = (80+(randn*4))/3.6 *↙
limit_factor;       %Increase velocity of a truck
637                         end
638                         state_2(vehicle_2, idili) = 1;
639                     end
640                 end
641
642             end
643     end
644
645     %Percentage of the progress
646     perpro = time/simend*100;
647     if(mod(perpro, 1) == 0)
648         %clc
649         disp(['Progress: ' num2str(perpro) '%'])
650     end
651
652     %Check of state_1 sort
653     for vehicle_1=n_1:size(state_1,1)
654         if(state_1(vehicle_1, ix) > state_1(vehicle_1-1, ix) && size(state_1,1) >↙
1)
655             disp('Attention: state_1 not sorted well!!!')
656             crash = 1;
657         end
658     end
659
660     %Check of state_2 sort
661     for vehicle_2=n_2:size(state_2,1)
662         if(state_2(vehicle_2, ix) > state_2(vehicle_2-1, ix) && size(state_2,1) >↙
1)
663             disp('Attention: state_2 not sorted well!!!')
664             crash = 1;
665         end
666     end
667
668     %Save states every second
669     if (mod(time, 1) == 0)
670         %save data\[statefile 'num2str(time)'] time state_1 state_2
671         save([dir '/statefile_' num2str(time)], 'time', 'state_1', 'state_2',↙
'ec1', 'ec2', 'lc1', 'lc2', 'av_time_c', 'av_time_t', 'lcc', 'ltc', 'crash')
672     end
673
674     %Break if crash
675     if(crash)
676         break;
677     end
678
679 end
680
681 end
```

9.2 plot graphics

```
function plot_graphics(simtime, l_highway_start, l_highway_end, withDensity, withVelocity, withVehicleCounter)
%plot_graphics(simtime, l_highway_start, l_highway_end, withDensity, withVelocity, withVehicleCounter)
%plots the traffic density (vehicles/km), the average speed and the outgoing
%traffic flow over a section of highway from l_highway_start to l_highway_end

%Indexes of the state matrix
ix = 1;         %Position
iv = 2;         %Velocity
iacc = 3;       %Acceleration
iv0 = 4;        %Desired velocity
iT = 5;         %Safe time headway
icp = 6;        %Changed position
itype = 7;      %Type 1:car; 2:truck
iol = 8;        %Original lane

%Define Parameters
l_highway = l_highway_end - l_highway_start;
N = 100;                            %space resolution
x = linspace(l_highway_start,l_highway_end,5*N+1);  %evaluating points
time_vec = linspace(0,simtime,5*N+1);               %time vector
time_vec2 = linspace(0,simtime,N/2+1);              %time vector
Mat_density1 = zeros(5*N+1,5*N+1);  %Spatiotemporal density Matrix lane 1
Mat_density2 = zeros(5*N+1,5*N+1);  %Spatiotemporal density Matrix lane 2
Mat_velocity1 = zeros(5*N+1,5*N+1); %Spatiotemporal velocity Matrix lane 1
Mat_velocity2 = zeros(5*N+1,5*N+1); %Spatiotemporal velocity Matrix lane 2
Mat_ec = zeros(N/2+1,2);            %Entering cars
Mat_lc = zeros(N/2+1,2);            %Leaving cars
ec1_temp = 0;
ec2_temp = 0;
lc1_temp = 0;
lc2_temp = 0;

    %Calculate density
    for timestep = 0:1:5*N

        % load statefile
        time = round(time_vec(timestep+1));
        load(['data/statefile_' num2str(time)])
        if(withDensity)
            for place = 1:(5*N+1)
                counter = 0;
                for vehicle_1 = 1:size(state_1,1)

                    if(state_1(vehicle_1,ix) < x(place) - 500)
                        break
                    end
                    if(state_1(vehicle_1,ix) < x(place) + 500)
                        counter = counter + 1;
                    end

                end
                Mat_density1(timestep+1,place) = counter;
                counter = 0;
                for vehicle_2 = 1:size(state_2,1)

                    if(state_2(vehicle_2,ix) < x(place) - 500)
                        break
                    end
```

```
59                  if(state_2(vehicle_2,ix) < x(place) + 500)
60                      counter = counter + 1;
61                  end
62
63              end
64              Mat_density2(timestep+1,place) = counter;
65          end
66      end
67
68      if(withVelocity)
69          for place = 1:(5*N+1)
70
71              counter = 0;
72              av_v = 0;
73              for vehicle_1 = 1:size(state_1,1)
74
75                  if(state_1(vehicle_1,ix) < x(place) - 100)
76                          break
77                  end
78                  if(state_1(vehicle_1,ix) < x(place) + 100)
79                          counter = counter + 1;
80                          av_v = av_v + state_1(vehicle_1,iv);
81                  end
82
83              end
84              if(counter)
85                  Mat_velocity1(timestep+1,place) = av_v/counter;
86              else
87                  Mat_velocity1(timestep+1,place) = 0;
88              end
89              counter = 0;
90              av_v = 0;
91              for vehicle_2 = 1:size(state_2,1)
92
93                  if(state_2(vehicle_2,ix) < x(place) - 100)
94                          break
95                  end
96                  if(state_2(vehicle_2,ix) < x(place) + 100)
97                          counter = counter + 1;
98                          av_v = av_v + state_2(vehicle_2,iv);
99                  end
100
101             end
102             if(counter)
103                 Mat_velocity2(timestep+1,place) = av_v/counter;
104             else
105                 Mat_velocity2(timestep+1,place) = 0;
106             end
107         end
108     end
109
110     if(withVehicleCounter)
111         if(mod(timestep,10) == 0)
112         if(time)
113             %Entering cars
114             Mat_ec(timestep/10+1,1) = ec1 - ec1_temp;
115             Mat_ec(timestep/10+1,2) = ec2 - ec2_temp;
116             ec1_temp = ec1;
117             ec2_temp = ec2;
118             %Leaving cars
```

```
                Mat_lc(timestep/10+1,1) = lc1 - lc1_temp;
                Mat_lc(timestep/10+1,2) = lc2 - lc2_temp;
                lc1_temp = lc1;
                lc2_temp = lc2;
            end
            end
        end

        %Percentage of the progress
        perpro = timestep/(5*N)*100;
        if(mod(perpro, 1) == 0)
            clc
            disp(['Progress: ' num2str(perpro) '%'])
        end

        if(time == simtime)
            avtc = av_time_c/lcc;
            avtt = av_time_t/ltc;
            disp(['The average transit time for a car is: ' num2str(avtc)]);
            disp(['The average transit time for a truck is: ' num2str(avtt)])
        end
    end

    Mat_ec = Mat_ec/(simtime/(5*N));
    Mat_lc = Mat_lc/(simtime/(5*N));

    if(withDensity)
        % plot density
        figure
        surf(x, time_vec, Mat_density1, 'EdgeColor', 'none');
        set(gca, 'Fontsize', 14)
        xlabel('position');
        ylabel('time');
        title('density lane 1 (vehicle/km)');
        colorbar;
        figure
        surf(x, time_vec, Mat_density2, 'EdgeColor', 'none');
        set(gca, 'Fontsize', 14)
        xlabel('position');
        ylabel('time');
        title('density lane 2 (vehicle/km)');
        colorbar;
    end

    if(withVelocity)
        % plot velocity
        figure
        hold on
        view(0,90);
        surf(x, time_vec, Mat_velocity1, 'EdgeColor', 'none');
        set(gca, 'Fontsize', 14)
        xlabel('position');
        ylabel('time');
        title('velocity lane 1 (m/s)');
        colorbar;
        figure
        hold on
        view(0,90);
        surf(x, time_vec, Mat_velocity2, 'EdgeColor', 'none');
        set(gca, 'Fontsize', 14)
```

```
179         xlabel('position');
180         ylabel('time');
181         title('velocity lane 2 (m/s)');
182         colorbar;
183     end
184 
185     if(withVehicleCounter)
186         %Incoming traffic flow
187         ictf = sum(Mat_ec,1)/(5*N)*3600
188         %Outgoing traffic flow
189         ogtf = sum(Mat_lc,1)/(5*N)*3600
190         %plot number of entering cars
191         figure
192         bar(time_vec2,Mat_ec, 'hist')
193         xlabel('time', 'Fontsize', 14);
194         ylabel('Number of entering cars per 10s', 'Fontsize', 14);
195         title('incoming trafficflow', 'Fontsize', 16)
196         legend('lane 1', 'lane 2')
197         set(gca, 'XLim', [0 simtime], 'Fontsize', 12)
198         %plot number of leaving cars
199         figure
200         bar(time_vec2,Mat_lc, 'hist')
201         xlabel('time', 'Fontsize', 14);
202         ylabel('Number of leaving cars per 10s', 'Fontsize', 14);
203         title('outgoing trafficflow', 'Fontsize', 16)
204         legend('lane 1', 'lane 2')
205         set(gca, 'XLim', [0 simtime], 'Fontsize', 12)
206     end
207 end
```

9.3 plot simulation

```
function plot_simulation(simtime, l_highway_start, l_highway_end)
% plot_simulation(simtime, l_highway_start, l_highway_end)
% plots the simulation of the highway for the duration of simtime
% and the section of highway from l_highway_start to l_highway_end

%Indexes of the state matrix
ix = 1;         %Position
iv = 2;         %Velocity
iacc = 3;       %Acceleration
iv0 = 4;        %Desired velocity
iT = 5;         %Safe time headway
icp = 6;        %Changed position
itype = 7;      %Type 1:car; 2:truck
iol = 8;        %Original lane

l_highway = l_highway_end - l_highway_start;

    for time = 0:1:simtime

        %Load statefile
        load(['data/statefile_' num2str(time)])
            %Plot street
                clf;
                y_dist=0.5;
                plot(l_highway_start:l_highway_end, 0*(0:l_highway), 'Color', [.75 .↙
75 .75], 'LineWidth', 600)
                hold on;
                plot([l_highway_start,l_highway_end] ,[0,0],'--w','LineWidth', 1)
                hold on;
                plot([l_highway_start,l_highway_end] ,[-y_dist,-↙
y_dist],'k','LineWidth', 2)
                hold on;
                plot([l_highway_start,l_highway_end] ,[y_dist,↙
y_dist],'k','LineWidth', 2)
                xlim([l_highway_start l_highway_end])
                ylim([-1.5*y_dist 1.5*y_dist])
                xlabel('Position [m]')
             %Plot cars
             for i = 1:size(state_1)

                 if (state_1(i,icp)==0)
                    draw_car_diff_colors(state_1(i,ix), y_dist/2, 10, 4, state_1(i,↙
itype), state_1(i,iol));
                 else
                    draw_car_diff_colors(state_1(i,ix), 0, 10, 4, state_1(i,itype),↙
state_1(i,iol));
                 end

             end
             for i = 1:size(state_2)

                 if (state_2(i,icp)==0)
                    draw_car_diff_colors(state_2(i,ix), -y_dist/2, 10, 4, state_2(i,↙
itype), state_2(i,iol));
                 else
                    draw_car_diff_colors(state_2(i,ix), 0, 10, 4, state_2(i,itype),↙
state_2(i,iol));
                 end

             end
```

```
54              % plot time
55              text(l_highway_start + 4/5*l_highway, 1.15*y_dist, ['time= ' num2str↙
(time)])
56              pause(0.05)
57      end
58  end
```

9.4 draw car diff colors

```
function draw_car(x0, y0, w, h, t, c)

% This function is drawing a car
% INPUT:
% x0, y0: Central point of the car
% w: Width of the car
% h: Height of the car
% Hold the graphics

hold on

% Define the coordinates for the car chassi
chassi_x = [w/2 w/2 2*w/6 w/6 -2*w/6 -w/2 -w/2];
chassi_y = [0 h/2 h/2 h h h/2 0]*0.01;
chassi_l_x = [3/2*w 3/2*w -w/2 -w/2];
chassi_l_y = [0 3/2*h 3/2*h 0]*0.01;
chassi_cl_x = [3/2*w 3/2*w 2*w 2*w];
chassi_cl_y = [0 5/4*h 5/4*h 0]*0.01;

% Define the coordinates for the wheels
angles = 0:0.1:(2*pi);
r = sqrt(w*w + h*h)/1000;
wheel_x = r*cos(angles)*100;
wheel_y = r*sin(angles);

% Draw the car!
if(t==1)
    if(c==1)
        patch(x0+chassi_x, y0+chassi_y, 'r')
        patch(x0-w/4+wheel_x, y0+wheel_y, 'k')
        patch(x0+w/4+wheel_x, y0+wheel_y, 'k')
    end
    if(c==2)
        patch(x0+chassi_x, y0+chassi_y, 'g')
        patch(x0-w/4+wheel_x, y0+wheel_y, 'k')
        patch(x0+w/4+wheel_x, y0+wheel_y, 'k')
    end
end
%if(c==0)
%patch(x0+chassi_x, y0+chassi_y, 'y')
%end
if(t==2)
patch(x0+chassi_l_x, y0+chassi_l_y, 'b')
patch(x0+chassi_cl_x, y0+chassi_cl_y, [1,0,1])
patch(x0-w/4+wheel_x, y0+wheel_y, 'k')
patch(x0+wheel_x, y0+wheel_y, 'k')
patch(x0+7/4*w+wheel_x, y0+wheel_y, 'k')
end

```

9.5 run simulation

```
%Simulation file
close all;
clear all;
clc;
%Denomination of saved files:
%Statefile_ d2_ sl_ n_ time

N = 20;                 %Number of simulations
simtime = 2000;         %Simulation time
l_highway = 10000;      %Length of simulated highway
avtc_Mat = zeros(20,3); %Matrix: Average transit time car
avtt_Mat = zeros(20,3); %Matrix: Average transit time truck
ictf_Mat = zeros(20,6); %Matrix: Incoming traffic flow
ogtf_Mat = zeros(20,6); %Matrix: Outgoing traffic flow


%Disturbance on (dist_factor = 0.6)
d = 2;                      %Disturbancetype
speed_limit = [0 100 80];


for k=1:size(speed_limit,2)
    sl = speed_limit(k);
    for n=1:20
        dir = ['data/dataset_d2_sl' num2str(sl) '_' num2str(n)];
        mkdir(dir)
        simulate_main(simtime,l_highway,d,sl,dir,0)
        load([dir '/statefile_' num2str(simtime)])
        if(crash == 0)
            [avtc, avtt, ictf, ogtf] = evaluate_simulation(dir, 2000, 10000);
            avtc_Mat(n,k) = avtc;
            avtt_Mat(n,k) = avtt;
            ictf_Mat(n,2*k-1:2*k) = ictf;
            ogtf_Mat(n,2*k-1:2*k) = ogtf;
        end
    end
end
```

9.6 evaluate simulation

```
function [avtc, avtt, ictf, ogtf]=evaluate_simulation(dir, simtime, l_highway)
%evaluate_simulation(dir, simtime, l_highway)
%avtc: average transit time for a car
%avtt: average transit time for a truck
%ictf: incoming traffic flow [vehicle/h]
%ogtf: outgoing traffic flow [vehicle/h]
%dir: Direction

%Define Parameters
N = 100;                                %Space resolution
x = linspace(0,l_highway,5*N+1);        %Evaluating points
time_vec = linspace(0,simtime,5*N+1); %Time vector
Mat_ec = zeros(N/2+1,2);                %Entering cars
Mat_lc = zeros(N/2+1,2);                %Leaving cars
ec1_temp = 0;
ec2_temp = 0;
lc1_temp = 0;
lc2_temp = 0;

    %Calculate density
    for timestep = 0:1:5*N

        %Load statefile
        time = round(time_vec(timestep+1));
        load([dir '/statefile_' num2str(time)])
        if (mod(timestep,10) == 0)
            if (time)
                %Entering cars
                Mat_ec(timestep/10+1,1) = ec1 - ec1_temp;
                Mat_ec(timestep/10+1,2) = ec2 - ec2_temp;
                ec1_temp = ec1;
                ec2_temp = ec2;
                %Leaving cars
                Mat_lc(timestep/10+1,1) = lc1 - lc1_temp;
                Mat_lc(timestep/10+1,2) = lc2 - lc2_temp;
                lc1_temp = lc1;
                lc2_temp = lc2;
            end

        end

        if(time == simtime)
            avtc = av_time_c/lcc;
            avtt = av_time_t/ltc;
        end

    end

    Mat_ec = Mat_ec/(simtime/(5*N));
    Mat_lc = Mat_lc/(simtime/(5*N));
    %Incoming traffic flow
    ictf = sum(Mat_ec,1)/(5*N)*3600;
    %Outgoing traffic flow
    ogtf = sum(Mat_lc,1)/(5*N)*3600;

end
```