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 veröffentlichen 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

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[1] 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 loosing essential characteristics of traffic dynamics. The model does not include passing or accidents.

[1] 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-Schreckenbenberg 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[2] 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

[2] http://en.wikipedia.org/wiki/Traffic_flow

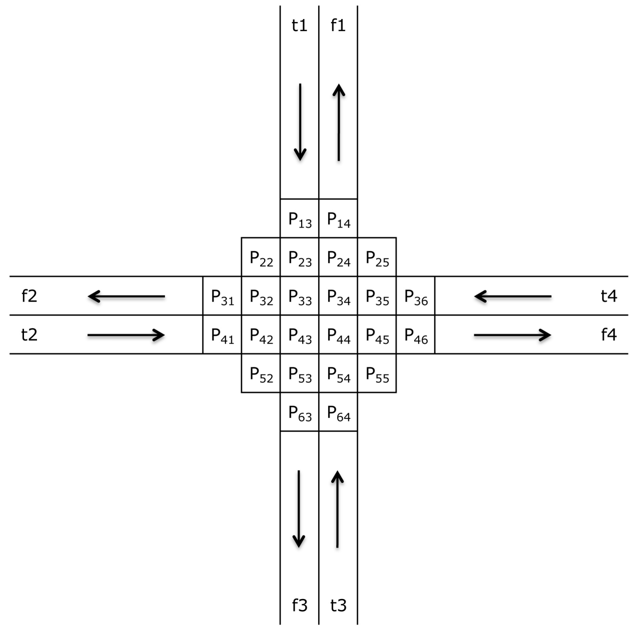

Figure 2: Crossroad cells

lane. In our model the crossroad is big enough for two cars coming form 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 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 a exit. It will take the second exit with a probability of

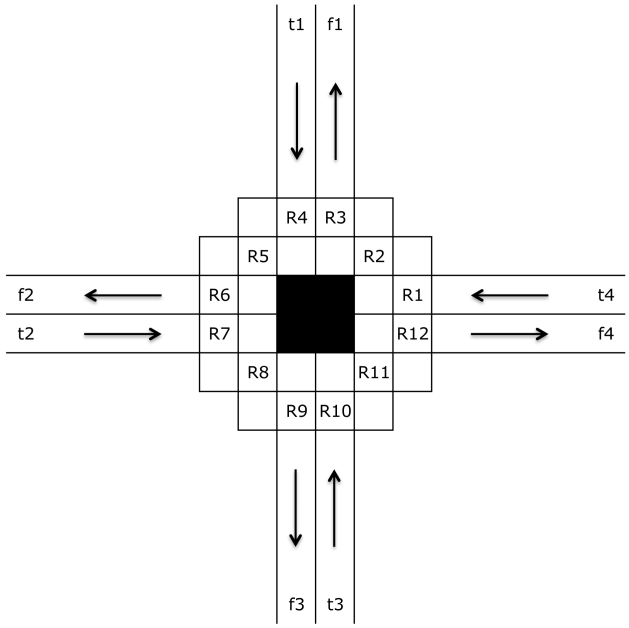

Figure 3: Roundabout cells

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 forth exit, meaning to use the roundabout to preform 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

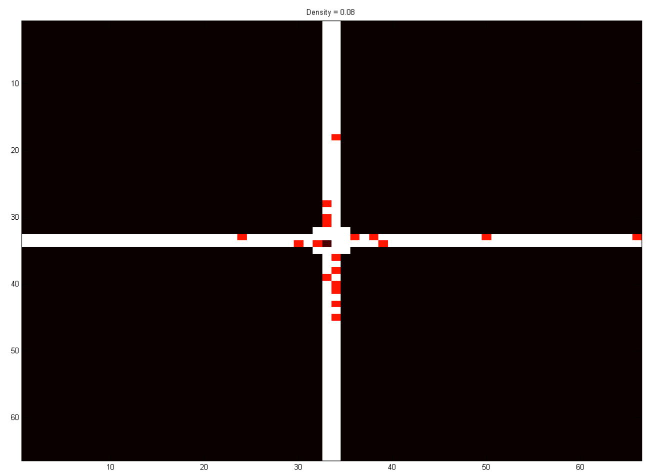

Figure 4: Simulation of a single crossroad

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 dive.

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 it's 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 form 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 form 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 a exit and a 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 a 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.

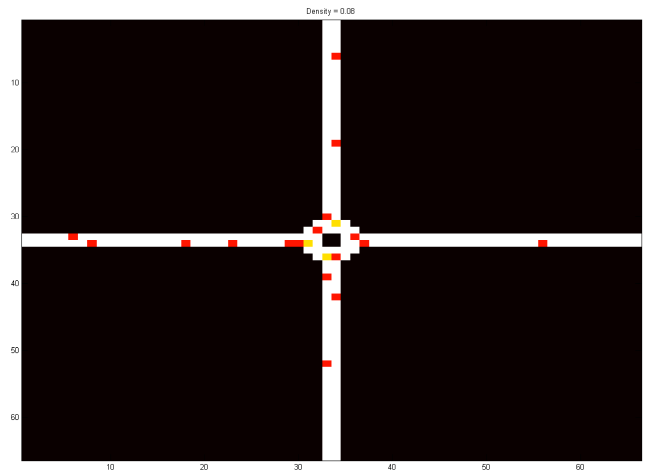

Figure 5: Simulation of a single roundabout

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 where activated simulations will be displayed. In the figure the colour of the cells symbolizes the following:

  Black $\longrightarrow$ empty space

  White $\longrightarrow$ road

  Red $\longrightarrow$ car

  Yellow $\longrightarrow$ car indicating to the right

  Dark red $\longrightarrow$ 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

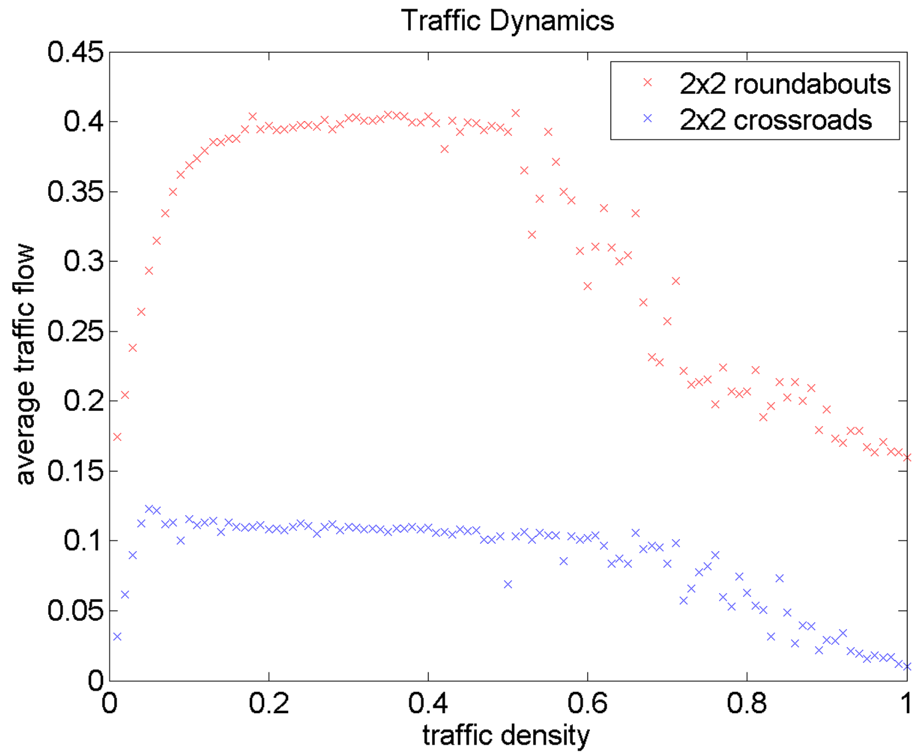

Figure 6: Full density spectrum of 2×2 pure roundabout (red) and pure crossroad (blue) configurations

Figure 6 shows the traffic flow of the two square pure city maps containing only one sort of intersection each. They simulation of the configuration with the 4 roundabout 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-

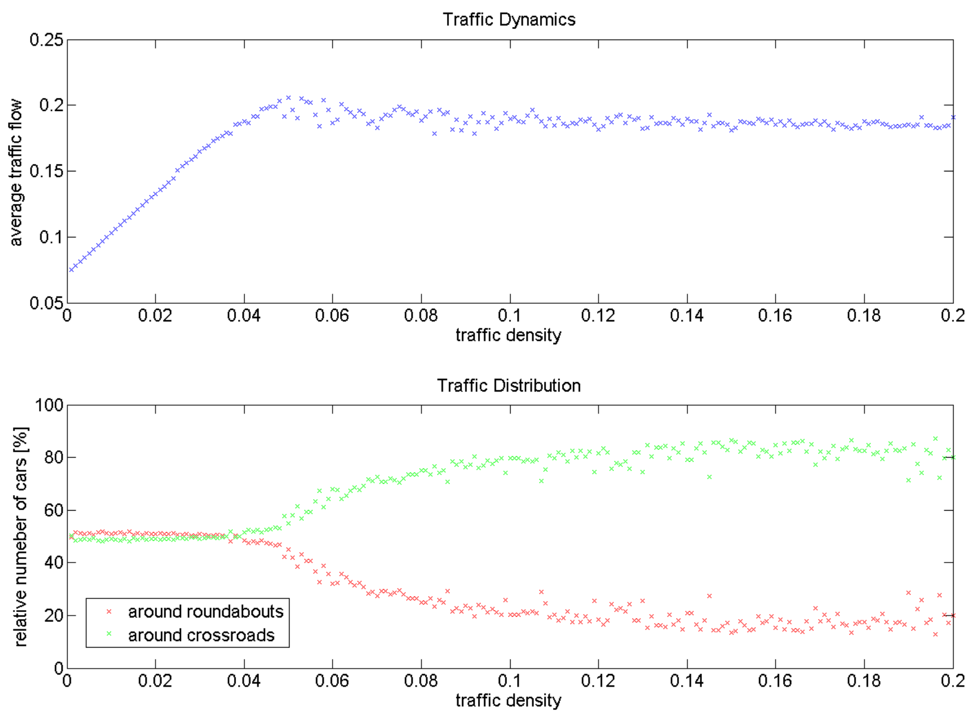

Figure 7: Traffic flow and distribution of the mixed 2x2 intersection configuration from figure 8

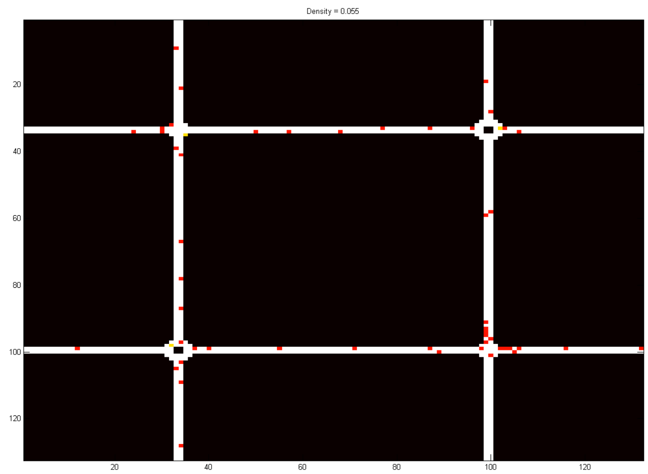

Figure 8: A mixed 2x2 intersection configuration

ure 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.

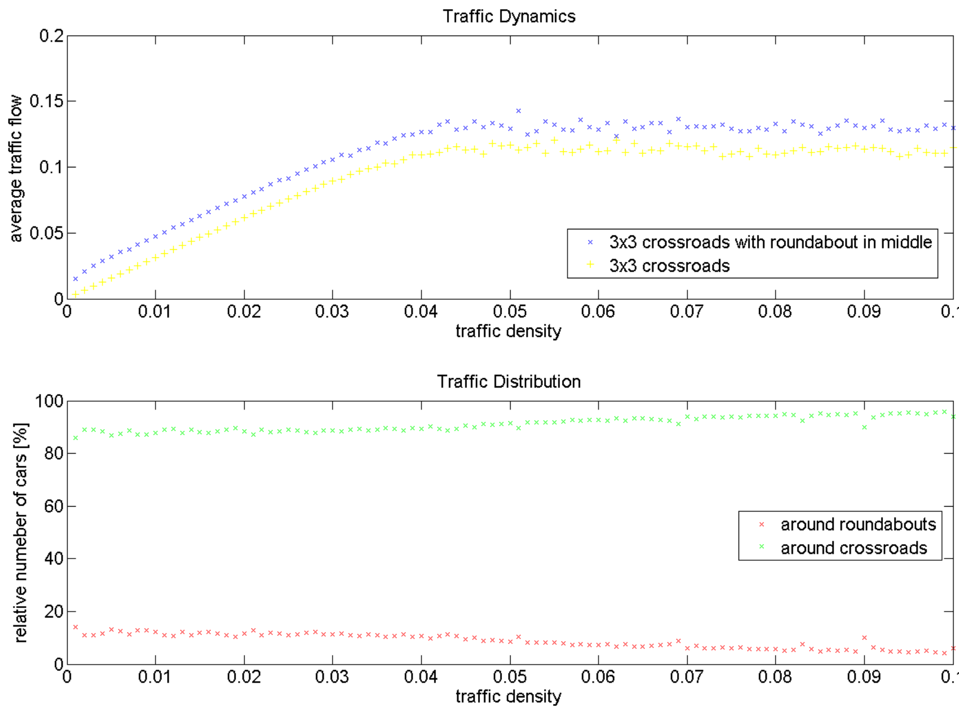

Figure 9: Traffic flow and distribution of the 3x3 crossroad configuration from figure 10

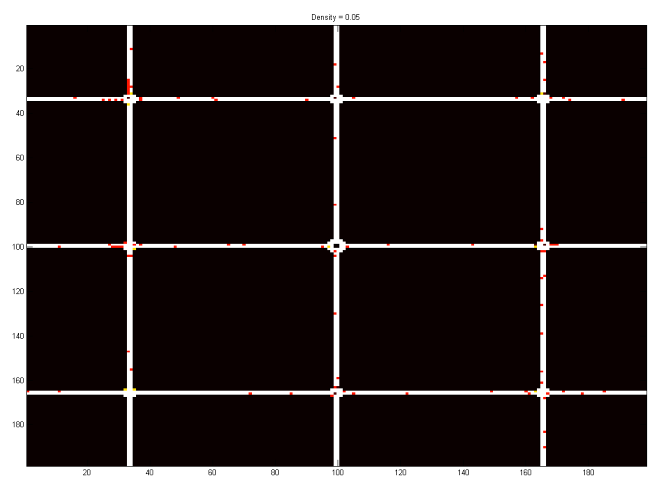

Figure 10: A 3x3 configuration with 8 crossroads around a roundabout in the center

5 Discussion

5.1 Comparison

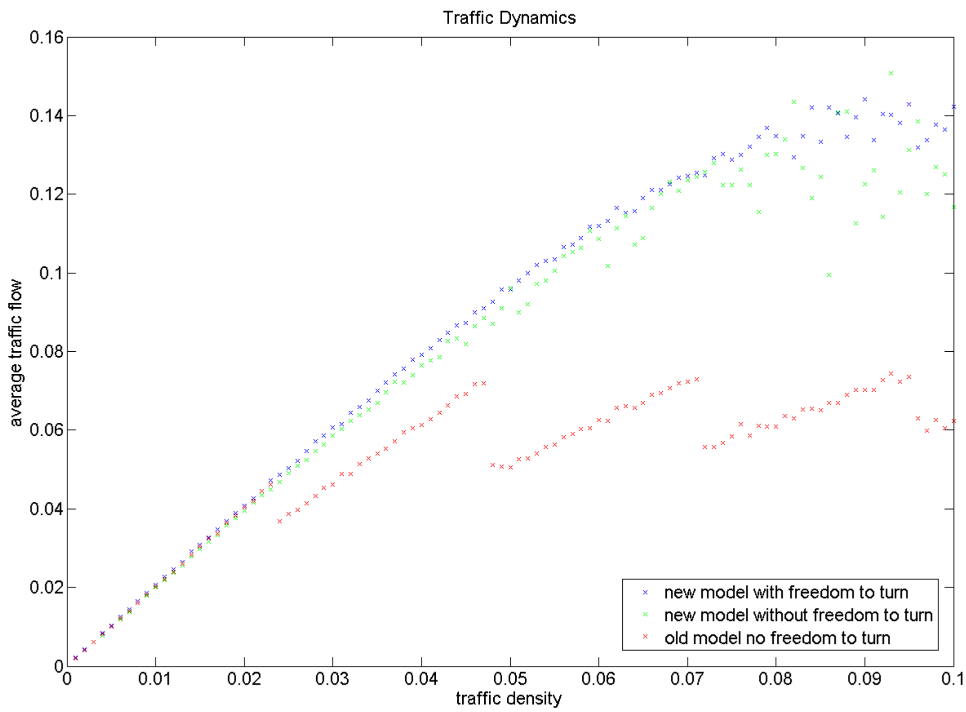

Figure 11: Comparison of models for 2x2 crossroad configurations

In figure 11 the model of Piskorec and Soller is compared to or 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

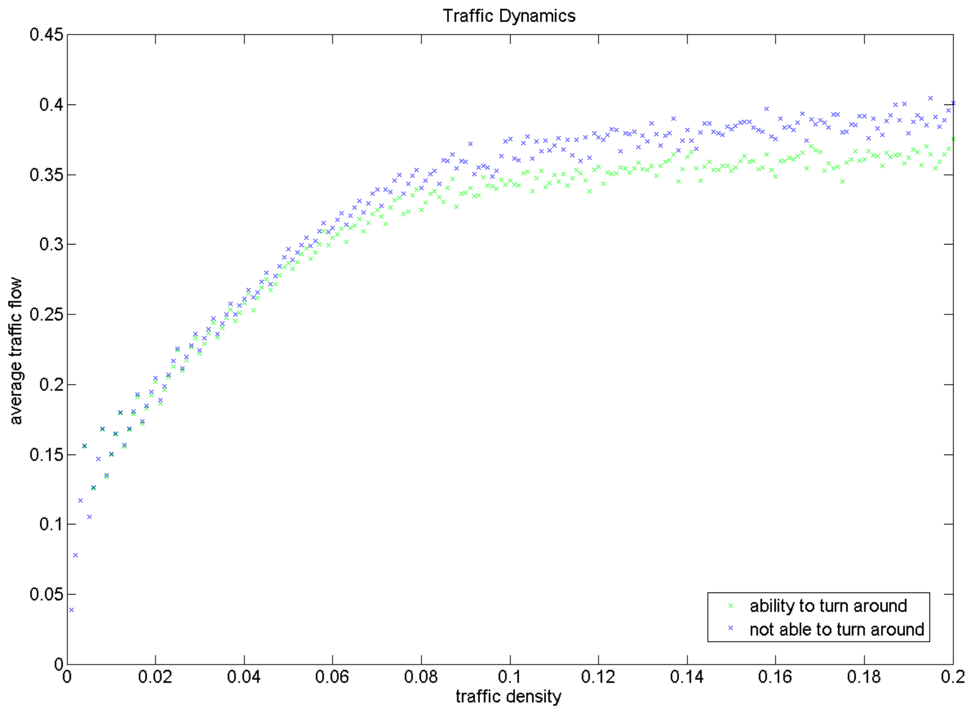

Figure 12: Comparison of single roundabouts with (green) and without(blue) the ability to turn abound

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/Cellular_automaton

7.2 Matlab-Code

7.2.1 traffic.m

```
function traffic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFIC Simulation of traffic in an city map containing roundabouts and
%crossroads.
%
%This program requires the following subprogams:
%TRAFFICSIM,ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
%
%
%User will be ask to determine city map,traffic density and whether
%simulation is to be displayed or not.
%
%The city map is entered by supplying a matrix with elements '1' for
%crossroads and '0' for roundabouts.
%
%The density can be a scalar or a vector. If the density is a scalar
%TRAFFIC will run the simulation for all densities given. The elements must
%be in the range of [0,1].
%
%If Users chooses to display simulation (by entering 'y') a figure will
%open showing the animation:
%-Black cells simbolize empty space
%-White cells simbolize road
%-Red cells simbolize cars
%-Yellow cells simbolize cars indicating to the right
%-Dark red celss simbolize cars indicating to the left
%
%After all simulations have finished TRAFFIC plots the average traffic flow
%versus the traffic density. If city map is a mix of crossroad and
%roundabouts the traffic distribution (cars around roundabouts or around
%crossroads) versus traffic density is also plotted.
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;

%promt city road configutation
c = input(['\nenter city map\n\ngive matrix elements: ', ...
    'Priority to the right (=1) and Roundabout (=0) \n\n', ...
    'i.e. [1 0 0;1 1 0;0 1 1]\n\n']);

%check c
```

```
[c_m,c_n] = size(c);
for a = 1:c_m
    for b = 1:c_n
        if ( c(c_m,c_n) ≠ 1 && c(c_m,c_n) ≠ 0 )
            disp('Elements must be 0 or 1');
            return
        end
    end
end
%check if city map is a mix of crossroads and roundaoubts or if it made up
%of purely one or the other
if ( sum(sum(c)) == c_m * c_n || sum(sum(c)) == 0 )
    mix = false;
else
    mix = true;
end

%promt traffic density
d = input('\nenter traffic density: ');
%check d
if ( max(d) > 1 || min(d) < 0)
    disp('density must be in range [0,1]');
    return
end

%ask if simulation should be displayed
show = input('\ndisplay simulation graphically? yes (=y) or no (=n) ','s');

%average flow and distributions for every density suppied
avFlow = zeros(1,max(size(d)));
avRo = zeros(1,max(size(d)));
avCr = zeros(1,max(size(d)));

if  ( show == 'y' || show == 'n' )
    %if wanted run simulation with graphics
    if ( show == 'y' )
        for di=1:max(size(d))
            [avFlow(di),avRo(di),avCr(di)] = trafficsim(d(di),c,true);
        end
    %if animation undesired run simulation without graphics
    else
        for di=1:max(size(d))
            [avFlow(di),avRo(di),avCr(di)] = trafficsim(d(di),c,false);
        end
    end

    figure(2);
    %is city map is a mix of roundabout and crossroads, plot distribution
    if ( mix )
        %plot relativ number of cars at roundabouts and number of cars at
```

```
        %crossroads versus traffic density
        subplot(2,1,2);
        plot(d,avRo*100,'rx',d,avCr*100,'gx');
        set(gca,'FontSize',16);
        title('Traffic Distribution');
        xlabel('traffic density');
        ylabel('relative numeber of cars [%]');
        legend('around roundabouts','around crossroads');
        ylim([0 100]);
        subplot(2,1,1);
    end

    %plot traffic flow versus traffic density
    plot(d,avFlow,'x');
    set(gca,'FontSize',16);
    title('Traffic Dynamics');
    xlabel('traffic density');
    ylabel('average traffic flow');
    %ylim([0 0.5]);
else
    disp('Input must be y or n!');
end
end
```

7.2.2 trafficsim.m

```
function [averageFlow,avCaRo,avCaCr] = trafficsim(density,config,display)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TRAFFICSIM Simulation of traffic in an city map containing roundabouts and
%crossroads.
%
%Output:
%AVERAGEFLOW, Average traffic flow for given city map and density
%AVCARO, Average amount of cars around roundabouts
%AVCACR, Average amount of cars around crossroads
%
%INPUT:
%DENSITY, Traffic density
%CONFIG, City map
%DISPlAY, Turn graphics on 'true' or off 'false'
%
%This program requires the following subprogams:
%ROUNDABOUT,CROSSROAD,CONNECTION,PDESTINATION
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%dawde probability
dawdleProb = 0.2;
%street length (>5)
l = 30;
%number of iterations
nIt=1000;

%dimensions of config, how many intersections in x and y direction are
%there?
[config_m,config_n] = size(config);

%in streets cell values indicate the following:
%0.4 means there is a car in this position (red in figure)
%1 means there is no car in this position (white in figure)

%initialize matrices for streets heading toward intersections
t = ones(4*config_m,l*config_n);
tspeed = zeros(4*config_m,l*config_n);
%number of elements in t
tsize = sum(sum(t));

%initialize matrices for street leading away from intersections
f = ones(4*config_m,l*config_n);
```

```
fspeed = zeros(4*config_m,l*config_n);

%initialize matrices for roundabouts
r = ones(config_m,12*config_n);
rspeed = zeros(config_m,12*config_n);
rex = zeros(config_m,12*config_n);

%initialize matrices for crossings with priority to the right
p = ones(6*config_m,6*config_n);
pspeed = zeros(6 *config_m,6*config_n);
came = zeros(6*config_m,6*config_n);
%deadlock prevention
deadlock = zeros(config_m,config_n);

%initialaize map
map = zeros(config_m*(2*l+6),config_n*(2*l+6));
%initialize gap
gap = 0;

%initialize flow calculation variables
avSpeedIt = zeros(nIt+1,1);
%counter for cars around crossroads
numCaCrIt = zeros(nIt+1,1);
%counter for cars around crossroads
numCaRoIt = zeros(nIt+1,1);

%distribute cars randomly on streets for starting point
overall_length = sum(sum(t)) + sum(sum(f));
numCars = ceil(density * overall_length);
q = 1;

while ( q ≤ numCars )
    w = randi(overall_length,1);
    if ( w ≤ tsize )
        if ( t(w) == 1)
            t(w) = 0.4;
            tspeed(w) = randi(5,1);
            q = q + 1;
        end
    end
    if ( w > tsize )
        if ( f(w-tsize) == 1 )
            f(w-tsize) = 0.4;
            fspeed(w-tsize) = randi(5,1);
            q = q +1 ;
        end
    end
end
```

```
%iterate over time
for time = 1:nIt+1

    %clear values for next step
    t_next = ones(4*config_m,l*config_n);
    tspeed_next = zeros(4*config_m,l*config_n);
    f_next = ones(4*config_m,l*config_n);
    fspeed_next = zeros(4*config_m,l*config_n);
    r_next = ones(config_m,12*config_n);
    rspeed_next = zeros(config_m,12*config_n);
    rex_next = zeros(config_m,12*config_n);
    p_next = ones(6*config_m,6*config_n);
    pspeed_next = ones(6*config_m,6*config_n);
    came_next = zeros(6*config_m,6*config_n);
    deadlock_next = zeros(config_m,config_n);

    %iterate over all intersection
    for a = 1:config_m
        for b = 1:config_n

            %define Index starting points for each intersection
            tI_m = (a - 1) * 4;
            tI_n = (b - 1) * l;
            mapI_m = (a - 1) * (2 * l + 6);
            mapI_n = (b - 1) * (2 * l + 6);

            %positions outside intersections
            %for every intersection iterate along streets
            for c = tI_m + 1:tI_m +4
                for d = tI_n + 1:tI_n+l

                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    %streets to intersections

                    %deal with position directly in front of intersection
                    %separately later
                    if ( mod(d,l) ≠ 0 )
                        %if there is a car in this position, apply
                        %NS-Model
                        if ( t(c,d) == 0.4 )
                            %Nagel-Schreckenberg-Model
                            %NS 1. step: increase velocity if < 5
                            v = tspeed(c,d);
                            if ( v < 5)
                                v = v + 1;
                            end

                            %NS 2. step: adapt speed to gap
                            %how big is gap (to car ahead or intersection)?
                            e = 1;
```

```
        while (e ≤ 5 && d + e ≤ b * l && ...
                t(c,d+e) == 1 )
            e = e + 1;
        end
        gap = e - 1;
        %reduce speed if gap is too small
        if ( v > gap )
            v = gap;
        end

        %NS 3. step: dawdle
        if ( rand < dawdleProb && v ≠ 0 )
            v = v - 1;
        end

        %NS 4. step: drive, move cars tspeed(c,d) cells
        %forward
        %new position
        t_next(c,d+v) = 0.4;
        tspeed_next(c,d+v) = v;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%street from intersections

if ( f(c,d) == 0.4 )
    %Nagel-Schreckenberg-Model
    %NS 1. step: increase velocity if < 5
    v = fspeed(c,d);
    if ( v < 5)
        v = v + 1;
    end

    %NS 2.step: adpat speed to gap
    %how big is gap (to car ahead)?
    e = 1;
    while ( e ≤ 5 )
        %if gap is bigger than distance to edge,connect
        %steets
        if ( d + e > b * l )
            %testing position in new street
            hh = d + e - b * l;
            %connect to next street
            [ec,ed]=connection(a,b,c,hh, ...
                config_m,config_n,l);
            while ( t(ec,ed) == 1 && e ≤ 5 )
                e = e + 1;
                %testing position in new street
                hh = d + e - b * l;
```

```
                        %connect to next street
                        [ec,ed]=connection(a,b,c,hh, ...
                            config_m,config_n,l);
                    end
                    gap = e - 1;
                    e = 6;
                else
                    if ( f(c,d+e) == 1 )
                        e = e + 1;
                        if ( e == 6 )
                            gap = 5;
                        end
                    else
                        gap = e - 1;
                        e = 6;
                    end
                end
            end
            %reduce speed if gap is too small
            if ( v > gap )
                v = gap;
            end

            %NS 3. step: dawdle
            if ( rand ≤ dawdleProb && v ≠ 0 )
                v = v - 1;
            end

            %NS 4. step: drive, move cars fspeed(c,d) cells
            %forward
            %if new position is off this street, connect
            %streets
            if ( d + v > b * l )
                %position in new street
                hhh =  d + v - b * l;
                %connect next street
                [ec,ed] = connection(a,b,c,hhh, ...
                    config_m,config_n,l);
                t_next(ec,ed) = 0.4;
                tspeed_next(ec,ed) = v;
            else
                f_next(c,d+v) = 0.4;
                fspeed_next(c,d+v) = v;
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%roundabouts
```

```
            %check if intersection is a roundabout
            if  ( config(a,b) == 0 )
                %define index strating point for this roundabout
                rI_n = (b - 1) * 12;

                %do roundabout calculations for this roundabout and time
                %step
                %call ROUNDABOUT
                [t_next(tI_m+1:tI_m+4,tI_n+l), ...
                    tspeed_next(tI_m+1:tI_m+4,tI_n+l), ...
                    f_next(tI_m+1:tI_m+4,tI_n+1), ...
                    fspeed_next(tI_m+1:tI_m+4,tI_n+1), ...
                    r_next(a,rI_n+1:rI_n+12), ...
                    rspeed_next(a,rI_n+1:rI_n+12), ...
                    rex_next(a,rI_n+1:rI_n+12)] = ...
                    roundabout(t(tI_m+1:tI_m+4,tI_n+l), ...
                    f(tI_m+1:tI_m+4,tI_n+1), ...
                    r(a,rI_n+1:rI_n+12), ...
                    rex(a,rI_n+1:rI_n+12), ...
                    t_next(tI_m+1:tI_m+4,tI_n+l), ...
                    tspeed_next(tI_m+1:tI_m+4,tI_n+l), ...
                    f_next(tI_m+1:tI_m+4,tI_n+1), ...
                    fspeed_next(tI_m+1:tI_m+4,tI_n+1));

                %write roundabout into map
                map(mapI_m+l+1:mapI_m+l+6,mapI_n+l+1:mapI_n+l+6) = ...
                    [ 0 1 r(a,rI_n+4) r(a,rI_n+3) 1 0;
                    1 r(a,rI_n+5) 1 1 r(a,rI_n+2) 1;
                    r(a,rI_n+6) 1 0 0 1 r(a,rI_n+1);
                    r(a,rI_n+7) 1 0 0 1 r(a,rI_n+12);
                    1 r(a,rI_n+8) 1 1 r(a,rI_n+11) 1;
                    0 1 r(a,rI_n+9) r(a,rI_n+10) 1 0];

                %add cars around this crossroad in this time step to
                %counter for cars around crossroads
                for v = tI_m+1:tI_m+4
                    for w = tI_n+1:tI_n+l
                        if ( t(v,w) ≠ 1 )
                            numCaRoIt(time) = numCaRoIt(time) + 1;
                        end
                        if ( f(v,w) ≠ 1 )
                            numCaRoIt(time) = numCaRoIt(time) + 1;
                        end
                    end
                end
                for y = rI_n+1:rI_n+12
                    if ( r(a,y) ≠ 1 )
                        numCaRoIt(time) = numCaRoIt(time) + 1;
                    end
```

```
            end

        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %crossroads

        %check if intersection is a crossing with priority to the right
        if ( config(a,b) == 1 )
            %define index strating points for this crossraod
            pI_m = (a - 1) * 6;
            pI_n = (b - 1) * 6;

            %do crossroad calculations for this crossroad and time step
            %call CROSSROAD
            [t_next(tI_m+1:tI_m+4,tI_n+l), ...
                tspeed_next(tI_m+1:tI_m+4,tI_n+l), ...
                f_next(tI_m+1:tI_m+4,tI_n+1), ...
                fspeed_next(tI_m+1:tI_m+4,tI_n+1), ...
                p_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                pspeed_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                came_next(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                deadlock_next(a,b), ...
                map(mapI_m+l+1:mapI_m+l+6,mapI_n+l+1:mapI_n+l+6)] ...
                = crossroad(t(tI_m+1:tI_m+4,tI_n+l), ...
                f(tI_m+1:tI_m+4,tI_n+1), ...
                p(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                came(pI_m+1:pI_m+6,pI_n+1:pI_n+6), ...
                deadlock(a,b), ...
                t_next(tI_m+1:tI_m+4,tI_n+l), ...
                tspeed_next(tI_m+1:tI_m+4,tI_n+l), ...
                f_next(tI_m+1:tI_m+4,tI_n+1), ...
                fspeed_next(tI_m+1:tI_m+4,tI_n+1));

            %add cars around this roundabout in this time step to
            %counter for cars around roundabouts
            for v = tI_m+1:tI_m+4
                for w = tI_n+1:tI_n+l
                    if ( t(v,w) ≠ 1 )
                        numCaCrIt(time) = numCaCrIt(time) + 1;
                    end
                    if ( f(v,w) ≠ 1 )
                        numCaCrIt(time) = numCaCrIt(time) + 1;
                    end
                end
            end
            for x = pI_m+1:pI_m+6
                for y = pI_n+1:pI_n+6
                    if ( came(x,y) ≠ 0 )
                        numCaCrIt(time) = numCaCrIt(time) + 1;
```

```
                    end
                end
            end

        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %write streets into map
        for i = 1:l
            map(mapI_m+i,mapI_n+l+3) = t(tI_m+1,tI_n+i);
            map(mapI_m+l+4,mapI_n+i) = t(tI_m+2,tI_n+i);
            map(mapI_m+2*l+7-i,mapI_n+l+4) = t(tI_m+3,tI_n+i);
            map(mapI_m+l+3,mapI_n+2*l+7-i) = t(tI_m+4,tI_n+i);
            map(mapI_m+l+1-i,mapI_n+l+4) = f(tI_m+1,tI_n+i);
            map(mapI_m+l+3,mapI_n+l+1-i) = f(tI_m+2,tI_n+i);
            map(mapI_m+l+6+i,mapI_n+l+3) = f(tI_m+3,tI_n+i);
            map(mapI_m+l+4,mapI_n+l+6+i) = f(tI_m+4,tI_n+i);
        end

        %illustrate trafic situation (now not of next time step)
        if ( display)
            figure(1);
            imagesc(map);
            colormap(hot);
            titlestring = sprintf('Density = %g',density);
            title(titlestring);
            drawnow;
        end


    end
end

%calculate average velosity per time step
avSpeedIt(time) = ( sum(sum(tspeed)) + sum(sum(fspeed)) + ...
    sum(sum(rspeed)) + sum(sum(pspeed)) ) / numCars;

%pause(1);

%move on time step on
t = t_next;
tspeed = tspeed_next;
f = f_next;
fspeed = fspeed_next;
r = r_next;
rspeed = rspeed_next;
rex = rex_next;
p = p_next;
pspeed = pspeed_next;
came = came_next;
```

```
    deadlock = deadlock_next;
end

%overall average velocity
averageSpeed = sum(avSpeedIt) / max(size(avSpeedIt));
%overall average flow
averageFlow = density * averageSpeed;

%average relative amount of cars around roundabouts
avCaRo = sum(numCaRoIt) / ( max(size(numCaRoIt)) * numCars );
%average relative amount of cars around crossroads
avCaCr = sum(numCaCrIt) / ( max(size(numCaCrIt)) * numCars );

end
```

7.2.3 roundabout.m

```
function [tr_next, ...
    trspeed_next, ...
    fr_next, ...
    frspeed_next, ...
    rlocal_next, ...
    rspeedlocal_next, ...
    rexlocal_next] ...
    = roundabout(tr, ...
    fr, ...
    rlocal, ...
    rexlocal, ...
    tr_next, ...
    trspeed_next, ...
    fr_next,...
    frspeed_next)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ROUNDABOUT Calculation of update for a certain roundabout, density and
%time step
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%in roundabout cell values indicate if car is about to leave roundabout:
%0.4 means car is not taking next exit (red in figure)
%0.7 means car is taking next exit (yellow in figure)
%1 means no car in this position (white in figure)

%clear local next variables
rlocal_next = ones(1,12);
rspeedlocal_next = zeros(1,12);
rexlocal_next = zeros(1,12);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in front of roundabout

for k = 1:4
    if ( tr(k,1) == 0.4 )
        %entering roundabout with velocity 1 when possible
        %roundabout position index
        iR = mod(3*k+1,12);
        if ( rexlocal(k*3) ≤ 1 && rlocal(iR) == 1 )
            %enter roundabout
            %decide which exit car is going to take
            u = randi(12,1);
```

```
            %probabilty 6/12 take it takes 2. exit
            if ( u ≤ 6 )
                rexlocal_next(iR) = 2;
                rlocal_next(iR) = 0.4;
                rspeedlocal_next(iR) = 1;
            end
            %probabilty 3/12 take it takes 1. exit
            if ( u ≥ 7 && u ≤ 9 )
                rexlocal_next(iR) = 1;
                %indicate
                rlocal_next(iR) = 0.7;
                rspeedlocal_next(iR) = 1;
            end
            %probabilty 3/12 take it takes 3. exit
            if ( u ≥ 10 && u ≤ 12 )
                rexlocal_next(iR) = 3;
                rlocal_next(iR) = 0.4;
                rspeedlocal_next(iR) = 1;
            end
            %probabilty 1/12 take it takes 4. exit (turns around)
            %if ( u == 12 )
            %    rexlocal_next(iR) = 4;
            %    rlocal_next(iR) = 0.4;
            %    rspeedlocal_next(iR) = 1;
            %end

        %car waiting in front of roundabout
        else
            tr_next(k,1) = tr(k,1);
            trspeed_next(k,1) = 0;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%car in roundabout

for j = 1:12
    if ( rlocal(j) ≠ 1 )

        %cars in roundabout not at an exit
        if (mod(j,3) ≠ 0 )
            %if space free, move one forward
            if ( rlocal(j+1) == 1 )
                %take new position
                rlocal_next(j+1) = rlocal(j);
                rspeedlocal_next(j+1) = 1;
                rexlocal_next(j+1) = rexlocal(j);
            %if no space free, stay
            else
```

```
                rlocal_next(j) = rlocal(j);
                rspeedlocal_next(j) = 0;
                rexlocal_next(j) = rexlocal(j);
            end

        %car at an exit
        else

            %if car is at its exit
            if ( rexlocal(j) == 1 )
                %if space free, leave roundabout
                if ( fr(j/3,1) == 1 )
                    fr_next(j/3,1) = 0.4;
                    frspeed_next(j/3,1) = 1;
                %if no space free, stay
                else
                    rlocal_next(j) = rlocal(j);
                    rspeedlocal_next(j) = 0;
                    rexlocal_next(j) = rexlocal(j);
                end

            %car at an exit but not the one its taking
            else
                %connect r(12) with r(1)
                if (j == 12 )
                    %if space free, move one forward and decrease exit
                    %counter
                    if ( rlocal(1) == 1 )
                        %decrease exit by one
                        rexlocal_next(1) = rexlocal(12) - 1;
                        rspeedlocal_next(1) = 1;
                        if ( rexlocal_next(1) == 1 )
                            %indicate
                            rlocal_next(1) = 0.7;
                        else
                            rlocal_next(1) = 0.4;
                        end
                    %if no space free, stay
                    else
                        rlocal_next(12) = rlocal(12);
                        rspeedlocal_next(12) = 0;
                        rexlocal_next(12) = rexlocal(12);
                    end
                else
                    %if space free, move one forward and decrease exit
                    %counter
                    if ( rlocal(j+1) == 1 )
                        %decrease exit by one
                        rexlocal_next(j+1) = rexlocal(j) - 1;
                        rspeedlocal_next(j+1) = 1;
```

```
                        if ( rexlocal_next(j+1) == 1 )
                            %indicate
                            rlocal_next(j+1) = 0.7;
                        else
                            rlocal_next(j+1) = 0.4;
                        end
                    %if no space free, stay
                    else
                        rlocal_next(j) = rlocal(j);
                        rspeedlocal_next(j) = 0;
                        rexlocal_next(j) = rexlocal(j);
                    end
                end
            end
        end
    end
end

end
```

7.2.4 crossroad.m

```
function [tp_next, ...
    tpspeed_next, ...
    fp_next, ...
    fpspeed_next, ...
    plocal_next ...
    pspeedlocal_next, ...
    camelocal_next, ...
    deadlocklocal_next, ...
    plocal] ...
    = crossroad(tp, ...
    fp, ...
    plocal, ...
    camelocal, ...
    deadlocklocal, ...
    tp_next, ...
    tpspeed_next, ...
    fp_next, ...
    fpspeed_next)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CROSSROAD Calculation of update for a certain crossroad, density and time
%step
%
%This program requires the following subprogams:
%PDESTINATION
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%in crossroad cell values indicate where cars is going:
%0.1 means car is turning left (dark red in figure)
%0.4 means car is going straight ahead (red in figure)
%0.7 means car is turning right (yellow in figure)
%1 means no car in this position (white in figure)

%clear local next variables
plocal_next = ones(6,6);
pspeedlocal_next = zeros(6,6);
camelocal_next = zeros(6,6);
deadlocklocal_next = 0;

%'paint' unused corners of plocal black
plocal(1,1) = 0;
plocal(1,6) = 0;
plocal(6,1) = 0;
```

```
plocal(6,6) = 0;
plocal(1,2) = 0;
plocal(1,5) = 0;
plocal(2,1) = 0;
plocal(2,6) = 0;
plocal(5,1) = 0;
plocal(5,6) = 0;
plocal(6,2) = 0;
plocal(6,5) = 0;

%key to unlock deadlock for this iteration and this
%intersection
unlock = randi(4,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars in front of crossroad

%car waiting from above
if ( tp(1,1) == 0.4 )
    %if space is free and there is no car coming from the
    %left going straight ahead already in crossing, enter
    if ( plocal(1,3) == 1 && camelocal(2,3) ≠ 4 && ...
            camelocal(2,4) ≠ 1 && ...
            ¬( camelocal(2,5) == 4 && plocal(2,5) == 0.4 ))
        %decide where car is heading
        plocal_next(1,3) = pdestination;
        pspeedlocal_next(1,3) = 1;
        %mark which entrance car came from
        camelocal_next(1,3) = 1;
    %if not wait
    else
        tp_next(1,1) = tp(1,1);
        tpspeed_next(1,1) = 0;
    end
end

%car waiting from left
if ( tp(2,1) == 0.4 )
    %if space is free and there is no car coming from the
    %left going straight ahead already in crossing, enter
    if ( plocal(4,1) == 1 && camelocal(4,2) ≠ 1 && ...
            camelocal(3,2) ≠ 1 && ...
            ¬( camelocal(2,2) == 1 && plocal(2,2) == 0.4 ) )
        %decide where car is heading
        plocal_next(4,1) = pdestination;
        pspeedlocal_next(4,1) = 1;
        %mark which entrance car came from
        camelocal_next(4,1) = 2;
    %if not wait
    else
```

```
        tp_next(2,1) = tp(2,1);
        tpspeed_next(2,1) = 0;
    end
end

%car waiting from below
if ( tp(3,1) == 0.4 )
    %if space is free and there is no car coming from the
    %left going straight ahead already in crossing, enter
    if ( plocal(6,4) == 1 && camelocal(5,4) ≠ 2 && ...
            camelocal(5,3) ≠ 2 && ...
            ¬( camelocal(5,2) == 2 && plocal(5,2) == 0.4 ) )
        %decide where car is heading
        plocal_next(6,4) = pdestination;
        pspeedlocal_next(6,4) = 1;
        %mark which entrance car came from
        camelocal_next(6,4) = 3;
    %if not wait
    else
        tp_next(3,1) = tp(3,1);
        tpspeed_next(3,1) = 0;
    end
end

%car waiting from right
if ( tp(4,1) == 0.4 )
    %if space is free and there is no car coming from the
    %left going straight ahead already in crossing, enter
    if ( plocal(3,6) == 1 && camelocal(3,5) ≠ 3 && ...
            camelocal(4,5) ≠ 3 && ...
            ¬( camelocal(5,5) == 3 && plocal(5,5) == 0.4 ) )
        %decide where car is heading
        plocal_next(3,6) = pdestination;
        pspeedlocal_next(3,6) = 1;
        %mark which entrance car came from
        camelocal_next(3,6) = 4;
    %if not wait
    else
        tp_next(4,1) = tp(4,1);
        tpspeed_next(4,1) = 0;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars going turning right step 1

%car coming form above, turning right
%1. step
if ( plocal(1,3) == 0.7 )
    %if space free, car has right of way and can drive
```

```
    if ( plocal(2,2) == 1 && plocal(2,3) ≠ 0.4 )
        plocal_next(2,2) = plocal(1,3);
        pspeedlocal_next(2,2) = 1;
        camelocal_next(2,2) = camelocal(1,3);
    % if space not free, stay
    else
        plocal_next(1,3) = plocal(1,3);
        pspeedlocal_next(1,3) = 0;
        camelocal_next(1,3) = camelocal(1,3);
    end
end

%car coming form left, turning right
%1. step
if ( plocal(4,1) == 0.7 )
    %if space free, car has right of way and can drive
    if ( plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 )
        plocal_next(5,2) = plocal(4,1);
        pspeedlocal_next(5,2) = 1;
        camelocal_next(5,2) = camelocal(4,1);
    % if space not free, stay
    else
        plocal_next(4,1) = plocal(4,1);
        pspeedlocal_next(4,1) = 0;
        camelocal_next(4,1) = camelocal(4,1);
    end
end

%car coming form below, turning right
%1. step
if ( plocal(6,4) == 0.7 )
    %if space free, car has right of way and can drive
    if ( plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 )
        plocal_next(5,5) = plocal(6,4);
        pspeedlocal_next(5,5) = 1;
        camelocal_next(5,5) = camelocal(6,4);
    % if space not free, stay
    else
        plocal_next(6,4) = plocal(6,4);
        pspeedlocal_next(6,4) = 0;
        camelocal_next(6,4) = camelocal(6,4);
    end
end

%car coming form right, turning right
%1. step
if ( plocal(3,6) == 0.7 )
    %if space free, car has right of way and can drive
    if ( plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 )
        plocal_next(2,5) = plocal(3,6);
```

```
        pspeedlocal_next(2,5) = 1;
        camelocal_next(2,5) = camelocal(3,6);
    % if space not free, stay
    else
        plocal_next(3,6) = plocal(3,6);
        pspeedlocal_next(3,6) = 0;
        camelocal_next(3,6) = camelocal(3,6);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars going straight ahead step 1

%car coming form above, going stright ahead
%1. step
if ( plocal(1,3) == 0.4 )
    %if space is free and there are no are coming from the
    %right or is there has been a deadlock and driver have
    %agreed by hand signal to let this car go, dive
    %!warning: only works if this step is done after update
    %of cars in front of crossraod!
    if ( plocal(2,2) == 1 && plocal(2,3) ≠ 0.4 && ...
            ( ( tp_next(2,1) == 1 && plocal_next(4,1) == 1 && ...
            plocal(4,1) == 1 ) || ( deadlocklocal == 4 && unlock == 1 ) ) )
        plocal_next(2,2) = plocal(1,3);
        pspeedlocal_next(2,2) = 1;
        camelocal_next(2,2) = camelocal(1,3);
        %no deadlock, clear deadlock counter
        deadlocklocal_next = 0;
    % if not, stay
    else
        plocal_next(1,3) = plocal(1,3);
        pspeedlocal_next(1,3) = 0;
        camelocal_next(1,3) = camelocal(1,3);
        %increase deadlock counter, if it reaches 4 a
        %deadlock occurs and will have to be solve in next
        %time step by a hand signals between drivers
        deadlocklocal_next = deadlocklocal_next + 1;
    end
end

%car coming form left, going stright ahead
%1. step
if ( plocal(4,1) == 0.4 )
    %if space is free and there are no are coming from the
    %right or is there has been a deadlock and driver have
    %agreed by hand signal to let this car go, dive
    %!warning: only works if this step is done after update
    %of cars in front of crossraod!
    if ( plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 && ...
```

```
            ( ( tp_next(3,1) == 1 && plocal_next(6,4) == 1 && ...
            plocal(6,4) == 1 ) || ( deadlocklocal == 4 && unlock == 2 ) ) )
        plocal_next(5,2) = plocal(4,1);
        pspeedlocal_next(5,2) = 1;
        camelocal_next(5,2) = camelocal(4,1);
        %no deadlock, clear deadlock counter
        deadlocklocal_next = 0;
    % if not, stay
    else
        plocal_next(4,1) = plocal(4,1);
        pspeedlocal_next(4,1) = 0;
        camelocal_next(4,1) = camelocal(4,1);
        %increase deadlock counter, if it reaches 4 a
        %deadlock occurs and will have to be solve in next
        %time step by a hand signals between drivers
        deadlocklocal_next = deadlocklocal_next + 1;
    end
end

%car coming form below, going stright ahead
%1. step
if ( plocal(6,4) == 0.4 )
    %if space is free and there are no are coming from the
    %right or is there has been a deadlock and driver have
    %agreed by hand signal to let this car go, dive
    %!warning: only works if this step is done after update
    %of cars in front of crossraod!
    if ( plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 && ...
            ( ( tp_next(4,1) == 1 && plocal_next(3,6) == 1 && ...
            plocal(3,6) == 1 ) || ( deadlocklocal == 4 && unlock == 3 ) ) )
        plocal_next(5,5) = plocal(6,4);
        pspeedlocal_next(5,5) = 1;
        camelocal_next(5,5) = camelocal(6,4);
        %no deadlock, clear deadlock counter
        deadlocklocal_next = 0;
    % if not, stay
    else
        plocal_next(6,4) = plocal(6,4);
        pspeedlocal_next(6,4) = 0;
        camelocal_next(6,4) = camelocal(6,4);
        %increase deadlock counter, if it reaches 4 a
        %deadlock occurs and will have to be solve in next
        %time step by a hand signals between drivers
        deadlocklocal_next = deadlocklocal_next + 1;
    end
end

%car coming form right, going stright ahead
%1. step
if ( plocal(3,6) == 0.4 )
```

```
    %if space is free and there are no are coming from the
    %right or is there has been a deadlock and driver have
    %agreed by hand signal to let this car go, dive
    %!warning: only works if this step is done after update
    %of cars in front of crossraod!
    if ( plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 && ...
            ( ( tp_next(1,1) == 1 && plocal_next(1,3) == 1 && ...
            plocal(1,3) == 1 ) || ( deadlocklocal == 4 && unlock == 4 ) ) )
        plocal_next(2,5) = plocal(3,6);
        pspeedlocal_next(2,5) = 1;
        camelocal_next(2,5) = camelocal(3,6);
        %no deadlock, clear deadlock counter
        deadlocklocal_next = 0;
    % if not, stay
    else
        plocal_next(3,6) = plocal(3,6);
        pspeedlocal_next(3,6) = 0;
        camelocal_next(3,6) = camelocal(3,6);
        %increase deadlock counter, if it reaches 4 a
        %deadlock occurs and will have to be solve in next
        %time step by a hand signals between drivers
        deadlocklocal_next = deadlocklocal_next + 1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars turning right step 2
%cars going straight ahead step 5

%2. step for car coming from above, turning right
%5. step for car coming from right, going straight ahead
if ( plocal(2,2) == 0.7 || ( plocal(2,2) == 0.4 && camelocal(2,2) == 4 ) )
    %if space free, car has right of way and can drive
    if ( plocal(3,1) == 1 )
        plocal_next(3,1) = plocal(2,2);
        pspeedlocal_next(3,1) = 1;
        camelocal_next(3,1) = camelocal(2,2);
        % if space not free, stay
    else
        plocal_next(2,2) = plocal(2,2);
        pspeedlocal_next(2,2) = 0;
        camelocal_next(2,2) = camelocal(2,2);
    end
end

%2. step for car coming from left, turning right
%5. step for car coming from above, going straight ahead
if ( plocal(5,2) == 0.7 || ( plocal(5,2) == 0.4 && camelocal(5,2) == 1 ) )
    %if space free, car has right of way and can drive
    if ( plocal(6,3) == 1 )
```

```
        plocal_next(6,3) = plocal(5,2);
        pspeedlocal_next(6,3) = 1;
        camelocal_next(6,3) = camelocal(5,2);
    % if space not free, stay
    else
        plocal_next(5,2) = plocal(5,2);
        pspeedlocal_next(5,2) = 0;
        camelocal_next(5,2) = camelocal(5,2);
    end
end

%2. step for car coming from below, turning right
%5. step for car coming from left, going straight ahead
if ( plocal(5,5) == 0.7 || ( plocal(5,5) == 0.4 && camelocal(5,5) == 2 ) )
    %if space free, car has right of way and can drive
    if ( plocal(4,6) == 1 )
        plocal_next(4,6) = plocal(5,5);
        pspeedlocal_next(4,6) = 1;
        camelocal_next(4,6) = camelocal(5,5);
    % if space not free, stay
    else
        plocal_next(5,5) = plocal(5,5);
        pspeedlocal_next(5,5) = 0;
        camelocal_next(5,5) = camelocal(5,5);
    end
end

%2. step for car coming from right, turning right
%5. step for car coming from below, going straight ahead
if ( plocal(2,5) == 0.7 || ( plocal(2,5) == 0.4 && camelocal(2,5) == 3) )
    %if space free, car has right of way and can drive
    if ( plocal(1,4) == 1 )
        plocal_next(1,4) = plocal(2,5);
        pspeedlocal_next(1,4) = 1;
        camelocal_next(1,4) = camelocal(2,5);
    % if space not free, stay
    else
        plocal_next(2,5) = plocal(2,5);
        pspeedlocal_next(2,5) = 0;
        camelocal_next(2,5) = camelocal(2,5);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars going staight ahead step 2 to 4

%car coming form above, going staight ahead
%2. step
if ( plocal(2,2) == 0.4 && camelocal(2,2) == 1 )
    %if space is free, drive
```

```
    if ( plocal(3,2) == 1 )
        plocal_next(3,2) = plocal(2,2);
        pspeedlocal_next(3,2) = 1;
        camelocal_next(3,2) = camelocal(2,2);
    % if not, wait
    else
        plocal_next(2,2) = plocal(2,2);
        pspeedlocal_next(2,2) = 0;
        camelocal_next(2,2) = camelocal(2,2);
    end
end
%3. step
if ( plocal(3,2) == 0.4 )
    %if space is free, drive
    if ( plocal(4,2) == 1 && plocal(4,1) ≠ 0.1 )
        plocal_next(4,2) = plocal(3,2);
        pspeedlocal_next(4,2) = 1;
        camelocal_next(4,2) = camelocal(3,2);
    % if not, wait
    else
        plocal_next(3,2) = plocal(3,2);
        pspeedlocal_next(3,2) = 0;
        camelocal_next(3,2) =camelocal(3,2);
    end
end
%4. step
if ( plocal(4,2) == 0.4 )
    %if space is free, drive
    if ( plocal(5,2) == 1 )
        plocal_next(5,2) = plocal(4,2);
        pspeedlocal_next(5,2) = 1;
        camelocal_next(5,2) = camelocal(4,2);
    % if not, wait
    else
        plocal_next(4,2) = plocal(4,2);
        pspeedlocal_next(4,2) = 0;
        camelocal_next(4,2) = camelocal(4,2);
    end
end

%car coming form left, going staight ahead
%2. step
if ( plocal(5,2) == 0.4 && camelocal(5,2) == 2 )
    %if space is free, drive
    if ( plocal(5,3) == 1 )
        plocal_next(5,3) = plocal(5,2);
        pspeedlocal_next(5,3) = 1;
        camelocal_next(5,3) = camelocal(5,2);
    % if not, wait
    else
```

```
        plocal_next(5,2) = plocal(5,2);
        pspeedlocal_next(5,2) = 0;
        camelocal_next(5,2) = camelocal(5,2);
    end
end
%3. step
if ( plocal(5,3) == 0.4 )
    %if space is free, drive
    if ( plocal(5,4) == 1 && plocal(6,4) ≠ 0.1 )
        plocal_next(5,4) = plocal(5,3);
        pspeedlocal_next(5,4) = 1;
        camelocal_next(5,4) = camelocal(5,3);
    % if not, wait
    else
        plocal_next(5,3) = plocal(5,3);
        pspeedlocal_next(5,3) = 0;
        camelocal_next(5,3) =camelocal(5,3);
    end
end
%4. step
if ( plocal(5,4) == 0.4 )
    %if space is free, drive
    if ( plocal(5,5) == 1 )
        plocal_next(5,5) = plocal(5,4);
        pspeedlocal_next(5,5) = 1;
        camelocal_next(5,5) = camelocal(5,4);
    % if not, wait
    else
        plocal_next(5,4) = plocal(5,4);
        pspeedlocal_next(5,4) = 0;
        camelocal_next(5,4) = camelocal(5,4);
    end
end

%car coming form below, going staight ahead
%2. step
if ( plocal(5,5) == 0.4 && camelocal(5,5) == 3 )
    %if space is free, drive
    if ( plocal(4,5) == 1 )
        plocal_next(4,5) = plocal(5,5);
        pspeedlocal_next(4,5) = 1;
        camelocal_next(4,5) = camelocal(5,5);
    % if not, wait
    else
        plocal_next(5,5) = plocal(5,5);
        pspeedlocal_next(5,5) = 0;
        camelocal_next(5,5) = camelocal(5,5);
    end
end
%3. step
```

```
if ( plocal(4,5) == 0.4 )
    %if space is free, drive
    if ( plocal(3,5) == 1 && plocal(3,6) ≠ 0.1 )
        plocal_next(3,5) = plocal(4,5);
        pspeedlocal_next(3,5) = 1;
        camelocal_next(3,5) = camelocal(4,5);
    % if not, wait
    else
        plocal_next(4,5) = plocal(4,5);
        pspeedlocal_next(4,5) = 0;
        camelocal_next(4,5) =camelocal(4,5);
    end
end
%4. step
if ( plocal(3,5) == 0.4 )
    %if space is free, drive
    if ( plocal(2,5) == 1 )
        plocal_next(2,5) = plocal(3,5);
        pspeedlocal_next(2,5) = 1;
        camelocal_next(2,5) = camelocal(3,5);
    % if not, wait
    else
        plocal_next(3,5) = plocal(3,5);
        pspeedlocal_next(3,5) = 0;
        camelocal_next(3,5) = camelocal(3,5);
    end
end

%car coming form right, going staight ahead
%2. step
if ( plocal(2,5) == 0.4 && camelocal(2,5) == 4 )
    %if space is free, drive
    if ( plocal(2,4) == 1 )
        plocal_next(2,4) = plocal(2,5);
        pspeedlocal_next(2,4) = 1;
        camelocal_next(2,4) = camelocal(2,5);
    % if not, wait
    else
        plocal_next(2,5) = plocal(2,5);
        pspeedlocal_next(2,5) = 0;
        camelocal_next(2,5) = camelocal(2,5);
    end
end
%3. step
if ( plocal(2,4) == 0.4 )
    %if space is free, drive
    if ( plocal(2,3) == 1 && plocal(1,3) ≠ 0.1 )
        plocal_next(2,3) = plocal(2,4);
        pspeedlocal_next(2,3) = 1;
        camelocal_next(2,3) = camelocal(2,4);
```

```
    % if not, wait
    else
        plocal_next(2,4) = plocal(2,4);
        pspeedlocal_next(2,4) = 0;
        camelocal_next(2,4) =camelocal(2,4);
    end
end
%4. step
if ( plocal(2,3) == 0.4 )
    %if space is free, drive
    if ( plocal(2,2) == 1 )
        plocal_next(2,2) = plocal(2,3);
        pspeedlocal_next(2,2) = 1;
        camelocal_next(2,2) = camelocal(2,3);
    % if not, wait
    else
        plocal_next(2,3) = plocal(2,3);
        pspeedlocal_next(2,3) = 0;
        camelocal_next(2,3) = camelocal(2,3);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars turning left

%car coming from above turning left
%1. step
if ( plocal(1,3) == 0.1 )
    %if next two spaces are free and there is no car coming
    %form right turning in front of this car, drive
    if ( plocal(2,3) == 1 && plocal(3,4) == 1 && ...
            plocal(4,2) ≠ 0.1 && plocal(3,3) == 1 )
        plocal_next(2,3) = plocal(1,3);
        pspeedlocal_next(2,3) = 1;
        camelocal_next(2,3) = camelocal(1,3);
    %if not, stay
    else
        plocal_next(1,3) = plocal(1,3);
        pspeedlocal_next(1,3) = 0;
        camelocal_next(1,3) = camelocal(1,3);
    end
end
%2. step
if ( plocal(2,3) == 0.1 )
    %is space is free, drive
    if ( plocal(3,4) == 1 )
        plocal_next(3,4) = plocal(2,3);
        pspeedlocal_next(3,4) = 1;
        camelocal_next(3,4) = camelocal(2,3);
    %if not, stay
```

```
    else
        plocal_next(2,3) = plocal(2,3);
        pspeedlocal_next(2,3) = 0;
        camelocal_next(2,3) = camelocal(2,3);
    end
end
%3 .step
if ( plocal(3,4) == 0.1 )
    %if space is free and there is no car coming from the
    %opposite side going straight ahead and no car coming
    %from the right , drive
    if ( plocal(4,5) == 1 && plocal(4,6) == 1 && ...
            plocal(5,5) == 1 && plocal(5,4) ≠ 0.4 )
        plocal_next(4,5) = plocal(3,4);
        pspeedlocal_next(4,5) = 1;
        camelocal_next(4,5) = camelocal(3,4);
    %if not, stay
    else
        plocal_next(3,4) = plocal(3,4);
        pspeedlocal_next(3,4) = 0;
        camelocal_next(3,4) = camelocal(3,4);
    end
end
%4. step
if ( plocal(4,5) == 0.1 )
    %if space is free, drive
    if ( plocal(4,6) == 1 && plocal(5,5) ≠ 0.7 && ...
            ¬( plocal(5,5) == 0.4 && camelocal(5,5) == 2 ) )
        plocal_next(4,6) = plocal(4,5);
        pspeedlocal_next(4,6) = 1;
        camelocal_next(4,6) = camelocal(4,5);
    %if not, stay
    else
        plocal_next(4,5) = plocal(4,5);
        pspeedlocal_next(4,5) = 0;
        camelocal_next(4,5) = camelocal(4,5);
    end
end

%car coming from the left turning left
%1. step
if ( plocal(4,1) == 0.1 )
    %if next two spaces are free and there is no car coming
    %form right turning in front of this car, drive
    if ( plocal(4,2) == 1 && plocal(3,3) == 1 && ...
            plocal(5,4) ≠ 0.1 && plocal(4,3) == 1 )
        plocal_next(4,2) = plocal(4,1);
        pspeedlocal_next(4,2) = 1;
        camelocal_next(4,2) = camelocal(4,1);
    %if not, stay
```

```
    else
        plocal_next(4,1) = plocal(4,1);
        pspeedlocal_next(4,1) = 0;
        camelocal_next(4,1) = camelocal(4,1);
    end
end
%2. step
if ( plocal(4,2) == 0.1 )
    %is space is free, drive
    if ( plocal(3,3) == 1 )
        plocal_next(3,3) = plocal(4,2);
        pspeedlocal_next(3,3) = 1;
        camelocal_next(3,3) = camelocal(4,2);
    %if not, stay
    else
        plocal_next(4,2) = plocal(4,2);
        pspeedlocal_next(4,2) = 0;
        camelocal_next(4,2) = camelocal(4,2);
    end
end
%3 .step
if ( plocal(3,3) == 0.1 )
    %if space is free and there is no car coming from the
    %opposite side going straight ahead and no car coming
    %from the right , drive
    if ( plocal(2,4) == 1 && plocal(1,4) == 1 && ...
            plocal(2,5) == 1 && plocal(3,5) ≠ 0.4 )
        plocal_next(2,4) = plocal(3,3);
        pspeedlocal_next(2,4) = 1;
        camelocal_next(2,4) = camelocal(3,3);
    %if not, stay
    else
        plocal_next(3,3) = plocal(3,3);
        pspeedlocal_next(3,3) = 0;
        camelocal_next(3,3) = camelocal(3,3);
    end
end
%4. step
if ( plocal(2,4) == 0.1 )
    %if space is free, drive
    if ( plocal(1,4) == 1 && plocal(2,5) ≠ 0.7 && ...
            ¬( plocal(2,5) == 0.4 && camelocal(2,5) == 3 ) )
        plocal_next(1,4) = plocal(2,4);
        pspeedlocal_next(1,4) = 1;
        camelocal_next(1,4) = camelocal(2,4);
    %if not, stay
    else
        plocal_next(2,4) = plocal(2,4);
        pspeedlocal_next(2,4) = 0;
        camelocal_next(2,4) = camelocal(2,4);
```

```
    end
end

%car coming from below turning left
%1. step
if ( plocal(6,4) == 0.1 )
    %if next two spaces are free and there is no car coming
    %form right turning in front of this car, drive
    if ( plocal(5,4) == 1 && plocal(4,3) == 1 && ...
            plocal(3,5) ~= 0.1 && plocal(4,4) == 1 )
        plocal_next(5,4) = plocal(6,4);
        pspeedlocal_next(5,4) = 1;
        camelocal_next(5,4) = camelocal(6,4);
    %if not, stay
    else
        plocal_next(6,4) = plocal(6,4);
        pspeedlocal_next(6,4) = 1;
        camelocal_next(6,4) = camelocal(6,4);
    end
end
%2. step
if ( plocal(5,4) == 0.1 )
    %is space is free, drive
    if ( plocal(4,3) == 1 )
        plocal_next(4,3) = plocal(5,4);
        pspeedlocal_next(4,3) = 1;
        camelocal_next(4,3) = camelocal(5,4);
    %if not, stay
    else
        plocal_next(5,4) = plocal(5,4);
        pspeedlocal_next(5,4) = 0;
        camelocal_next(5,4) = camelocal(5,4);
    end
end
%3 .step
if ( plocal(4,3) == 0.1 )
    %if space is free and there is no car coming from the
    %opposite side going straight ahead and no car coming
    %from the right , drive
    if ( plocal(3,2) == 1 && plocal(3,1) == 1 && ...
            plocal(2,2) == 1 && plocal(2,3) ~= 0.4 )
        plocal_next(3,2) = plocal(4,3);
        pspeedlocal_next(3,2) = 1;
        camelocal_next(3,2) = camelocal(4,3);
    %if not, stay
    else
        plocal_next(4,3) = plocal(4,3);
        pspeedlocal_next(4,3) = 0;
        camelocal_next(4,3) = camelocal(4,3);
    end
```

```
end
%4. step
if ( plocal(3,2) == 0.1 )
    %if space is free, drive
    if ( plocal(3,1) == 1 && plocal(2,2) ≠ 0.7 && ...
            ¬( plocal(2,2) == 0.4 && camelocal(2,2) == 4 ) )
        plocal_next(3,1) = plocal(3,2);
        pspeedlocal_next(3,1) = 1;
        camelocal_next(3,1) = camelocal(3,2);
    %if not, stay
    else
        plocal_next(3,2) = plocal(3,2);
        pspeedlocal_next(3,2) = 0;
        camelocal_next(3,2) = camelocal(3,2);
    end
end

%car coming from right turning left
%1. step
if ( plocal(3,6) == 0.1 )
    %if next two spaces are free and there is no car coming
    %form right turning in front of this car, drive
    if ( plocal(3,5) == 1 && plocal(4,4) == 1 && ...
            plocal(2,3) ≠ 0.1 && plocal(3,4) == 1 )
        plocal_next(3,5) = plocal(3,6);
        pspeedlocal_next(3,5) = 1;
        camelocal_next(3,5) = camelocal(3,6);
    %if not, stay
    else
        plocal_next(3,6) = plocal(3,6);
        pspeedlocal_next(3,6) = 0;
        camelocal_next(3,6) = camelocal(3,6);
    end
end
%2. step
if ( plocal(3,5) == 0.1 )
    %is space is free, drive
    if ( plocal(4,4) == 1 )
        plocal_next(4,4) = plocal(3,5);
        pspeedlocal_next(4,4) = 1;
        camelocal_next(4,4) = camelocal(3,5);
    %if not, stay
    else
        plocal_next(3,5) = plocal(3,5);
        pspeedlocal_next(3,5) = 0;
        camelocal_next(3,5) = camelocal(3,5);
    end
end
%3 .step
if ( plocal(4,4) == 0.1 )
```

```
    %if space is free and there is no car coming from the
    %opposite side going straight ahead and no car coming
    %from the right , drive
    if ( plocal(5,3) == 1 && plocal(6,3) == 1 && ...
            plocal(5,2) == 1 && plocal(4,2) ≠ 0.4 )
        plocal_next(5,3) = plocal(4,4);
        pspeedlocal_next(5,3) = 1;
        camelocal_next(5,3) = camelocal(4,4);
    %if not, stay
    else
        plocal_next(4,4) = plocal(4,4);
        pspeedlocal_next(4,4) = 0;
        camelocal_next(4,4) = camelocal(4,4);
    end
end
%4. step
if ( plocal(5,3) == 0.1 )
    %if space is free, drive
    if ( plocal(6,3) == 1 && plocal(5,2) ≠ 0.7 && ...
            ¬( plocal(5,2) == 0.4 && camelocal(5,2) == 1 ) )
        plocal_next(6,3) = plocal(5,3);
        pspeedlocal_next(6,3) = 1;
        camelocal_next(6,3) = camelocal(5,3);
    %if not, stay
    else
        plocal_next(5,3) = plocal(5,3);
        pspeedlocal_next(5,3) = 1;
        camelocal_next(5,3) = camelocal(5,3);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cars leaving crossing

%car leaving to the top
if ( plocal(1,4) ≠ 1 )
    %if space free, leave crossing with speed 1
    if ( fp(1,1) == 1 )
        fp_next(1,1) = 0.4;
        fpspeed_next(1,1) = 1;
    %if space not free, stay
    else
        plocal_next(1,4) = plocal(1,4);
        pspeedlocal_next(1,4) = 0;
        camelocal_next(1,4) = camelocal(1,4);
    end
end

%car leaving to the left
if ( plocal(3,1) ≠ 1 )
```

```
    %if space free, leave crossing with speed 1
    if ( fp(2,1) == 1 )
        fp_next(2,1) = 0.4;
        fpspeed_next(2,1) = 1;
    %if space not free, stay
    else
        plocal_next(3,1) = plocal(3,1);
        pspeedlocal_next(3,1) = 0;
        camelocal_next(3,1) = camelocal(3,1);
    end
end

%car leaving to the bottom
if ( plocal(6,3) ≠ 1 )
    %if space free, leave crossing with speed 1
    if ( fp(3,1) == 1 )
        fp_next(3,1) = 0.4;
        fpspeed_next(3,1) = 1;
    %if space not free, stay
    else
        plocal_next(6,3) = plocal(6,3);
        pspeedlocal_next(6,3) = 0;
        camelocal_next(6,3) = camelocal(6,3);
    end
end

%car leaving to the bottom
if ( plocal(4,6) ≠ 1 )
    %if space free, leave crossing with speed 1
    if ( fp(4,1) == 1 )
        fp_next(4,1) = 0.4;
        fpspeed_next(4,1) = 1;
    %if space not free, stay
    else
        plocal_next(4,6) = plocal(4,6);
        pspeedlocal_next(4,6) = 0;
        camelocal_next(4,6) = camelocal(4,6);
    end
end

end
```

7.2.5 connection.m

```
function [cNew,dNew] = connection(aOld,bOld,cOld,posNew,m,n,length)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CONNECTION Deside to which street a certain street connects to
%
%INPUT:
%AOLD column index of intersection
%BOLD, row index of intersection
%COLD, column index in t of old position
%posNEW, position in new street
%M, number of columns in city map
%N, number of rows in city map
%LENGTH, Length of a street
%
%OUTPUT:
%CNEW, Column index in t of new position
%DNEW, Row index in t of new position
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%street heading up from intersection
if ( mod(cOld,4) == 1 )
    %if there is a intersections above, connect to it
    if ( aOld > 1)
        cNew = (aOld - 2) * 4 + 3;
        dNew = (bOld - 1) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = (m - 1) * 4 + 3;
        dNew = (bOld - 1) * length + posNew;
    end
end

%street heading left from intersection
if ( mod(cOld,4) == 2 )
    %if there is a intersection to the left, connect to it
    if ( bOld > 1 )
        cNew = aOld * 4;
        dNew = (bOld - 2) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = aOld * 4;
        dNew = (n - 1) * length + posNew;
    end
```

```
end

%street heading down from intersection
if ( mod(cOld,4) == 3 )
    %if there is a intersection below, connect to it
    if ( aOld < m )
        cNew = aOld * 4 + 1;
        dNew = (bOld - 1) * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = 1;
        dNew = (bOld - 1) * length + posNew;
    end
end

%street heading right from intersection
if ( mod(cOld,4) == 0 )
    %if there is a intersection to the right, connect to it
    if ( bOld < n )
        cNew = (aOld - 1) * 4 + 2;
        dNew = bOld * length + posNew;
    %otherwise connect to other side of map
    else
        cNew = (aOld - 1) * 4 + 2;
        dNew = posNew;
    end
end
```

7.2.6 pdestination.m

```
function [pfirst] = pdestination
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PDESTINATION Deside where a car is going
%
%OUTPUT:
%PFIRST = 0.1 car turns right
%       = 0.4 car goes straight ahead
%       = 0.7 car turns left
%
%A project by Bastian Buecheler and Tony Wood in the GeSS course "Modelling
%and Simulation of Social Systems with MATLAB" at ETH Zurich.
%Spring 2010
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%decide which direction car is going
u = randi(12,1);
%probabilty 6/12 car goes straight ahead
if ( u ≤ 6 )
    pfirst = 0.4;
end
%probabilty 3/12 car turns right
if ( u ≥ 7 && u ≤ 9 )
    %indicate right
    pfirst = 0.7;
end
%probabilty 3/12 car turns left
if ( u ≥ 10 && u ≤ 12 )
    pfirst = 0.1;
end

end
```