



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

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

Project Report

**Traffic Dynamics**

Traffic Flow Comparison of Roundabouts and Crossroads

Tony Wood & Bastian Bücheler

Zurich  
20. May 2010

## **Eigenständigkeitserklärung**

Hiermit erkläre ich, dass ich diese Gruppenarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen-Hilsmittel verwenden habe, und alle Stellen, die wörtlich oder sinngemäss aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Gruppenarbeit nicht, auch nicht auszugsweise, bereits für andere Prüfung ausgefertigt wurde.

Tony Wood

Bastian Bücheler

## **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.

Tony Wood

Bastian Bücheler

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Nagel-Schreckenberg Model</b>	<b>7</b>
<b>3</b>	<b>Approach</b>	<b>8</b>
3.1	Description . . . . .	8
3.1.1	Crossroad with priority to the right . . . . .	8
3.1.2	Roundabout . . . . .	9
3.2	Implementation . . . . .	10
3.2.1	Crossroad . . . . .	11
3.2.2	Roundabout . . . . .	13
3.3	Execution . . . . .	14
3.3.1	User Instructions . . . . .	15
<b>4</b>	<b>Results</b>	<b>16</b>
<b>5</b>	<b>Discussion</b>	<b>20</b>
5.1	Comparison . . . . .	20
5.2	Validation of Simplification . . . . .	21
<b>6</b>	<b>Summary and Outlook</b>	<b>22</b>
<b>7</b>	<b>References and Code</b>	<b>23</b>
7.1	References . . . . .	23
7.2	Matlab-Code . . . . .	24
7.2.1	traffic.m . . . . .	24
7.2.2	trafficsim.m . . . . .	27
7.2.3	roundabout.m . . . . .	36
7.2.4	crossroad.m . . . . .	40
7.2.5	connection.m . . . . .	58
7.2.6	pdestination.m . . . . .	60

# 1 Introduction

Nowadays, the streets get more and more crowded as more people own a car. Consequently mankind is facing several problems in managing the extra traffic. With this in mind, it is interesting and at the same time important to look at different solutions of intersections, where many of these problems occur.

Luka Piskorec and Simon Soller covered a similar approach in last years course. They modelled the traffic with the help of a cellular automata in MATLAB and they did two intersection types, namely priority to the right and signals. In between the intersections they modeled the streets with the Nagel-Schreckenberg model, which takes in to account many social effects and is at the same time relatively easy to implement. We build on their experience and add several new aspects; one of which will be the implementation of a roundabout. This type of intersection is pretty common in Switzerland. In addition we give the cars the freedom of choosing a direction, so they don't just travel from left to right or top to bottom and vice versa.

With these changes we think we can provide a model the traffic in Switzerland in a more realistic manner. Since we develop a new model of an intersection as well as more freedom of individual cars, we could not use our predecessor's code. Therefore we wrote a new program. We do, however, compare our model with the one from our predecessors through a common feature, i.e. the priority to the right intersection.

The aim of our research is to analyze the traffic flow of different intersection configurations. Ultimately, we want to determine which type of intersection gives the highest traffic flow. Since the roundabout has become popular in recent years, we expect the it to preform best.



Figure 1: Famous roundabout in England

## 2 Nagel-Schreckenberg Model

We decided to use the Nagel-Schreckenberg Model<sup>1</sup> to simulate our traffic. This Model was developed in the early 1990 by two physicist Kai Nagel and Michael Schreckenberg. It was the first to explain how a traffic jam can evolve from nothing.

The Model is straight forward and intuitive to use. The streets get split up in cells about the size of a car. One car is approximately 7.5 meter long. In the MATLAB implementation, one second corresponds to one iteration. If the car travels one cell at a second, it corresponds to 7.5 m/s or 27 km/h. This gives us the maximum speed which is 5 cells per second or 135 km/h.

Each car then has to follow four simple rules:

1. If the maximum speed is not yet achieved, it has to accelerate by one.
2. If the gap in front of the car is smaller than the actual speed, it has to match the speed to the gap.
3. The speed of the car gets reduced by one with a probability  $p$ , unless it is already standing still. This is called dawdling.
4. In the end, when everything is calculated, the cars move by their current speed.

With the third rule one can model three social phenomena at the same time:

1. A car that could accelerate, because it didn't reach the maximal speed yet and because the gap in front of it is big enough, doesn't take the opportunity because of dawdling.
2. A car that already drives with full speed, can fall back. We get fluctuations in the top speed segment.
3. A car that has to slowdown because of a car in front of it, could slow down even further through dawdling.

The model is minimal, which means that it can't be simplified further without losing essential characteristics of traffic dynamics. The model does not include passing or accidents.

---

<sup>1</sup>see <http://de.wikipedia.org/wiki/Nagel-Schreckenberg-Modell>

## 3 Approach

### 3.1 Description

Our Model of city traffic covers two different intersections, a crossroad with priority to the right and a roundabout. These intersections can be arbitrarily distributed on a  $m \times n$  grid and are connected by streets (i.e. see figures 4,5,8 and 10). The streets consist of two lanes, one in each direction.

The Model is discrete and follows the principle of cellular automaton. Space is divided up into cells. Each cell is in one of a finite number of states and is updated every time step according to the system dynamics. Outside intersections the states are {'car', 'no car'} combined with a speed state {0,1,2,3,4,5}. At intersections there are additional states. While the cars on the streets behave according to the Nagel-Schreckenberg model, the behavior at the intersections is described below.

Cars leaving the map continue on the opposite side. This guarantees a constant car density. The traffic flow<sup>2</sup> is defined as the product of the traffic density and the average speed of all cars.

$$flow = density \cdot speed$$

#### 3.1.1 Crossroad with priority to the right

There are several rules to follow at a crossroad. A car in front of a crossroad will only enter if there is enough free space. As soon as the vehicle enters the intersection, it indicates in which direction it will to go. The car will go straight ahead with a probability of 1/2 and turn left or right with the probability of 1/4 each. We assume that cars in crossroads have the maximum speed 1 in units of cells per iteration.

The main traffic rule to respect is the priority to the right. A car in a crossroad has to give way to cars coming from its right. It can happen that all cars at a crossroad have a car coming from their right. Priority to the right leads to a deadlock in this case. To resolve this blockage of the intersection, the drivers determine by hand signals who can use the crossing first.

If there are two cars coming from opposite sides of the crossroad and no traffic is coming from their sides, they can drive as long as none of them is turning left. If one of them intends to turn left, it will have to give way before crossing the other's

---

<sup>2</sup>[http://en.wikipedia.org/wiki/Traffic\\_flow](http://en.wikipedia.org/wiki/Traffic_flow)



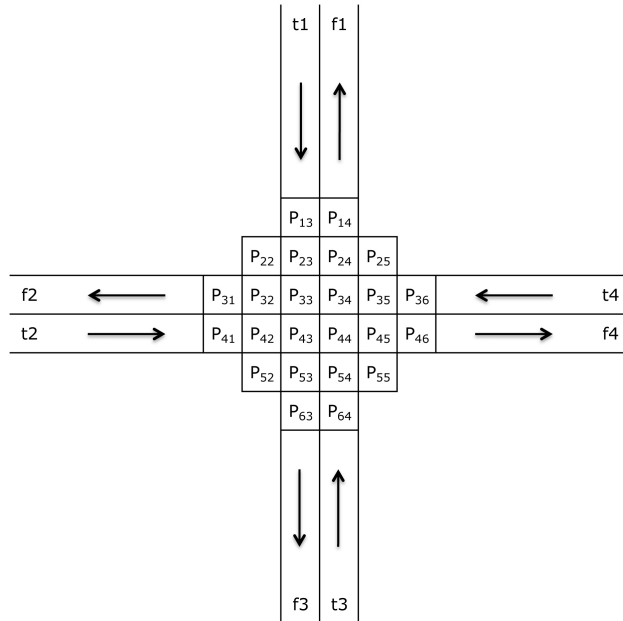


Figure 2: Crossroad cells

lane. In our model the crossroad is big enough for two cars coming from opposite sides to turn left without affecting each other. Thus they can do so at the same time.

Our Model also allows a car turning left to wait in the middle of the crossroad. This way cars behind cars turning left don't get held up as much because they can drive around the waiting vehicle.

A crossroad is made up of 24 cells (see Figure 2). The possible states for each cell are {'car turning right', 'car going straight ahead', 'car turning left', 'no car'} combined with a speed {0,1} and the index of the street a car came from {0,1,2,3,4}.

### 3.1.2 Roundabout

The basic rule in a roundabout is simple. Cars in the roundabout have the right of way over cars wanting to enter the intersection. The maximum speed in the roundabout is again limited to 1 cell per iteration.

If there is no car approaching from inside the roundabout cars waiting will enter. In the roundabout a car will drive whenever there is free space in front of it and eventually leave through an exit. It will take the second exit with a probability of

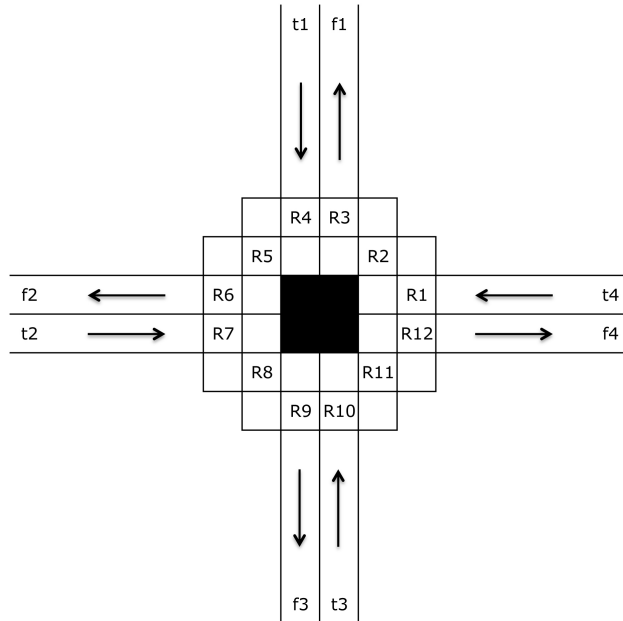


Figure 3: Roundabout cells

1/2. There is a 1/4 probability each that the vehicle takes the first or the third exit. In practice cars can also take the fourth exit, meaning to use the roundabout to perform a U-turn. This option is not included in our default model to make it more comparable to a crossroad.

A car in the roundabout will indicate that it is leaving as soon as the next exit is its destination. No indication means that the car is not taking the next exit.

Roundabouts consist of 12 cells (see Figure 3). The possible states for these are {'car not taking next exit', 'car taking next exit', 'no car'} in combination with a speed {0,1} and the amount of exits a car is going to drive towards before it leaves {0,1,2,3,}.

### 3.2 Implementation

The city map and the traffic density are given by the user. The city map is a  $m \times n$  matrix with elements 0 or 1. 0 symbolizes a roundabout and 1 stands for a crossroad. The density for a single simulation is a scalar. If the density given by the user is a vector, the program will run a simulation for every element of that vector.

Streets are divided into two categories, ones leading towards a intersection and ones leading away form intersections. The end of a street leading away from a intersection is connected to the beginning of the street heading towards the next intersection in the corresponding direction. The overall distance between two intersections is 60 cells which corresponds to 420 meters in practice. Eventually all street and intersections are written into a overall map.

For the initial distribution of cars in the city, the cars get placed on streets only. All intersections are empty at the starting point. The default number of iterations per simulation is set to 1000 which corresponds to just 16 minutes. In every time iteration the values of all cells in the next time step are calculated, the traffic flow is evaluated and if the graphics are activated the current situation is displayed.

For the computation of the updates the program iterates over all intersections. The streets are saved in blocks of four such that they are linked to a certain intersection. In these blocks the row index identifies the street locally. Street numbering runs anti-clockwise and streets above the intersection have index 1 (see figures 2 and 3).

In every intersection, first, the streets get updated according to the rules of Nagel-Schreckenberg (see section 2 on page 7). A car moving along the street is implemented by the cells where the car was passing its state on to the cell where the car will be. The speed state will be the distance these cells are apart. As mentioned above, cars leaving one street continue on another one. Cars leaving the map reappear on the opposite side.

Depending on whether the intersection is a crossroad or a roundabout, different subprograms are called to do the updates in the intersection. Details on this are explained below.

After all iterations over time the average traffic flow is calculated and the simulation is complete. Once all simulations are finished the program will plot the traffic flow versus the density. If the city map was a mix of crossroads and roundabouts, there will be an additional graph comparing the amount of cars in the crossroads and roundabouts. The x-axis for this second plot is again the traffic density.

### **3.2.1 Crossroad**

For the update of a crossroad every one of the 24 cells is written down separately. This is why the code of CROSSROAD is so long. First, the cars waiting in front of

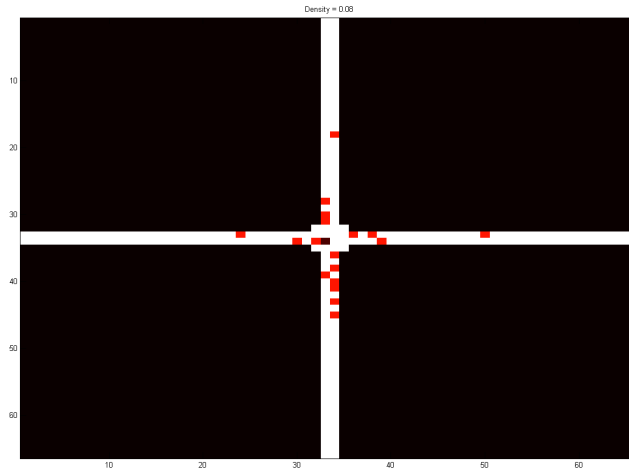


Figure 4: Simulation of a single crossroad

the crossroad are considered. These cars are represented by the last cells of streets heading towards the crossroads in the state ‘car’. If the crossroad is not blocked by other traffic crossing in front and there is enough space for the car to enter, it will do so in the next time step. When entering the car randomly gets a direction assigned and gets marked from which road it came.

Once a car has entered, it will show the other drivers where it intends to go by its indication. If its not indicating the car wants to go straight ahead. Cars turning right do not have to give way to anyone. Cells at the entrance of a crossroad in the state of ‘car turning right’ can therefore be updated by simply checking if there is space free for the car to move on or not.

Cars going straight ahead have to give way to cars coming from the right. This is why cars at the entrance that aren’t indicating need to check more of the traffic situation before they proceed. Before a cell at a entrance releases a car, it makes sure that this car will not block the crossing.

Without any further mechanism, this system would lead to deadlocks when cars from all sides want to go straight at the same time. For this reason, every time a car at the entrance wanting to go straight has to give way, it increases a counter. If the counter reaches 4, a deadlock has occurred. In the next time step, one of the four waiting vehicles will be randomly given the privilege to drive.

A car turning left, in its first step at the entrance, has to check that there is enough space in the middle of the crossroad and that there is no traffic to its left already in the crossroad. If that is the case, the car can move forward and wait in the middle of the crossroad for the traffic coming from the opposite side to clear before it continues.

In the crossroad, a car moving forward means, in terms of the individual cells, that one cell passes its state on to a new cell. Because the maximum speed in intersections is 1 the next cell is always a direct neighbour.

When a car leaves the crossroad the cell at that exit writes a car with speed 1 into the first cell of the linked street leading away from this intersection.

After the updates of all cells are computed, the program writes the current state of the crossroad into the overall map. It also adds the amount of cars in and around this crossroad to a counter. This counter is used to show the relative distributions of cars.

### 3.2.2 Roundabout

The update of the roundabout cells is simpler because there is a periodic pattern. The roundabout can be thought of as a road of length 12 with maximum speed 1. The end of this road is connected to its beginning and every 3 cells there is an exit and an entry (see figure 3).

Cars at the end of streets leading towards a roundabout will enter in the next time step if the cell at that entrance is free and the cell to the left of it isn't in the state 'car not taking next exit'. When a car enters, it randomly gets a number from {1,2,3} which determines the exit that this car is going to take.

If this exit counter is 1, the car is in the state 'car taking next exit'. Thus it is indicating. Once it has reached the next exit, the cell at the exit will, if there is free space, write a car with speed 1 into the linked street heading away from the roundabout.

When a car moves on from an exit cell in the state 'car not taking next exit', its exit counter is decreased. If it now is 1, it turns its state into 'car taking next exit'.

Once all updates have been calculated, the current situation of the roundabout is written into the overall map and the number of cars in and around the roundabout are

added to a counter. This counter is used to show how many car are near roundabouts compared to the amount near crossroads.

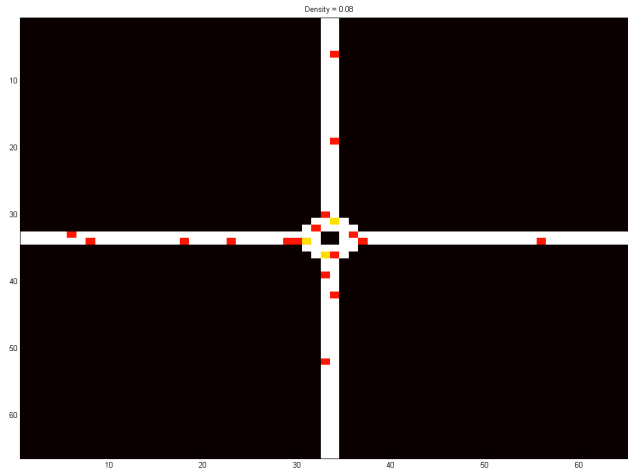


Figure 5: Simulation of a single roundabout

### 3.3 Execution

Our program consists of 6 MATLAB functions. The main function is called TRAFFIC. It is basically the interface between the user and the simulation. TRAFFIC asks the user for input data and starts the simulation accordingly.

TRAFFICSIM runs the simulations. It uses the functions CONNECTION, ROUNDABOUT and CROSSROAD. The input arguments of TRAFFICSIM are a traffic density, a city map configuration matrix and a Boolean telling it if it should display the simulation graphically or not. It returns the average traffic flow and the average number of cars near roundabouts and crossroads respectively.

CONNECTION connects streets. ROUNDABOUT does a updates of a certain roundabout. CROSSROAD runs a update of a specific crossroad. For this it uses the function PDESTINATION.

### 3.3.1 User Instructions

- Include the 6 functions TRAFFIC, TRAFFICSIM, CONNECTION, ROUNDABOUT, CROSSROAD and PDESTINATION in the MATLAB path
- Execute the function TRAFFIC (no arguments).
- Enter city map. City map is a matrix with elements 1 and 0. 1 stands for a crossroad with priority to the right. 0 stands for a roundabout.
- Enter traffic density. If a vector is entered, simulations will run for all elements of this vector.
- Activate graphics by entering 'y'. Deactivate graphics by entering 'n'.
- If graphics were activated simulations will be displayed. In the figure the colour of the cells symbolizes the following:
  - Black → empty space
  - White → road
  - Red → car
  - Yellow → car indicating to the right
  - Dark red → car indicating to the left
- After all simulations have finished the average traffic flow versus the traffic density is plotted. If the city map is a mix of crossroad and roundabouts the traffic distribution (cars around roundabouts or around crossroads) versus traffic density is also plotted.

## 4 Results

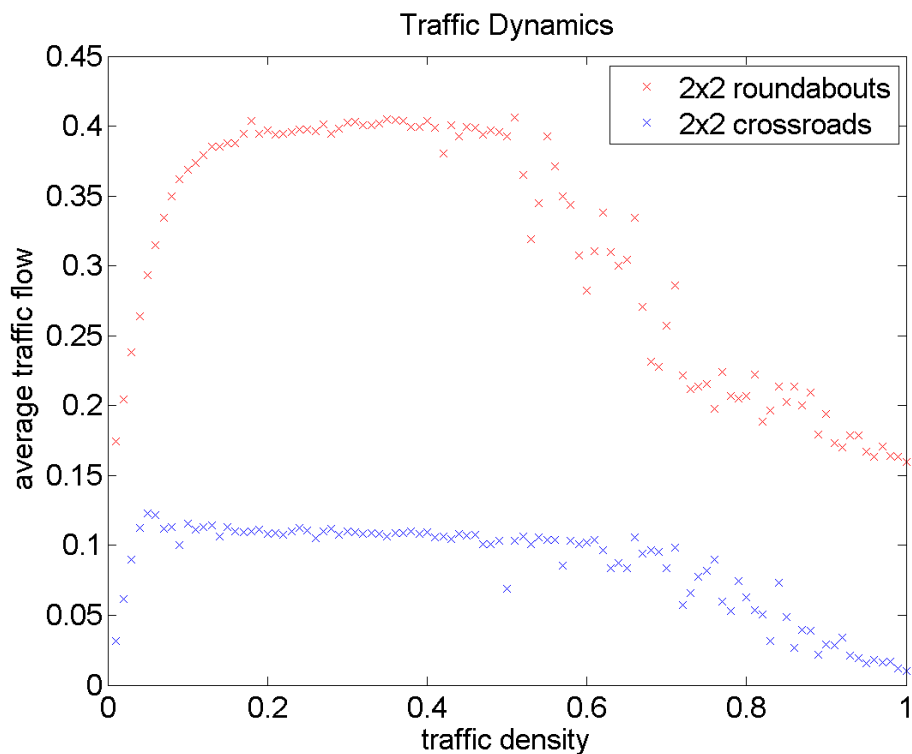


Figure 6: Full density spectrum of  $2 \times 2$  pure roundabout (red) and pure crossroad (blue) configurations

Figure 6 shows the traffic flow of the two square pure city maps containing only one sort of intersection each. The simulation of the configuration with the 4 roundabouts shows a significantly higher traffic flow over the entire density range.

The general behavior of the traffic flow in dependence of the traffic density is to rise steeply in a low densities range. Figure 6 also shows the drop of the traffic flow at high densities. We observe this drop for all configurations at densities above 0.5. The behavior between this steep rise at low densities and drop at high densities appears to be different for the two types of intersections. In both cases the traffic flow doesn't change much in this region. But while the traffic flow continues to climb a little for roundabouts, it decreases slightly for crossroads.

Figure 7 shows the traffic flow of a mixed city map configuration according to fig-



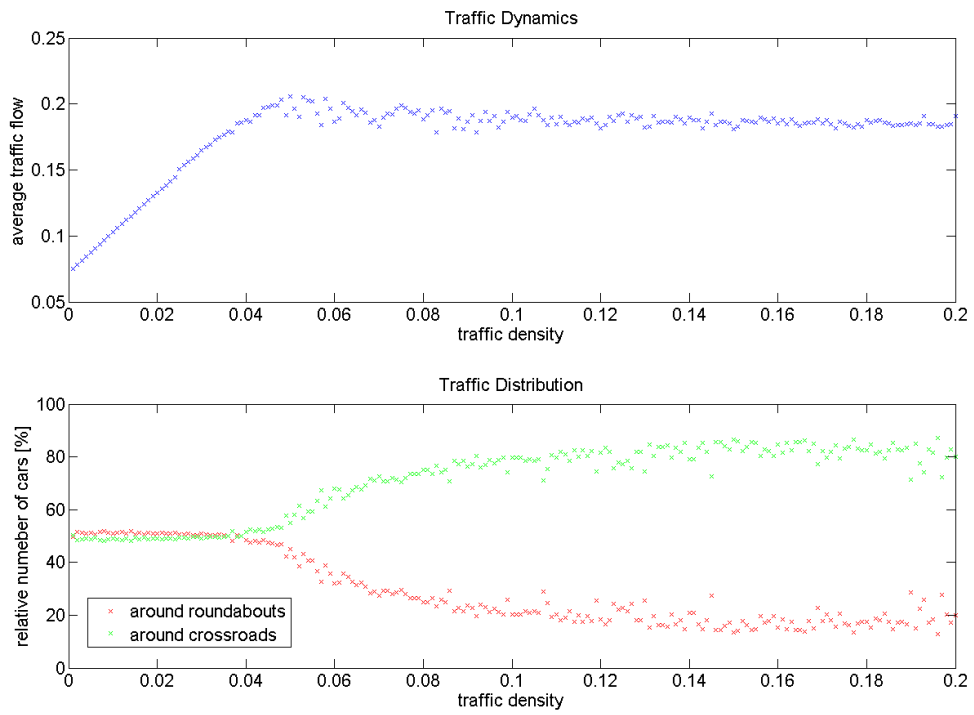


Figure 7: Traffic flow and distribution of the mixed 2x2 intersection configuration from figure 8

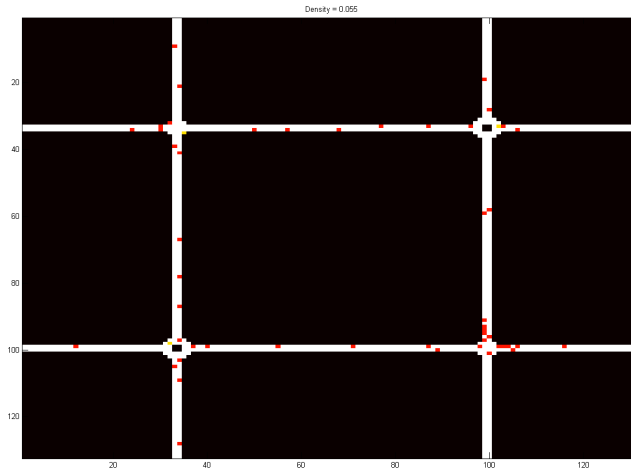


Figure 8: A mixed 2x2 intersection configuration

Figure 8 over the density range  $[0,0.2]$ . At low densities, where the traffic flow increases strongly with the density, the the number of cars around the 2 roundabouts is all most the same as the amount around crossroads. There are a few more cars round the roundabouts though. This distribution changes drastically in higher densities where the traffic flow saturates. At a traffic density of 0.2, approximately 80 percent of the cars are near crossroads.

Figure 9 shows that the configuration of figure 10 has a higher traffic flow than a pure 3x3 crossroad configuration. Replacing one out of nine intersections has a visible influence on the traffic flow.

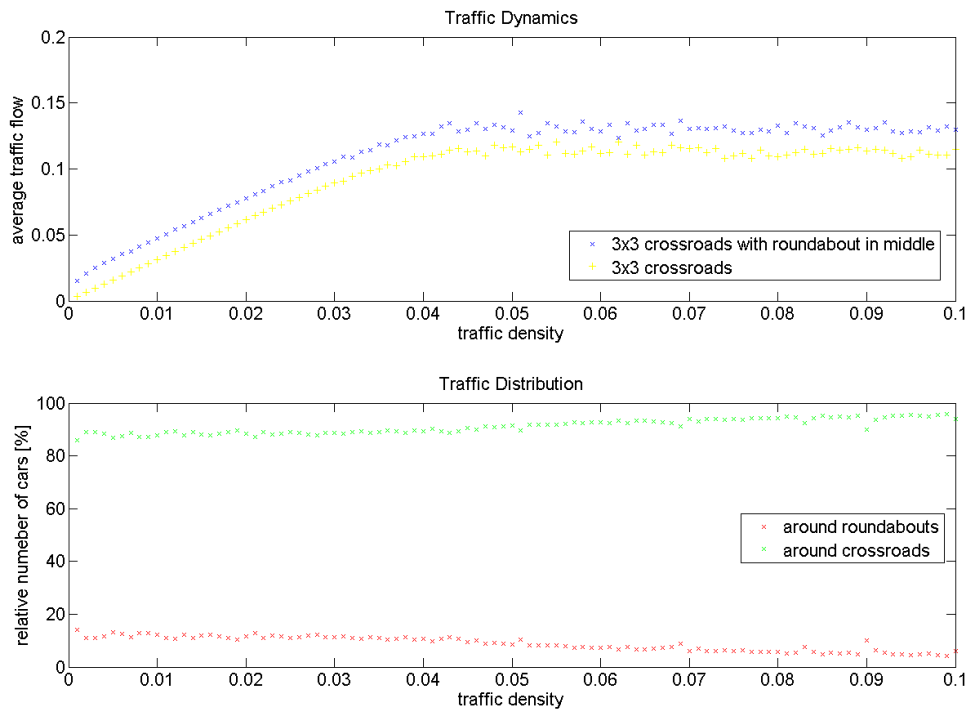


Figure 9: Traffic flow and distribution of the 3x3 crossroad configuration from figure 10

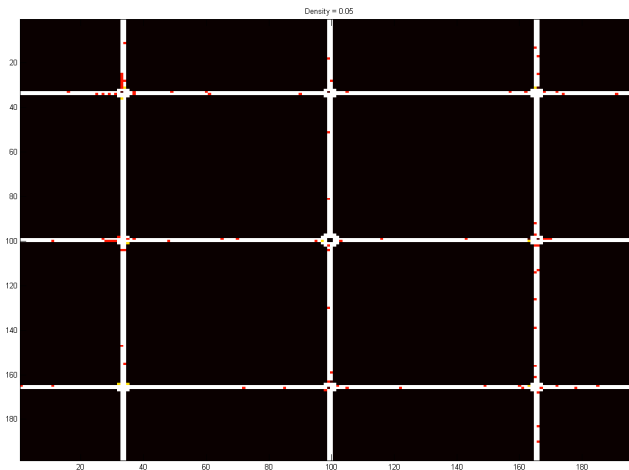


Figure 10: A 3x3 configuration with 8 crossroads around a roundabout in the center

## 5 Discussion

### 5.1 Comparison

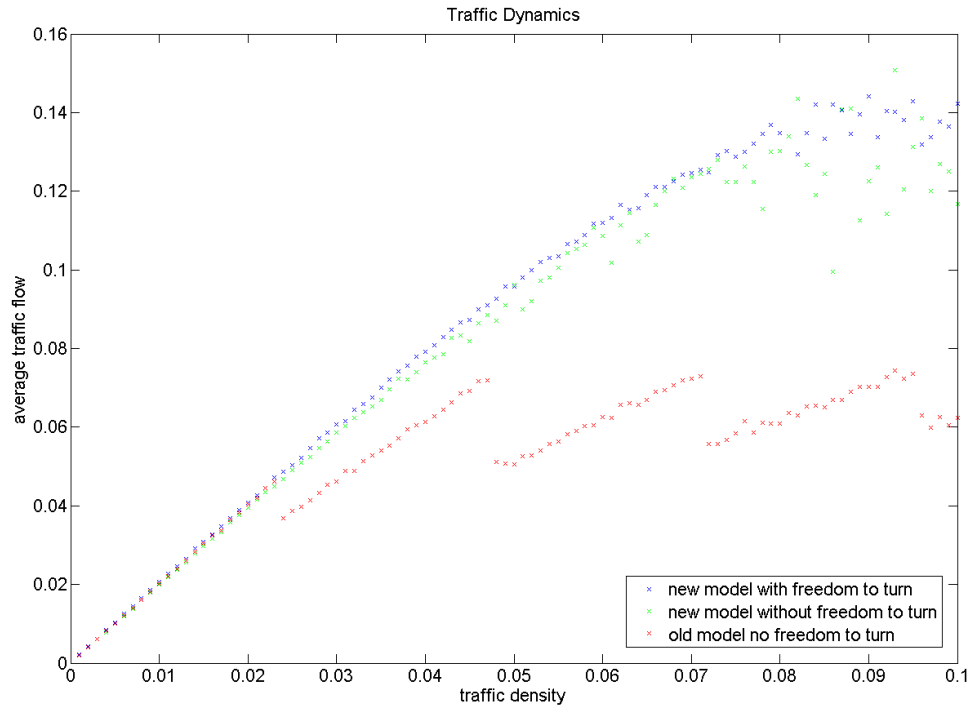


Figure 11: Comparison of models for 2x2 crossroad configurations

In figure 11 the model of Piskorec and Soller is compared to our model. For this we have shortened the distance between the intersection down to 20 cells, which is the default in their implementation. The graph also shows an edited version of our implementation. In this the ability to turn at the intersection has been removed. All three models have been evaluated in a 2x2 pure crossroad configuration for densities from 0 to 0.2.

For traffic densities over 0.02 our model produces a high traffic flow. This effect is obviously not caused by the ability to turn. The freedom of turning reduces the traffic flow slightly. Furthermore, our model does not show the periodic jumps Piskorec and Soller encountered.

## 5.2 Validation of Simplification

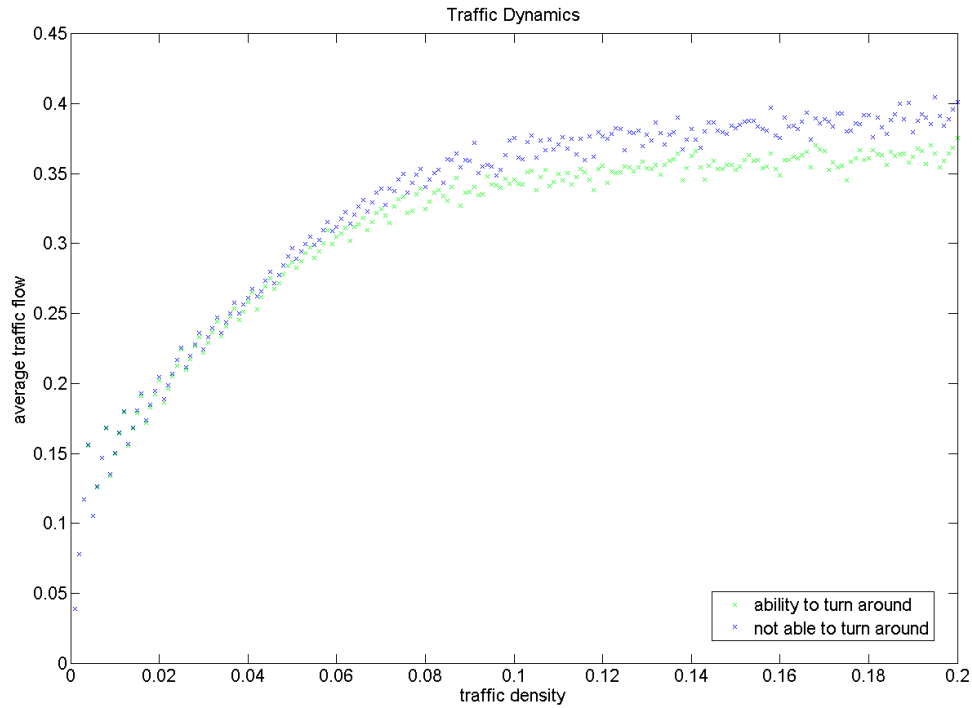


Figure 12: Comparison of single roundabouts with (green) and without (blue) the ability to turn around

To be able to compare crossroads and roundabouts, in our default model, we disabled cars to take the fourth exit in roundabouts. This means cars can't use the roundabout to turn around. Figure 12 shows the effect of this simplification. The modified model allowing cars to turn around produces a lower traffic flow for densities over 0.05. The difference however is not very big.

## 6 Summary and Outlook

In the introduction we asked which type of intersection produces the higher traffic flow. We now can answer this question clearly. Roundabouts have a much higher throughput than crossroads at every density. In a combination of the two intersection types, congestion predominantly occurs at the crossroads. Our model confirms that the increase in popularity of the roundabout over the last years is justified.

Although our model is more sophisticated than the one of our predecessor's, there are still some unrealistic aspects. For example, cars drive just as fast as they can in order not to crash. Crash report show that this is not true in practise. Also, the dimensions of our intersections are questionable. According to the cell size defined in section 2 on page 7 our intersections are 42 meters wide which is larger than normal. In addition, crossroads often have more advanced configurations of lanes than have modeled. Roundabout with two lanes are also common.

## 7 References and Code

### 7.1 References

- K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic, J. Phys. I France 2 2221–2229 (1992)
- Foils GESS - Lecture with Computer Exercises: Modeling and Simulating Social Systems with MATLAB - 2010
- Luka Piskorec and Simon Soller, Traffic Dynamics - The effectiveness of signalization and the priority to the right simulated with Cellular Automata, 2009
- <http://de.wikipedia.org/wiki/Nagel-Schreckenberg-Modell>
- [http://en.wikipedia.org/wiki/Traffic\\_flow](http://en.wikipedia.org/wiki/Traffic_flow)
- [http://en.wikipedia.org/wiki/Cellular\\_automaton](http://en.wikipedia.org/wiki/Cellular_automaton)

## 7.2 Matlab-Code

### 7.2.1 traffic.m

```
1 function traffic
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %TRAFFIC Simulation of traffic in an city map containing roundabouts and
4 %crossroads.
5 %
6 %This program requires the following subprogams:
7 %TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
8 %
9 %
10 %User will be ask to determine city map,traffic density and whether
11 %simulation is to be displayed or not.
12 %
13 %The city map is entered by supplying a matrix with elements '1' for
14 %crossroads and '0' for roundabouts.
15 %
16 %The density can be a scalar or a vector. If the density is a scalar
17 %TRAFFIC will run the simulation for all densities given. The elements must
18 %be in the range of [0,1].
19 %
20 %If Users chooses to display simulation (by entering 'y') a figure will
21 %open showing the animation:
22 %-Black cells simbolize empty space
23 %-White cells simbolize road
24 %-Red cells simbolize cars
25 %-Yellow cells simbolize cars indicating to the right
26 %-Dark red celss simbolize cars indicating to the left
27 %
28 %After all simulations have finished TRAFFIC plots the average traffic flow
29 %versus the traffic density. If city map is a mix of crossroad and
30 %roundabouts the traffic distribution (cars around roundabouts or around
31 %crossroads) versus traffic density is also plotted.
32 %
33 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
34 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
35 %Spring 2010
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38 close all;
39
40 %prompt city road configutation
41 c = input(['\nenter city map\n\ngive matrix elements: ', ...
42          'Priority to the right (=1) and Roundabout (=0) \n\n', ...
43          'i.e. [1 0 0;1 1 0;0 1 1]\n\n']);
44
45 %check c
```



```

46 [c_m,c_n] = size(c);
47 for a = 1:c_m
48     for b = 1:c_n
49         if ( c(c_m,c_n) ≠ 1 && c(c_m,c_n) ≠ 0 )
50             disp('Elements must be 0 or 1');
51             return
52         end
53     end
54 end
55 %check if city map is a mix of crossroads and roundabouts or if it made up
56 %of purely one or the other
57 if ( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 )
58     mix = false;
59 else
60     mix = true;
61 end
62
63 %prompt traffic density
64 d = input('\nenter traffic density: ');
65 %check d
66 if ( max(d) > 1 || min(d) < 0)
67     disp('density must be in range [0,1]');
68     return
69 end
70
71 %ask if simulation should be displayed
72 show = input('\ndisplay simulation graphically? yes (=y) or no (=n) ','s');
73
74 %average flow and distributions for every density supplied
75 avFlow = zeros(1,max(size(d)));
76 avRo = zeros(1,max(size(d)));
77 avCr = zeros(1,max(size(d)));
78
79 if ( show == 'y' || show == 'n' )
80     %if wanted run simulation with graphics
81     if ( show == 'y' )
82         for di=1:max(size(d))
83             [avFlow(di),avRo(di),avCr(di)] = trafficsim(d(di),c,true);
84         end
85     %if animation undesired run simulation without graphics
86     else
87         for di=1:max(size(d))
88             [avFlow(di),avRo(di),avCr(di)] = trafficsim(d(di),c,false);
89         end
90     end
91
92     figure(2);
93     %is city map is a mix of roundabout and crossroads, plot distribution
94     if ( mix )
95         %plot relativ number of cars at roundabouts and number of cars at

```

```

96     %crossroads versus traffic density
97     subplot(2,1,2);
98     plot(d,avRo*100,'rx',d,avCr*100,'gx');
99     set(gca,'FontSize',16);
100    title('Traffic Distribution');
101    xlabel('traffic density');
102    ylabel('relative number of cars [%]');
103    legend('around roundabouts','around crossroads');
104    ylim([0 100]);
105    subplot(2,1,1);
106 end
107
108 %plot traffic flow versus traffic density
109 plot(d,avFlow,'x');
110 set(gca,'FontSize',16);
111 title('Traffic Dynamics');
112 xlabel('traffic density');
113 ylabel('average traffic flow');
114 %ylim([0 0.5]);
115 else
116     disp('Input must be y or n!');
117 end
118 end

```

## 7.2.2 trafficsim.m

```
1 function [averageFlow,avCaRo,avCaCr] = trafficsim(density,config,display)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %TRAFFICSIM Simulation of traffic in an city map containing roundabouts and
4 %crossroads.
5 %
6 %Output:
7 %AVERAGEFLOW, Average traffic flow for given city map and density
8 %AVCARO, Average amount of cars around roundabouts
9 %AVCACR, Average amount of cars around crossroads
10 %
11 %INPUT:
12 %DENSITY, Traffic density
13 %CONFIG, City map
14 %DISPLAY, Turn graphics on 'true' or off 'false'
15 %
16 %This program requires the following subprogams:
17 %ROUNABOUT,CROSSROAD,CONNECTION,PDESTINATION
18 %
19 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
20 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
21 %Spring 2010
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23
24 %dawde probability
25 dawdleProb = 0.2;
26 %street length (>5)
27 l = 30;
28 %number of iterations
29 nIt=1000;
30
31 %dimensions of config, how many intersections in x and y direction are
32 %there?
33 [config_m,config_n] = size(config);
34
35 %in streets cell values indicate the following:
36 %0.4 means there is a car in this position (red in figure)
37 %1 means there is no car in this position (white in figure)
38
39 %initialize matrices for streets heading toward intersections
40 t = ones(4*config_m,l*config_n);
41 tspeed = zeros(4*config_m,l*config_n);
42 %number of elements in t
43 tsize = sum(sum(t));
44
45 %initialize matrices for street leading away from intersections
46 f = ones(4*config_m,l*config_n);
```

```

47 fspeed = zeros(4*config_m,1*config_n);
48
49 %initialize matrices for roundabouts
50 r = ones(config_m,12*config_n);
51 rspeed = zeros(config_m,12*config_n);
52 rex = zeros(config_m,12*config_n);
53
54 %initialize matrices for crossings with priority to the right
55 p = ones(6*config_m,6*config_n);
56 pspeed = zeros(6 *config_m,6*config_n);
57 came = zeros(6*config_m,6*config_n);
58 %deadlock prevention
59 deadlock = zeros(config_m,config_n);
60
61 %initialaize map
62 map = zeros(config_m*(2*1+6),config_n*(2*1+6));
63 %initialize gap
64 gap = 0;
65
66 %initialize flow calculation variables
67 avSpeedIt = zeros(nIt+1,1);
68 %counter for cars around crossroads
69 numCaCrIt = zeros(nIt+1,1);
70 %counter for cars around crossroads
71 numCaRoIt = zeros(nIt+1,1);
72
73 %distribute cars randomly on streets for starting point
74 overall_length = sum(sum(t)) + sum(sum(f));
75 numCars = ceil(density * overall_length);
76 q = 1;
77
78 while ( q ≤ numCars )
79     w = randi(overall_length,1);
80     if ( w ≤ tsize )
81         if ( t(w) == 1 )
82             t(w) = 0.4;
83             tspeed(w) = randi(5,1);
84             q = q + 1;
85         end
86     end
87     if ( w > tsize )
88         if ( f(w-tsize) == 1 )
89             f(w-tsize) = 0.4;
90             fspeed(w-tsize) = randi(5,1);
91             q = q +1 ;
92         end
93     end
94 end
95
96

```

```

97 %iterate over time
98 for time = 1:nIt+1
99
100 %clear values for next step
101 t_next = ones(4*config_m,1*config_n);
102 tspeed_next = zeros(4*config_m,1*config_n);
103 f_next = ones(4*config_m,1*config_n);
104 fspeed_next = zeros(4*config_m,1*config_n);
105 r_next = ones(config_m,12*config_n);
106 rspeed_next = zeros(config_m,12*config_n);
107 rex_next = zeros(config_m,12*config_n);
108 p_next = ones(6*config_m,6*config_n);
109 pspeed_next = ones(6*config_m,6*config_n);
110 came_next = zeros(6*config_m,6*config_n);
111 deadlock_next = zeros(config_m,config_n);
112
113 %iterate over all intersection
114 for a = 1:config_m
115     for b = 1:config_n
116
117         %define Index starting points for each intersection
118         tI_m = (a - 1) * 4;
119         tI_n = (b - 1) * 1;
120         mapI_m = (a - 1) * (2 * 1 + 6);
121         mapI_n = (b - 1) * (2 * 1 + 6);
122
123         %positions outside intersections
124         %for every intersection iterate along streets
125         for c = tI_m + 1:tI_m + 4
126             for d = tI_n + 1:tI_n+1
127
128                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
129                 %streets to intersections
130
131                 %deal with position directly in front of intersection
132                 %separately later
133                 if ( mod(d,1)  $\neq$  0 )
134                     %if there is a car in this position, apply
135                     %NS-Model
136                     if ( t(c,d) == 0.4 )
137                         %Nagel-Schreckenberg-Model
138                         %NS 1. step: increase velocity if < 5
139                         v = tspeed(c,d);
140                         if ( v < 5)
141                             v = v + 1;
142                         end
143
144                         %NS 2. step: adapt speed to gap
145                         %how big is gap (to car ahead or intersection)?
146                         e = 1;

```

```

147         while ( e ≤ 5 && d + e ≤ b * l && ...
148             t(c,d+e) == 1 )
149             e = e + 1;
150         end
151         gap = e - 1;
152         %reduce speed if gap is too small
153         if ( v > gap )
154             v = gap;
155         end
156
157         %NS 3. step: dawdle
158         if ( rand < dawdleProb && v ≠ 0 )
159             v = v - 1;
160         end
161
162         %NS 4. step: drive, move cars tspeed(c,d) cells
163         %forward
164         %new position
165         t_next(c,d+v) = 0.4;
166         tspeed_next(c,d+v) = v;
167     end
168 end
169
170 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
171 %street from intersections
172
173 if ( f(c,d) == 0.4 )
174     %Nagel-Schreckenberg-Model
175     %NS 1. step: increase velocity if < 5
176     v = fspeed(c,d);
177     if ( v < 5 )
178         v = v + 1;
179     end
180
181     %NS 2.step: adpat speed to gap
182     %how big is gap (to car ahead)?
183     e = 1;
184     while ( e ≤ 5 )
185         %if gap is bigger than distance to edge,connect
186         %steets
187         if ( d + e > b * l )
188             %testing position in new street
189             hh = d + e - b * l;
190             %connect to next street
191             [ec,ed]=connection(a,b,c,hh, ...
192                 config_m,config_n,l);
193             while ( t(ec,ed) == 1 && e ≤ 5 )
194                 e = e + 1;
195                 %testing position in new street
196                 hh = d + e - b * l;

```

```

197         %connect to next street
198         [ec,ed]=connection(a,b,c,hh, ...
199             config_m,config_n,l);
200     end
201     gap = e - 1;
202     e = 6;
203     else
204         if ( f(c,d+e) == 1 )
205             e = e + 1;
206             if ( e == 6 )
207                 gap = 5;
208             end
209         else
210             gap = e - 1;
211             e = 6;
212         end
213     end
214 end
215 %reduce speed if gap is too small
216 if ( v > gap )
217     v = gap;
218 end
219
220 %NS 3. step: dawdle
221 if ( rand ≤ dawdleProb && v ≠ 0 )
222     v = v - 1;
223 end
224
225 %NS 4. step: drive, move cars fspeed(c,d) cells
226 %forward
227 %if new position is off this street, connect
228 %streets
229 if ( d + v > b * l )
230     %position in new street
231     hhh = d + v - b * l;
232     %connect next street
233     [ec,ed] = connection(a,b,c,hhh, ...
234         config_m,config_n,l);
235     t_next(ec,ed) = 0.4;
236     tspeed_next(ec,ed) = v;
237 else
238     f_next(c,d+v) = 0.4;
239     fspeed_next(c,d+v) = v;
240 end
241     end
242 end
243 end
244
245 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
246 %roundabouts

```

```

247
248 %check if intersection is a roundabout
249 if ( config(a,b) == 0 )
250     %define index strating point for this roundabout
251     rI_n = (b - 1) * 12;
252
253     %do roundabout calculations for this roundabout and time
254     %step
255     %call ROUNDABOUT
256     [t_next(tI_m+1:tI_m+4,tI_n+1), ...
257      tspeed_next(tI_m+1:tI_m+4,tI_n+1), ...
258      f_next(tI_m+1:tI_m+4,tI_n+1), ...
259      fspeed_next(tI_m+1:tI_m+4,tI_n+1), ...
260      r_next(a,rI_n+1:rI_n+12), ...
261      rspeed_next(a,rI_n+1:rI_n+12), ...
262      rex_next(a,rI_n+1:rI_n+12)] = ...
263     roundabout(t(tI_m+1:tI_m+4,tI_n+1), ...
264     f(tI_m+1:tI_m+4,tI_n+1), ...
265     r(a,rI_n+1:rI_n+12), ...
266     rex(a,rI_n+1:rI_n+12), ...
267     t_next(tI_m+1:tI_m+4,tI_n+1), ...
268     tspeed_next(tI_m+1:tI_m+4,tI_n+1), ...
269     f_next(tI_m+1:tI_m+4,tI_n+1), ...
270     fspeed_next(tI_m+1:tI_m+4,tI_n+1));
271
272     %write roundabout into map
273     map(mapI_m+1+1:mapI_m+1+6,mapI_n+1+1:mapI_n+1+6) = ...
274     [ 0 1 r(a,rI_n+4) r(a,rI_n+3) 1 0;
275     1 r(a,rI_n+5) 1 1 r(a,rI_n+2) 1;
276     r(a,rI_n+6) 1 0 0 1 r(a,rI_n+1);
277     r(a,rI_n+7) 1 0 0 1 r(a,rI_n+12);
278     1 r(a,rI_n+8) 1 1 r(a,rI_n+11) 1;
279     0 1 r(a,rI_n+9) r(a,rI_n+10) 1 0];
280
281     %add cars around this crossroad in this time step to
282     %counter for cars around crossroads
283     for v = tI_m+1:tI_m+4
284         for w = tI_n+1:tI_n+1
285             if ( t(v,w) ≠ 1 )
286                 numCaRoIt(time) = numCaRoIt(time) + 1;
287             end
288             if ( f(v,w) ≠ 1 )
289                 numCaRoIt(time) = numCaRoIt(time) + 1;
290             end
291         end
292     end
293     for y = rI_n+1:rI_n+12
294         if ( r(a,y) ≠ 1 )
295             numCaRoIt(time) = numCaRoIt(time) + 1;
296         end

```



```

297         end
298
299     end
300
301     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
302     %crossroads
303
304     %check if intersection is a crossing with priority to the right
305     if ( config(a,b) == 1 )
306         %define index strating points for this crossraod
307         pI.m = (a - 1) * 6;
308         pI.n = (b - 1) * 6;
309
310         %do crossroad calculations for this crossroad and time step
311         %call CROSSROAD
312         [t_next(tI.m+1:tI.m+4,tI.n+1), ...
313          tspeed_next(tI.m+1:tI.m+4,tI.n+1), ...
314          f_next(tI.m+1:tI.m+4,tI.n+1), ...
315          fspeed_next(tI.m+1:tI.m+4,tI.n+1), ...
316          p_next(pI.m+1:pI.m+6,pI.n+1:pI.n+6), ...
317          pspeed_next(pI.m+1:pI.m+6,pI.n+1:pI.n+6), ...
318          came_next(pI.m+1:pI.m+6,pI.n+1:pI.n+6), ...
319          deadlock_next(a,b), ...
320          map(mapI.m+1+1:mapI.m+1+6,mapI.n+1+1:mapI.n+1+6)] ...
321         = crossroad(t(tI.m+1:tI.m+4,tI.n+1), ...
322          f(tI.m+1:tI.m+4,tI.n+1), ...
323          p(pI.m+1:pI.m+6,pI.n+1:pI.n+6), ...
324          came(pI.m+1:pI.m+6,pI.n+1:pI.n+6), ...
325          deadlock(a,b), ...
326          t_next(tI.m+1:tI.m+4,tI.n+1), ...
327          tspeed_next(tI.m+1:tI.m+4,tI.n+1), ...
328          f_next(tI.m+1:tI.m+4,tI.n+1), ...
329          fspeed_next(tI.m+1:tI.m+4,tI.n+1));
330
331         %add cars around this roundabout in this time step to
332         %counter for cars around roundabouts
333         for v = tI.m+1:tI.m+4
334             for w = tI.n+1:tI.n+1
335                 if ( t(v,w) ≠ 1 )
336                     numCaCrIt(time) = numCaCrIt(time) + 1;
337                 end
338                 if ( f(v,w) ≠ 1 )
339                     numCaCrIt(time) = numCaCrIt(time) + 1;
340                 end
341             end
342         end
343         for x = pI.m+1:pI.m+6
344             for y = pI.n+1:pI.n+6
345                 if ( came(x,y) ≠ 0 )
346                     numCaCrIt(time) = numCaCrIt(time) + 1;

```

```

347         end
348     end
349 end
350
351 end
352
353 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
354 %write streets into map
355 for i = 1:l
356     map(mapI_m+i,mapI_n+l+3) = t(tI_m+l,tI_n+i);
357     map(mapI_m+l+4,mapI_n+i) = t(tI_m+2,tI_n+i);
358     map(mapI_m+2*l+7-i,mapI_n+l+4) = t(tI_m+3,tI_n+i);
359     map(mapI_m+l+3,mapI_n+2*l+7-i) = t(tI_m+4,tI_n+i);
360     map(mapI_m+l+1-i,mapI_n+l+4) = f(tI_m+l,tI_n+i);
361     map(mapI_m+l+3,mapI_n+l+1-i) = f(tI_m+2,tI_n+i);
362     map(mapI_m+l+6+i,mapI_n+l+3) = f(tI_m+3,tI_n+i);
363     map(mapI_m+l+4,mapI_n+l+6+i) = f(tI_m+4,tI_n+i);
364 end
365
366 %illustrate traffic situation (now not of next time step)
367 if ( display)
368     figure(1);
369     imagesc(map);
370     colormap(hot);
371     titlestring = sprintf('Density = %g',density);
372     title(titlestring);
373     drawnow;
374 end
375
376
377     end
378 end
379
380 %calculate average velocity per time step
381 avSpeedIt(time) = ( sum(sum(tspeed)) + sum(sum(fspeed)) + ...
382     sum(sum(rspeed)) + sum(sum(pspeed)) ) / numCars;
383
384 %pause(1);
385
386 %move on time step on
387 t = t_next;
388 tspeed = tspeed_next;
389 f = f_next;
390 fspeed = fspeed_next;
391 r = r_next;
392 rspeed = rspeed_next;
393 rex = rex_next;
394 p = p_next;
395 pspeed = pspeed_next;
396 came = came_next;

```

```
397     deadlock = deadlock_next;
398 end
399
400 %overall average velocity
401 averageSpeed = sum(avSpeedIt) / max(size(avSpeedIt));
402 %overall average flow
403 averageFlow = density * averageSpeed;
404
405 %average relative amount of cars around roundabouts
406 avCaRo = sum(numCaRoIt) / ( max(size(numCaRoIt)) * numCars );
407 %average relative amount of cars around crossroads
408 avCaCr = sum(numCaCrIt) / ( max(size(numCaCrIt)) * numCars );
409
410 end
```

### 7.2.3 roundabout.m

```
1 function [tr_next, ...
2     trspeed_next, ...
3     fr_next, ...
4     frspeed_next, ...
5     rlocal_next, ...
6     rspeedlocal_next, ...
7     rexlocal_next] ...
8     = roundabout(tr, ...
9     fr, ...
10    rlocal, ...
11    rexlocal, ...
12    tr_next, ...
13    trspeed_next, ...
14    fr_next,...
15    frspeed_next)
16 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17 %ROUNDABOUT Calculation of update for a certain roundabout, density and
18 %time step
19 %
20 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
21 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
22 %Spring 2010
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24
25 %in roundabout cell values indicate if car is about to leave roundabout:
26 %0.4 means car is not taking next exit (red in figure)
27 %0.7 means car is taking next exit (yellow in figure)
28 %1 means no car in this position (white in figure)
29
30 %clear local next variables
31 rlocal_next = ones(1,12);
32 rspeedlocal_next = zeros(1,12);
33 rexlocal_next = zeros(1,12);
34
35 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36 %car in front of roundabout
37
38 for k = 1:4
39     if ( tr(k,1) == 0.4 )
40         %entering roundabout with velocity 1 when possible
41         %roundabout position index
42         iR = mod(3*k+1,12);
43         if ( rexlocal(k*3) <= 1 && rlocal(iR) == 1 )
44             %enter roundabout
45             %decide which exit car is going to take
46             u = randi(12,1);
```

```

47         %probabilty 6/12 take it takes 2. exit
48         if ( u ≤ 6 )
49             rexlocal_next(iR) = 2;
50             rlocal_next(iR) = 0.4;
51             rspeedlocal_next(iR) = 1;
52         end
53         %probabilty 3/12 take it takes 1. exit
54         if ( u ≥ 7 && u ≤ 9 )
55             rexlocal_next(iR) = 1;
56             %indicate
57             rlocal_next(iR) = 0.7;
58             rspeedlocal_next(iR) = 1;
59         end
60         %probabilty 3/12 take it takes 3. exit
61         if ( u ≥ 10 && u ≤ 12 )
62             rexlocal_next(iR) = 3;
63             rlocal_next(iR) = 0.4;
64             rspeedlocal_next(iR) = 1;
65         end
66         %probabilty 1/12 take it takes 4. exit (turns around)
67         %if ( u == 12 )
68         %     rexlocal_next(iR) = 4;
69         %     rlocal_next(iR) = 0.4;
70         %     rspeedlocal_next(iR) = 1;
71         %end
72
73         %car waiting in front of roundabout
74         else
75             tr_next(k,1) = tr(k,1);
76             trspeed_next(k,1) = 0;
77         end
78     end
79 end
80
81 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 %car in roundabout
83
84 for j = 1:12
85     if ( rlocal(j) ≠ 1 )
86
87         %cars in roundabout not at an exit
88         if (mod(j,3) ≠ 0 )
89             %if space free, move one forward
90             if ( rlocal(j+1) == 1 )
91                 %take new position
92                 rlocal_next(j+1) = rlocal(j);
93                 rspeedlocal_next(j+1) = 1;
94                 rexlocal_next(j+1) = rexlocal(j);
95             %if no space free, stay
96             else

```

```

97         rlocal_next(j) = rlocal(j);
98         rspeedlocal_next(j) = 0;
99         rexlocal_next(j) = rexlocal(j);
100     end
101
102     %car at an exit
103     else
104
105         %if car is at its exit
106         if ( rexlocal(j) == 1 )
107             %if space free, leave roundabout
108             if ( fr(j/3,1) == 1 )
109                 fr_next(j/3,1) = 0.4;
110                 frspeed_next(j/3,1) = 1;
111             %if no space free, stay
112             else
113                 rlocal_next(j) = rlocal(j);
114                 rspeedlocal_next(j) = 0;
115                 rexlocal_next(j) = rexlocal(j);
116             end
117
118             %car at an exit but not the one its taking
119             else
120                 %connect r(12) with r(1)
121                 if ( j == 12 )
122                     %if space free, move one forward and decrease exit
123                     %counter
124                     if ( rlocal(1) == 1 )
125                         %decrease exit by one
126                         rexlocal_next(1) = rexlocal(12) - 1;
127                         rspeedlocal_next(1) = 1;
128                         if ( rexlocal_next(1) == 1 )
129                             %indicate
130                             rlocal_next(1) = 0.7;
131                         else
132                             rlocal_next(1) = 0.4;
133                         end
134                     %if no space free, stay
135                     else
136                         rlocal_next(12) = rlocal(12);
137                         rspeedlocal_next(12) = 0;
138                         rexlocal_next(12) = rexlocal(12);
139                     end
140                 else
141                     %if space free, move one forward and decrease exit
142                     %counter
143                     if ( rlocal(j+1) == 1 )
144                         %decrease exit by one
145                         rexlocal_next(j+1) = rexlocal(j) - 1;
146                         rspeedlocal_next(j+1) = 1;

```

```

147         if ( rexlocal_next(j+1) == 1 )
148             %indicate
149             rlocal_next(j+1) = 0.7;
150         else
151             rlocal_next(j+1) = 0.4;
152         end
153     %if no space free, stay
154     else
155         rlocal_next(j) = rlocal(j);
156         rspeedlocal_next(j) = 0;
157         rexlocal_next(j) = rexlocal(j);
158     end
159 end
160 end
161 end
162 end
163 end
164
165 end

```

## 7.2.4 crossroad.m

```
1 function [tp_next, ...
2     tpspeed_next, ...
3     fp_next, ...
4     fpspeed_next, ...
5     plocal_next ...
6     pspeedlocal_next, ...
7     camelocal_next, ...
8     deadlocklocal_next, ...
9     plocal] ...
10 = crossroad(tp, ...
11     fp, ...
12     plocal, ...
13     camelocal, ...
14     deadlocklocal, ...
15     tp_next, ...
16     tpspeed_next, ...
17     fp_next, ...
18     fpspeed_next)
19 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20 %CROSSROAD Calculation of update for a certain crossroad, density and time
21 %step
22 %
23 %This program requires the following subprogams:
24 %PDESTINATION
25 %
26 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
27 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
28 %Spring 2010
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31 %in crossroad cell values indicate where cars is going:
32 %0.1 means car is turning left (dark red in figure)
33 %0.4 means car is going straight ahead (red in figure)
34 %0.7 means car is turning right (yellow in figure)
35 %1 means no car in this position (white in figure)
36
37 %clear local next variables
38 plocal_next = ones(6,6);
39 pspeedlocal_next = zeros(6,6);
40 camelocal_next = zeros(6,6);
41 deadlocklocal_next = 0;
42
43 %'paint' unused corners of plocal black
44 plocal(1,1) = 0;
45 plocal(1,6) = 0;
46 plocal(6,1) = 0;
```



```

47 plocal(6,6) = 0;
48 plocal(1,2) = 0;
49 plocal(1,5) = 0;
50 plocal(2,1) = 0;
51 plocal(2,6) = 0;
52 plocal(5,1) = 0;
53 plocal(5,6) = 0;
54 plocal(6,2) = 0;
55 plocal(6,5) = 0;
56
57 %key to unlock deadlock for this iteration and this
58 %intersection
59 unlock = randi(4,1);
60
61 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
62 %cars in front of crossroad
63
64 %car waiting from above
65 if ( tp(1,1) == 0.4 )
66     %if space is free and there is no car coming from the
67     %left going straight ahead already in crossing, enter
68     if ( plocal(1,3) == 1 && camelocal(2,3) ≠ 4 && ...
69         camelocal(2,4) ≠ 1 && ...
70         ¬( camelocal(2,5) == 4 && plocal(2,5) == 0.4 ) )
71         %decide where car is heading
72         plocal_next(1,3) = pdestination;
73         pspeedlocal_next(1,3) = 1;
74         %mark which entrance car came from
75         camelocal_next(1,3) = 1;
76     %if not wait
77     else
78         tp_next(1,1) = tp(1,1);
79         tpspeed_next(1,1) = 0;
80     end
81 end
82
83 %car waiting from left
84 if ( tp(2,1) == 0.4 )
85     %if space is free and there is no car coming from the
86     %left going straight ahead already in crossing, enter
87     if ( plocal(4,1) == 1 && camelocal(4,2) ≠ 1 && ...
88         camelocal(3,2) ≠ 1 && ...
89         ¬( camelocal(2,2) == 1 && plocal(2,2) == 0.4 ) )
90         %decide where car is heading
91         plocal_next(4,1) = pdestination;
92         pspeedlocal_next(4,1) = 1;
93         %mark which entrance car came from
94         camelocal_next(4,1) = 2;
95     %if not wait
96     else

```

```

97         tp_next(2,1) = tp(2,1);
98         tpspeed_next(2,1) = 0;
99     end
100 end
101
102 %car waiting from below
103 if ( tp(3,1) == 0.4 )
104     %if space is free and there is no car coming from the
105     %left going straight ahead already in crossing, enter
106     if ( plocal(6,4) == 1 && camelocal(5,4) ≠ 2 && ...
107         camelocal(5,3) ≠ 2 && ...
108         ¬( camelocal(5,2) == 2 && plocal(5,2) == 0.4 ) )
109         %decide where car is heading
110         plocal_next(6,4) = pdestination;
111         pspeedlocal_next(6,4) = 1;
112         %mark which entrance car came from
113         camelocal_next(6,4) = 3;
114         %if not wait
115     else
116         tp_next(3,1) = tp(3,1);
117         tpspeed_next(3,1) = 0;
118     end
119 end
120
121 %car waiting from right
122 if ( tp(4,1) == 0.4 )
123     %if space is free and there is no car coming from the
124     %left going straight ahead already in crossing, enter
125     if ( plocal(3,6) == 1 && camelocal(3,5) ≠ 3 && ...
126         camelocal(4,5) ≠ 3 && ...
127         ¬( camelocal(5,5) == 3 && plocal(5,5) == 0.4 ) )
128         %decide where car is heading
129         plocal_next(3,6) = pdestination;
130         pspeedlocal_next(3,6) = 1;
131         %mark which entrance car came from
132         camelocal_next(3,6) = 4;
133         %if not wait
134     else
135         tp_next(4,1) = tp(4,1);
136         tpspeed_next(4,1) = 0;
137     end
138 end
139
140 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
141 %cars going turning right step 1
142
143 %car coming form above, turning right
144 %1. step
145 if ( plocal(1,3) == 0.7 )
146     %if space free, car has right of way and can drive

```

```

147     if ( plocal(2,2) == 1 && plocal(2,3) ≠ 0.4 )
148         plocal_next(2,2) = plocal(1,3);
149         pspeedlocal_next(2,2) = 1;
150         camelocal_next(2,2) = camelocal(1,3);
151     % if space not free, stay
152     else
153         plocal_next(1,3) = plocal(1,3);
154         pspeedlocal_next(1,3) = 0;
155         camelocal_next(1,3) = camelocal(1,3);
156     end
157 end
158
159 %car coming form left, turning right
160 %1. step
161 if ( plocal(4,1) == 0.7 )
162     %if space free, car has right of way and can drive
163     if ( plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 )
164         plocal_next(5,2) = plocal(4,1);
165         pspeedlocal_next(5,2) = 1;
166         camelocal_next(5,2) = camelocal(4,1);
167     % if space not free, stay
168     else
169         plocal_next(4,1) = plocal(4,1);
170         pspeedlocal_next(4,1) = 0;
171         camelocal_next(4,1) = camelocal(4,1);
172     end
173 end
174
175 %car coming form below, turning right
176 %1. step
177 if ( plocal(6,4) == 0.7 )
178     %if space free, car has right of way and can drive
179     if ( plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 )
180         plocal_next(5,5) = plocal(6,4);
181         pspeedlocal_next(5,5) = 1;
182         camelocal_next(5,5) = camelocal(6,4);
183     % if space not free, stay
184     else
185         plocal_next(6,4) = plocal(6,4);
186         pspeedlocal_next(6,4) = 0;
187         camelocal_next(6,4) = camelocal(6,4);
188     end
189 end
190
191 %car coming form right, turning right
192 %1. step
193 if ( plocal(3,6) == 0.7 )
194     %if space free, car has right of way and can drive
195     if ( plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 )
196         plocal_next(2,5) = plocal(3,6);

```

```

197         pspeedlocal.next(2,5) = 1;
198         camelocal.next(2,5) = camelocal(3,6);
199     % if space not free, stay
200     else
201         plocal.next(3,6) = plocal(3,6);
202         pspeedlocal.next(3,6) = 0;
203         camelocal.next(3,6) = camelocal(3,6);
204     end
205 end
206
207 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
208 %cars going straight ahead step 1
209
210 %car coming form above, going stright ahead
211 %1. step
212 if ( plocal(1,3) == 0.4 )
213     %if space is free and there are no are coming from the
214     %right or is there has been a deadlock and driver have
215     %agreed by hand signal to let this car go, dive
216     %!warning: only works if this step is done after update
217     %of cars in front of crossraad!
218     if ( plocal(2,2) == 1 && plocal(2,3) ≠ 0.4 && ...
219         ( ( tp.next(2,1) == 1 && plocal.next(4,1) == 1 && ...
220           plocal(4,1) == 1 ) || ( deadlocklocal == 4 && unlock == 1 ) ) )
221         plocal.next(2,2) = plocal(1,3);
222         pspeedlocal.next(2,2) = 1;
223         camelocal.next(2,2) = camelocal(1,3);
224         %no deadlock, clear deadlock counter
225         deadlocklocal.next = 0;
226     % if not, stay
227     else
228         plocal.next(1,3) = plocal(1,3);
229         pspeedlocal.next(1,3) = 0;
230         camelocal.next(1,3) = camelocal(1,3);
231         %increase deadlock counter, if it reaches 4 a
232         %deadlock occurs and will have to be solve in next
233         %time step by a hand signals between drivers
234         deadlocklocal.next = deadlocklocal.next + 1;
235     end
236 end
237
238 %car coming form left, going stright ahead
239 %1. step
240 if ( plocal(4,1) == 0.4 )
241     %if space is free and there are no are coming from the
242     %right or is there has been a deadlock and driver have
243     %agreed by hand signal to let this car go, dive
244     %!warning: only works if this step is done after update
245     %of cars in front of crossraad!
246     if ( plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 && ...

```

```

247         ( ( tp.next(3,1) == 1 && plocal.next(6,4) == 1 && ...
248         plocal(6,4) == 1 ) || ( deadlocklocal == 4 && unlock == 2 ) ) )
249     plocal.next(5,2) = plocal(4,1);
250     pspeedlocal.next(5,2) = 1;
251     camelocal.next(5,2) = camelocal(4,1);
252     %no deadlock, clear deadlock counter
253     deadlocklocal.next = 0;
254     % if not, stay
255     else
256         plocal.next(4,1) = plocal(4,1);
257         pspeedlocal.next(4,1) = 0;
258         camelocal.next(4,1) = camelocal(4,1);
259         %increase deadlock counter, if it reaches 4 a
260         %deadlock occurs and will have to be solve in next
261         %time step by a hand signals between drivers
262         deadlocklocal.next = deadlocklocal.next + 1;
263     end
264 end
265
266 %car coming form below, going stright ahead
267 %1. step
268 if ( plocal(6,4) == 0.4 )
269     %if space is free and there are no are coming from the
270     %right or is there has been a deadlock and driver have
271     %agreed by hand signal to let this car go, dive
272     %!warning: only works if this step is done after update
273     %of cars in front of crossraod!
274     if ( plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 && ...
275         ( ( tp.next(4,1) == 1 && plocal.next(3,6) == 1 && ...
276         plocal(3,6) == 1 ) || ( deadlocklocal == 4 && unlock == 3 ) ) )
277         plocal.next(5,5) = plocal(6,4);
278         pspeedlocal.next(5,5) = 1;
279         camelocal.next(5,5) = camelocal(6,4);
280         %no deadlock, clear deadlock counter
281         deadlocklocal.next = 0;
282         % if not, stay
283         else
284             plocal.next(6,4) = plocal(6,4);
285             pspeedlocal.next(6,4) = 0;
286             camelocal.next(6,4) = camelocal(6,4);
287             %increase deadlock counter, if it reaches 4 a
288             %deadlock occurs and will have to be solve in next
289             %time step by a hand signals between drivers
290             deadlocklocal.next = deadlocklocal.next + 1;
291         end
292     end
293
294 %car coming form right, going stright ahead
295 %1. step
296 if ( plocal(3,6) == 0.4 )

```

```

297 %if space is free and there are no are coming from the
298 %right or is there has been a deadlock and driver have
299 %agreed by hand signal to let this car go, dive
300 %!warning: only works if this step is done after update
301 %of cars in front of crossraod!
302 if ( plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 && ...
303     ( ( tp_next(1,1) == 1 && plocal_next(1,3) == 1 && ...
304         plocal(1,3) == 1 ) || ( deadlocklocal == 4 && unlock == 4 ) ) )
305     plocal_next(2,5) = plocal(3,6);
306     pspeedlocal_next(2,5) = 1;
307     camelocal_next(2,5) = camelocal(3,6);
308     %no deadlock, clear deadlock counter
309     deadlocklocal_next = 0;
310 % if not, stay
311 else
312     plocal_next(3,6) = plocal(3,6);
313     pspeedlocal_next(3,6) = 0;
314     camelocal_next(3,6) = camelocal(3,6);
315     %increase deadlock counter, if it reaches 4 a
316     %deadlock occurs and will have to be solve in next
317     %time step by a hand signals between drivers
318     deadlocklocal_next = deadlocklocal_next + 1;
319 end
320 end
321
322 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
323 %cars turning right step 2
324 %cars going straight ahead step 5
325
326 %2. step for car coming from above, turning right
327 %5. step for car coming from right, going straight ahead
328 if ( plocal(2,2) == 0.7 || ( plocal(2,2) == 0.4 && camelocal(2,2) == 4 ) )
329     %if space free, car has right of way and can drive
330     if ( plocal(3,1) == 1 )
331         plocal_next(3,1) = plocal(2,2);
332         pspeedlocal_next(3,1) = 1;
333         camelocal_next(3,1) = camelocal(2,2);
334         % if space not free, stay
335     else
336         plocal_next(2,2) = plocal(2,2);
337         pspeedlocal_next(2,2) = 0;
338         camelocal_next(2,2) = camelocal(2,2);
339     end
340 end
341
342 %2. step for car coming from left, turning right
343 %5. step for car coming from above, going straight ahead
344 if ( plocal(5,2) == 0.7 || ( plocal(5,2) == 0.4 && camelocal(5,2) == 1 ) )
345     %if space free, car has right of way and can drive
346     if ( plocal(6,3) == 1 )

```

```

347         plocal_next(6,3) = plocal(5,2);
348         pspeedlocal_next(6,3) = 1;
349         camelocal_next(6,3) = camelocal(5,2);
350         % if space not free, stay
351     else
352         plocal_next(5,2) = plocal(5,2);
353         pspeedlocal_next(5,2) = 0;
354         camelocal_next(5,2) = camelocal(5,2);
355     end
356 end
357
358 %2. step for car coming from below, turning right
359 %5. step for car coming from left, going straight ahead
360 if ( plocal(5,5) == 0.7 || ( plocal(5,5) == 0.4 && camelocal(5,5) == 2 ) )
361     %if space free, car has right of way and can drive
362     if ( plocal(4,6) == 1 )
363         plocal_next(4,6) = plocal(5,5);
364         pspeedlocal_next(4,6) = 1;
365         camelocal_next(4,6) = camelocal(5,5);
366     % if space not free, stay
367     else
368         plocal_next(5,5) = plocal(5,5);
369         pspeedlocal_next(5,5) = 0;
370         camelocal_next(5,5) = camelocal(5,5);
371     end
372 end
373
374 %2. step for car coming from right, turning right
375 %5. step for car coming from below, going straight ahead
376 if ( plocal(2,5) == 0.7 || ( plocal(2,5) == 0.4 && camelocal(2,5) == 3 ) )
377     %if space free, car has right of way and can drive
378     if ( plocal(1,4) == 1 )
379         plocal_next(1,4) = plocal(2,5);
380         pspeedlocal_next(1,4) = 1;
381         camelocal_next(1,4) = camelocal(2,5);
382     % if space not free, stay
383     else
384         plocal_next(2,5) = plocal(2,5);
385         pspeedlocal_next(2,5) = 0;
386         camelocal_next(2,5) = camelocal(2,5);
387     end
388 end
389
390 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
391 %cars going staight ahead step 2 to 4
392
393 %car coming form above, going staight ahead
394 %2. step
395 if ( plocal(2,2) == 0.4 && camelocal(2,2) == 1 )
396     %if space is free, drive

```

```

397     if ( plocal(3,2) == 1 )
398         plocal_next(3,2) = plocal(2,2);
399         pspeedlocal_next(3,2) = 1;
400         camelocal_next(3,2) = camelocal(2,2);
401     % if not, wait
402     else
403         plocal_next(2,2) = plocal(2,2);
404         pspeedlocal_next(2,2) = 0;
405         camelocal_next(2,2) = camelocal(2,2);
406     end
407 end
408 %3. step
409 if ( plocal(3,2) == 0.4 )
410     %if space is free, drive
411     if ( plocal(4,2) == 1 && plocal(4,1) ≠ 0.1 )
412         plocal_next(4,2) = plocal(3,2);
413         pspeedlocal_next(4,2) = 1;
414         camelocal_next(4,2) = camelocal(3,2);
415     % if not, wait
416     else
417         plocal_next(3,2) = plocal(3,2);
418         pspeedlocal_next(3,2) = 0;
419         camelocal_next(3,2) = camelocal(3,2);
420     end
421 end
422 %4. step
423 if ( plocal(4,2) == 0.4 )
424     %if space is free, drive
425     if ( plocal(5,2) == 1 )
426         plocal_next(5,2) = plocal(4,2);
427         pspeedlocal_next(5,2) = 1;
428         camelocal_next(5,2) = camelocal(4,2);
429     % if not, wait
430     else
431         plocal_next(4,2) = plocal(4,2);
432         pspeedlocal_next(4,2) = 0;
433         camelocal_next(4,2) = camelocal(4,2);
434     end
435 end
436
437 %car coming form left, going straight ahead
438 %2. step
439 if ( plocal(5,2) == 0.4 && camelocal(5,2) == 2 )
440     %if space is free, drive
441     if ( plocal(5,3) == 1 )
442         plocal_next(5,3) = plocal(5,2);
443         pspeedlocal_next(5,3) = 1;
444         camelocal_next(5,3) = camelocal(5,2);
445     % if not, wait
446     else

```



```

447         plocal_next(5,2) = plocal(5,2);
448         pspeedlocal_next(5,2) = 0;
449         camelocal_next(5,2) = camelocal(5,2);
450     end
451 end
452 %3. step
453 if ( plocal(5,3) == 0.4 )
454     %if space is free, drive
455     if ( plocal(5,4) == 1 && plocal(6,4) ≠ 0.1 )
456         plocal_next(5,4) = plocal(5,3);
457         pspeedlocal_next(5,4) = 1;
458         camelocal_next(5,4) = camelocal(5,3);
459     % if not, wait
460     else
461         plocal_next(5,3) = plocal(5,3);
462         pspeedlocal_next(5,3) = 0;
463         camelocal_next(5,3) = camelocal(5,3);
464     end
465 end
466 %4. step
467 if ( plocal(5,4) == 0.4 )
468     %if space is free, drive
469     if ( plocal(5,5) == 1 )
470         plocal_next(5,5) = plocal(5,4);
471         pspeedlocal_next(5,5) = 1;
472         camelocal_next(5,5) = camelocal(5,4);
473     % if not, wait
474     else
475         plocal_next(5,4) = plocal(5,4);
476         pspeedlocal_next(5,4) = 0;
477         camelocal_next(5,4) = camelocal(5,4);
478     end
479 end
480
481 %car coming form below, going staight ahead
482 %2. step
483 if ( plocal(5,5) == 0.4 && camelocal(5,5) == 3 )
484     %if space is free, drive
485     if ( plocal(4,5) == 1 )
486         plocal_next(4,5) = plocal(5,5);
487         pspeedlocal_next(4,5) = 1;
488         camelocal_next(4,5) = camelocal(5,5);
489     % if not, wait
490     else
491         plocal_next(5,5) = plocal(5,5);
492         pspeedlocal_next(5,5) = 0;
493         camelocal_next(5,5) = camelocal(5,5);
494     end
495 end
496 %3. step

```

```

497 if ( plocal(4,5) == 0.4 )
498     %if space is free, drive
499     if ( plocal(3,5) == 1 && plocal(3,6) ≠ 0.1 )
500         plocal_next(3,5) = plocal(4,5);
501         pspeedlocal_next(3,5) = 1;
502         camelocal_next(3,5) = camelocal(4,5);
503     % if not, wait
504     else
505         plocal_next(4,5) = plocal(4,5);
506         pspeedlocal_next(4,5) = 0;
507         camelocal_next(4,5) = camelocal(4,5);
508     end
509 end
510 %4. step
511 if ( plocal(3,5) == 0.4 )
512     %if space is free, drive
513     if ( plocal(2,5) == 1 )
514         plocal_next(2,5) = plocal(3,5);
515         pspeedlocal_next(2,5) = 1;
516         camelocal_next(2,5) = camelocal(3,5);
517     % if not, wait
518     else
519         plocal_next(3,5) = plocal(3,5);
520         pspeedlocal_next(3,5) = 0;
521         camelocal_next(3,5) = camelocal(3,5);
522     end
523 end
524
525 %car coming form right, going staight ahead
526 %2. step
527 if ( plocal(2,5) == 0.4 && camelocal(2,5) == 4 )
528     %if space is free, drive
529     if ( plocal(2,4) == 1 )
530         plocal_next(2,4) = plocal(2,5);
531         pspeedlocal_next(2,4) = 1;
532         camelocal_next(2,4) = camelocal(2,5);
533     % if not, wait
534     else
535         plocal_next(2,5) = plocal(2,5);
536         pspeedlocal_next(2,5) = 0;
537         camelocal_next(2,5) = camelocal(2,5);
538     end
539 end
540 %3. step
541 if ( plocal(2,4) == 0.4 )
542     %if space is free, drive
543     if ( plocal(2,3) == 1 && plocal(1,3) ≠ 0.1 )
544         plocal_next(2,3) = plocal(2,4);
545         pspeedlocal_next(2,3) = 1;
546         camelocal_next(2,3) = camelocal(2,4);

```

```

547     % if not, wait
548     else
549         plocal_next(2,4) = plocal(2,4);
550         pspeedlocal_next(2,4) = 0;
551         camelocal_next(2,4) = camelocal(2,4);
552     end
553 end
554 %4. step
555 if ( plocal(2,3) == 0.4 )
556     %if space is free, drive
557     if ( plocal(2,2) == 1 )
558         plocal_next(2,2) = plocal(2,3);
559         pspeedlocal_next(2,2) = 1;
560         camelocal_next(2,2) = camelocal(2,3);
561     % if not, wait
562     else
563         plocal_next(2,3) = plocal(2,3);
564         pspeedlocal_next(2,3) = 0;
565         camelocal_next(2,3) = camelocal(2,3);
566     end
567 end
568
569 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
570 %cars turning left
571
572 %car coming from above turning left
573 %1. step
574 if ( plocal(1,3) == 0.1 )
575     %if next two spaces are free and there is no car coming
576     %form right turning in front of this car, drive
577     if ( plocal(2,3) == 1 && plocal(3,4) == 1 && ...
578         plocal(4,2) ≠ 0.1 && plocal(3,3) == 1 )
579         plocal_next(2,3) = plocal(1,3);
580         pspeedlocal_next(2,3) = 1;
581         camelocal_next(2,3) = camelocal(1,3);
582     %if not, stay
583     else
584         plocal_next(1,3) = plocal(1,3);
585         pspeedlocal_next(1,3) = 0;
586         camelocal_next(1,3) = camelocal(1,3);
587     end
588 end
589 %2. step
590 if ( plocal(2,3) == 0.1 )
591     %is space is free, drive
592     if ( plocal(3,4) == 1 )
593         plocal_next(3,4) = plocal(2,3);
594         pspeedlocal_next(3,4) = 1;
595         camelocal_next(3,4) = camelocal(2,3);
596     %if not, stay

```

```

597     else
598         plocal.next(2,3) = plocal(2,3);
599         pspeedlocal.next(2,3) = 0;
600         camelocal.next(2,3) = camelocal(2,3);
601     end
602 end
603 %3 .step
604 if ( plocal(3,4) == 0.1 )
605     %if space is free and there is no car coming from the
606     %opposite side going straight ahead and no car coming
607     %from the right , drive
608     if ( plocal(4,5) == 1 && plocal(4,6) == 1 && ...
609         plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 )
610         plocal.next(4,5) = plocal(3,4);
611         pspeedlocal.next(4,5) = 1;
612         camelocal.next(4,5) = camelocal(3,4);
613     %if not, stay
614     else
615         plocal.next(3,4) = plocal(3,4);
616         pspeedlocal.next(3,4) = 0;
617         camelocal.next(3,4) = camelocal(3,4);
618     end
619 end
620 %4. step
621 if ( plocal(4,5) == 0.1 )
622     %if space is free, drive
623     if ( plocal(4,6) == 1 && plocal(5,5) ≠ 0.7 && ...
624         ¬( plocal(5,5) == 0.4 && camelocal(5,5) == 2 ) )
625         plocal.next(4,6) = plocal(4,5);
626         pspeedlocal.next(4,6) = 1;
627         camelocal.next(4,6) = camelocal(4,5);
628     %if not, stay
629     else
630         plocal.next(4,5) = plocal(4,5);
631         pspeedlocal.next(4,5) = 0;
632         camelocal.next(4,5) = camelocal(4,5);
633     end
634 end
635
636 %car coming from the left turning left
637 %1. step
638 if ( plocal(4,1) == 0.1 )
639     %if next two spaces are free and there is no car coming
640     %form right turning in front of this car, drive
641     if ( plocal(4,2) == 1 && plocal(3,3) == 1 && ...
642         plocal(5,4) ≠ 0.1 && plocal(4,3) == 1 )
643         plocal.next(4,2) = plocal(4,1);
644         pspeedlocal.next(4,2) = 1;
645         camelocal.next(4,2) = camelocal(4,1);
646     %if not, stay

```

```

647     else
648         plocal.next(4,1) = plocal(4,1);
649         pspeedlocal.next(4,1) = 0;
650         camelocal.next(4,1) = camelocal(4,1);
651     end
652 end
653 %2. step
654 if ( plocal(4,2) == 0.1 )
655     %if space is free, drive
656     if ( plocal(3,3) == 1 )
657         plocal.next(3,3) = plocal(4,2);
658         pspeedlocal.next(3,3) = 1;
659         camelocal.next(3,3) = camelocal(4,2);
660     %if not, stay
661     else
662         plocal.next(4,2) = plocal(4,2);
663         pspeedlocal.next(4,2) = 0;
664         camelocal.next(4,2) = camelocal(4,2);
665     end
666 end
667 %3 .step
668 if ( plocal(3,3) == 0.1 )
669     %if space is free and there is no car coming from the
670     %opposite side going straight ahead and no car coming
671     %from the right , drive
672     if ( plocal(2,4) == 1 && plocal(1,4) == 1 && ...
673         plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 )
674         plocal.next(2,4) = plocal(3,3);
675         pspeedlocal.next(2,4) = 1;
676         camelocal.next(2,4) = camelocal(3,3);
677     %if not, stay
678     else
679         plocal.next(3,3) = plocal(3,3);
680         pspeedlocal.next(3,3) = 0;
681         camelocal.next(3,3) = camelocal(3,3);
682     end
683 end
684 %4. step
685 if ( plocal(2,4) == 0.1 )
686     %if space is free, drive
687     if ( plocal(1,4) == 1 && plocal(2,5) ≠ 0.7 && ...
688         ¬( plocal(2,5) == 0.4 && camelocal(2,5) == 3 ) )
689         plocal.next(1,4) = plocal(2,4);
690         pspeedlocal.next(1,4) = 1;
691         camelocal.next(1,4) = camelocal(2,4);
692     %if not, stay
693     else
694         plocal.next(2,4) = plocal(2,4);
695         pspeedlocal.next(2,4) = 0;
696         camelocal.next(2,4) = camelocal(2,4);

```

```

697     end
698 end
699
700 %car coming from below turning left
701 %1. step
702 if ( plocal(6,4) == 0.1 )
703     %if next two spaces are free and there is no car coming
704     %form right turning in front of this car, drive
705     if ( plocal(5,4) == 1 && plocal(4,3) == 1 && ...
706         plocal(3,5) ≠ 0.1 && plocal(4,4) == 1 )
707         plocal_next(5,4) = plocal(6,4);
708         pspeedlocal_next(5,4) = 1;
709         camelocal_next(5,4) = camelocal(6,4);
710     %if not, stay
711     else
712         plocal_next(6,4) = plocal(6,4);
713         pspeedlocal_next(6,4) = 1;
714         camelocal_next(6,4) = camelocal(6,4);
715     end
716 end
717 %2. step
718 if ( plocal(5,4) == 0.1 )
719     %is space is free, drive
720     if ( plocal(4,3) == 1 )
721         plocal_next(4,3) = plocal(5,4);
722         pspeedlocal_next(4,3) = 1;
723         camelocal_next(4,3) = camelocal(5,4);
724     %if not, stay
725     else
726         plocal_next(5,4) = plocal(5,4);
727         pspeedlocal_next(5,4) = 0;
728         camelocal_next(5,4) = camelocal(5,4);
729     end
730 end
731 %3 .step
732 if ( plocal(4,3) == 0.1 )
733     %if space is free and there is no car coming from the
734     %opposite side going straight ahead and no car coming
735     %from the right , drive
736     if ( plocal(3,2) == 1 && plocal(3,1) == 1 && ...
737         plocal(2,2) == 1 && plocal(2,3) ≠ 0.4 )
738         plocal_next(3,2) = plocal(4,3);
739         pspeedlocal_next(3,2) = 1;
740         camelocal_next(3,2) = camelocal(4,3);
741     %if not, stay
742     else
743         plocal_next(4,3) = plocal(4,3);
744         pspeedlocal_next(4,3) = 0;
745         camelocal_next(4,3) = camelocal(4,3);
746     end

```

```

747 end
748 %4. step
749 if ( plocal(3,2) == 0.1 )
750     %if space is free, drive
751     if ( plocal(3,1) == 1 && plocal(2,2) ≠ 0.7 && ...
752         ¬( plocal(2,2) == 0.4 && camelocal(2,2) == 4 ) )
753         plocal_next(3,1) = plocal(3,2);
754         pspeedlocal_next(3,1) = 1;
755         camelocal_next(3,1) = camelocal(3,2);
756     %if not, stay
757     else
758         plocal_next(3,2) = plocal(3,2);
759         pspeedlocal_next(3,2) = 0;
760         camelocal_next(3,2) = camelocal(3,2);
761     end
762 end
763
764 %car coming from right turning left
765 %1. step
766 if ( plocal(3,6) == 0.1 )
767     %if next two spaces are free and there is no car coming
768     %form right turning in front of this car, drive
769     if ( plocal(3,5) == 1 && plocal(4,4) == 1 && ...
770         plocal(2,3) ≠ 0.1 && plocal(3,4) == 1 )
771         plocal_next(3,5) = plocal(3,6);
772         pspeedlocal_next(3,5) = 1;
773         camelocal_next(3,5) = camelocal(3,6);
774     %if not, stay
775     else
776         plocal_next(3,6) = plocal(3,6);
777         pspeedlocal_next(3,6) = 0;
778         camelocal_next(3,6) = camelocal(3,6);
779     end
780 end
781 %2. step
782 if ( plocal(3,5) == 0.1 )
783     %is space is free, drive
784     if ( plocal(4,4) == 1 )
785         plocal_next(4,4) = plocal(3,5);
786         pspeedlocal_next(4,4) = 1;
787         camelocal_next(4,4) = camelocal(3,5);
788     %if not, stay
789     else
790         plocal_next(3,5) = plocal(3,5);
791         pspeedlocal_next(3,5) = 0;
792         camelocal_next(3,5) = camelocal(3,5);
793     end
794 end
795 %3 .step
796 if ( plocal(4,4) == 0.1 )

```

```

797     %if space is free and there is no car coming from the
798     %opposite side going straight ahead and no car coming
799     %from the right , drive
800     if ( plocal(5,3) == 1 && plocal(6,3) == 1 && ...
801         plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 )
802         plocal_next(5,3) = plocal(4,4);
803         pspeedlocal_next(5,3) = 1;
804         camelocal_next(5,3) = camelocal(4,4);
805     %if not, stay
806     else
807         plocal_next(4,4) = plocal(4,4);
808         pspeedlocal_next(4,4) = 0;
809         camelocal_next(4,4) = camelocal(4,4);
810     end
811 end
812 %4. step
813 if ( plocal(5,3) == 0.1 )
814     %if space is free, drive
815     if ( plocal(6,3) == 1 && plocal(5,2) ≠ 0.7 && ...
816         ¬( plocal(5,2) == 0.4 && camelocal(5,2) == 1 ) )
817         plocal_next(6,3) = plocal(5,3);
818         pspeedlocal_next(6,3) = 1;
819         camelocal_next(6,3) = camelocal(5,3);
820     %if not, stay
821     else
822         plocal_next(5,3) = plocal(5,3);
823         pspeedlocal_next(5,3) = 1;
824         camelocal_next(5,3) = camelocal(5,3);
825     end
826 end
827
828 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
829 %cars leaving crossing
830
831 %car leaving to the top
832 if ( plocal(1,4) ≠ 1 )
833     %if space free, leave crossing with speed 1
834     if ( fp(1,1) == 1 )
835         fp_next(1,1) = 0.4;
836         fpspeed_next(1,1) = 1;
837     %if space not free, stay
838     else
839         plocal_next(1,4) = plocal(1,4);
840         pspeedlocal_next(1,4) = 0;
841         camelocal_next(1,4) = camelocal(1,4);
842     end
843 end
844
845 %car leaving to the left
846 if ( plocal(3,1) ≠ 1 )

```



```

847     %if space free, leave crossing with speed 1
848     if ( fp(2,1) == 1 )
849         fp_next(2,1) = 0.4;
850         fpspeed_next(2,1) = 1;
851     %if space not free, stay
852     else
853         plocal_next(3,1) = plocal(3,1);
854         pspeedlocal_next(3,1) = 0;
855         camelocal_next(3,1) = camelocal(3,1);
856     end
857 end
858
859 %car leaving to the bottom
860 if ( plocal(6,3) ≠ 1 )
861     %if space free, leave crossing with speed 1
862     if ( fp(3,1) == 1 )
863         fp_next(3,1) = 0.4;
864         fpspeed_next(3,1) = 1;
865     %if space not free, stay
866     else
867         plocal_next(6,3) = plocal(6,3);
868         pspeedlocal_next(6,3) = 0;
869         camelocal_next(6,3) = camelocal(6,3);
870     end
871 end
872
873 %car leaving to the bottom
874 if ( plocal(4,6) ≠ 1 )
875     %if space free, leave crossing with speed 1
876     if ( fp(4,1) == 1 )
877         fp_next(4,1) = 0.4;
878         fpspeed_next(4,1) = 1;
879     %if space not free, stay
880     else
881         plocal_next(4,6) = plocal(4,6);
882         pspeedlocal_next(4,6) = 0;
883         camelocal_next(4,6) = camelocal(4,6);
884     end
885 end
886
887 end

```

## 7.2.5 connection.m

```
1 function [cNew,dNew] = connection(aOld,bOld,cOld,posNew,m,n,length)
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %CONNECTION Decide to which street a certain street connects to
4 %
5 %INPUT:
6 %AOLD column index of intersection
7 %BOLD, row index of intersection
8 %COLD, column index in t of old position
9 %posNEW, position in new street
10 %M, number of columns in city map
11 %N, number of rows in city map
12 %LENGTH, Length of a street
13 %
14 %OUTPUT:
15 %CNEW, Column index in t of new position
16 %DNEW, Row index in t of new position
17 %
18 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
19 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
20 %Spring 2010
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22
23 %street heading up from intersection
24 if ( mod(cOld,4) == 1 )
25     %if there is a intersections above, connect to it
26     if ( aOld > 1)
27         cNew = (aOld - 2) * 4 + 3;
28         dNew = (bOld - 1) * length + posNew;
29     %otherwise connect to other side of map
30     else
31         cNew = (m - 1) * 4 + 3;
32         dNew = (bOld - 1) * length + posNew;
33     end
34 end
35
36 %street heading left from intersection
37 if ( mod(cOld,4) == 2 )
38     %if there is a intersection to the left, connect to it
39     if ( bOld > 1 )
40         cNew = aOld * 4;
41         dNew = (bOld - 2) * length + posNew;
42     %otherwise connect to other side of map
43     else
44         cNew = aOld * 4;
45         dNew = (n - 1) * length + posNew;
46     end
```

```

47 end
48
49 %street heading down from intersection
50 if ( mod(cOld,4) == 3 )
51     %if there is a intersection below, connect to it
52     if ( aOld < m )
53         cNew = aOld * 4 + 1;
54         dNew = (bOld - 1) * length + posNew;
55     %otherwise connect to other side of map
56     else
57         cNew = 1;
58         dNew = (bOld - 1) * length + posNew;
59     end
60 end
61
62 %street heading right from intersection
63 if ( mod(cOld,4) == 0 )
64     %if there is a intersection to the right, connect to it
65     if ( bOld < n )
66         cNew = (aOld - 1) * 4 + 2;
67         dNew = bOld * length + posNew;
68     %otherwise connect to other side of map
69     else
70         cNew = (aOld - 1) * 4 + 2;
71         dNew = posNew;
72     end
73 end

```

## 7.2.6 pdestination.m

```
1 function [pfirst] = pdestination
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %PDESTINATION Deside where a car is going
4 %
5 %OUTPUT:
6 %PFIRST = 0.1 car turns right
7 %       = 0.4 car goes straight ahead
8 %       = 0.7 car turns left
9 %
10 %A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
11 %and Simulation of Social Systems with MATLAB" at ETH Zurich.
12 %Spring 2010
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14
15 %decide which direction car is going
16 u = randi(12,1);
17 %probabilty 6/12 car goes straight ahead
18 if ( u ≤ 6 )
19     pfirst = 0.4;
20 end
21 %probabilty 3/12 car turns right
22 if ( u ≥ 7 && u ≤ 9 )
23     %indicate right
24     pfirst = 0.7;
25 end
26 %probabilty 3/12 car turns left
27 if ( u ≥ 10 && u ≤ 12 )
28     pfirst = 0.1;
29 end
30
31 end
```