



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Project Report

**Modelling the phenomenon
of congestion at Gotthard**

A case study

Eric Hayoz, Janick Zwysig

Zurich
December 2014

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.

Eric Hayoz



Janick Zwyssig





Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

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

I hereby declare that the written work I have submitted entitled

Modelling the phenomenon of congestion at Gotthard - A case study

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

Author(s)

Last name

Hayoz

Zwyssig

First name

Eric

Janick

Supervising lecturer

Last name

Kuhn

Woolley

First name

Tobias

Olivia

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

The above written work may be tested electronically for plagiarism.

Zurich, 12. December 2014

Place and date

Eric Hayoz

Signature

Janick Zwyssig

Contents

1	Abstract	5
2	Individual contributions	5
3	Introduction	6
3.1	Motivation	6
4	Description of the Model	6
4.1	The Nagel-Schreckenberg model	6
4.2	Example of an iteration	7
4.3	Situation	8
5	Implementation	9
5.1	Correlation between reality and model	9
5.2	Inflow	10
5.3	Average speed	10
5.4	Lane Change and red-light	10
5.5	Congestion Length	11
5.6	Processing the datasets delivered by ASTRA	11
6	Simulation Results and Discussion	12
6.1	Preparing dataset – curve fitting	12
6.2	Train model	14
7	Summary and Outlook	16
7.1	Outlook	16
8	Appendix	16

1 Abstract

Abstract of Eric Hayoz & Janick Zwysig, Modelling the phenomenon of congestion at Gotthard – A case study, 2014:

This work describes the modelling of congestion on the stage between Erstfeld and the Gotthard tunnel in Göschenen, in the canton of Uri. In a first step a simulation based on the Nagel-Schreckenberg model was created. The model was supplemented by a second lane, a red-light and the ability to change the lane. A mapping from real data to the model allows to compare values. The main goal is to train the model that one can compare the measured congestion to a real traffic jam. For that the authors have implemented a congestion measurement and have processed datasets, provided by ASTRA (Amt für Strassen). A parameter setting that fits the reference data best was found.

Finally the trained model was feed by another, unused dataset to compare the results. Unfortunately, due to abnormalities of the datasets and some restrictions of the model the found data

2 Individual contributions

We divided the code work as follows:

Eric Hayoz:

- Implementation NaSch-Modell for 2 cars
- Handling Dataset with inFlow and Speed
- Correlation Reality - Model
- Creating optiFinder

Janick Zwysig:

- Implementation NaSch-Modell for n cars
- Implementation laneChange & redLight
- curve fitting for results
- congestion measurement

In the end both authors contributed to every section. The article was written together.

3 Introduction

Nowadays, traffic jams belong to every day life. Because Switzerland is an important transit way to Southern Europe for traffic, the Gotthard tunnel is an unavoidable bottleneck. The tunnel stretches over 15 Kilometers from Göschenen, Kanton Uri to Airolo in the canton of Ticino. Over and over again it comes to long traffic jams, at peak times up to 20 Kilometers. Responsible for that are on the one hand holidays in Switzerland or neighbor countries and on the other hand the lane reduction from two to one lane. Switzerland has been discussing this issue for many years and the construction of a second tunnel was proposed. The purpose of the second tunnel should be built for security reasons and not in order to increase the traffic flow. This is due to an election 20 years ago, called the Alpenschutzinitiative. In the near future there will be a restauration of the tunnel. That's why the discussion about a second tunnel will commence once again.

The authors have the goal to investigate the Gotthard tunnel on the north side in Göschenen. That's why a model has been created. By means of the number of cars passing through Erstfeld (a village 19 Kilometers from the tunnel), the length of congestion can be measured. The Nagel-Schreckenberg model will be employed for the simulation.

3.1 Motivation

As one of the authors is born in the canton of Uri he knows about the traffic issues at the Gotthard tunnel. This explains the personal motivation to investigate the phenomenon of traffic congestions. The other author is often on the way by car, he has already wasted a lot of time in traffic congestions and thus has also have burning interest in efficient solutions to handle them.

4 Description of the Model

4.1 The Nagel-Schreckenberg model

This model was created in the 90s by the German physicists Kai Nagel and Michael Schreckenberg to simulate traffic flow on highways. It explained the so called *Phantom-Congestion* which occurs when drivers linger by not accelerating even if they have the opportunity to do so.

The model is a cellular automata based on four very simple rules, which are applied in sequence. One complete loop is called an iteration.

1. Acceleration: Each vehicle, that has not yet reached the maximum speed (which is 5), increases its speed by one unit.
2. Deceleration: If the size of the gap (in cells) between two vehicles is less than the speed of the following car (in speed units), this cars speed is reduced to a value equivalent to the size of the gap.
3. Linging: Every vehicle reduces its speed by 1 with a probability p ($0 \leq p \leq 1$).
4. Moving forward: Every vehicle moves forward the number of cells equal to its speed.

4.2 Example of an iteration




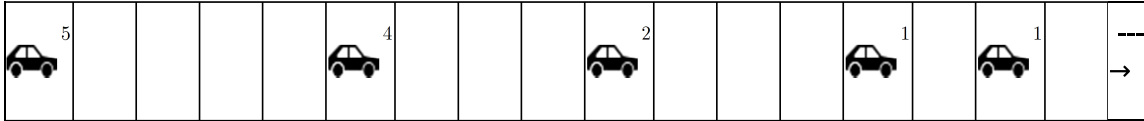
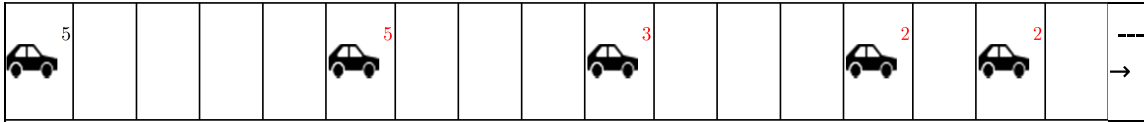
Symbol	Meaning
	A vehicle with a speed of 1 (27 km/h)
	A vehicle that has accelerated or decelerated to a speed of 3 (81 km/h)
	A vehicle has moved with a speed of 2 (going 2 cells forward)

Figure 1: Nagel-Schreckenberg model – Symbols

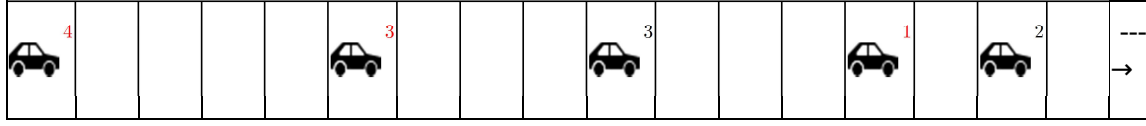
Random startup setting for time t



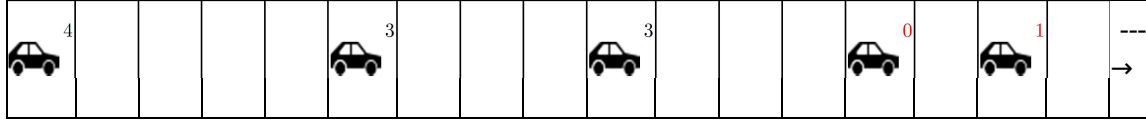
Step (1) – Acceleration, $v_{max} = 5$



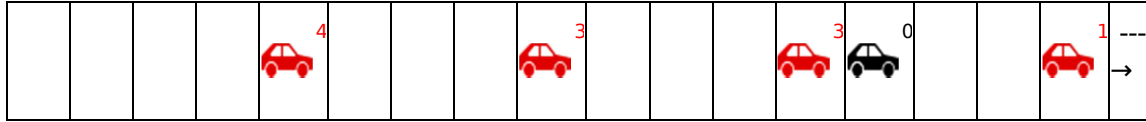
Step (2) – Deceleration



Step (3) – Linging ($\rho = 1/3$, affecting the two leading cars)



Step (4) – Moving (Setting for the time $t + 1$)



4.3 Situation

The distance between Erstfeld and the Gotthard tunnel is relevant for the model. The highway to the tunnel is two-laned and will be reduced to one lane in front of the tunnel. As on the figure 2 shown, the authors resigned to depicted any exits. The reason for this is first to reduce complexity of the model and second to neglect irrelevant factors. In case of congestion there is a red light that regulates the traffic flow, which is also taken into account in the model.

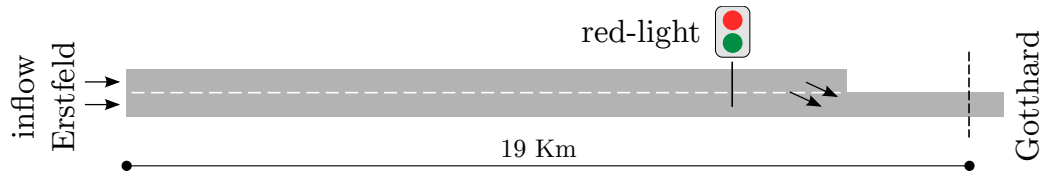


Figure 2: Sketch of model

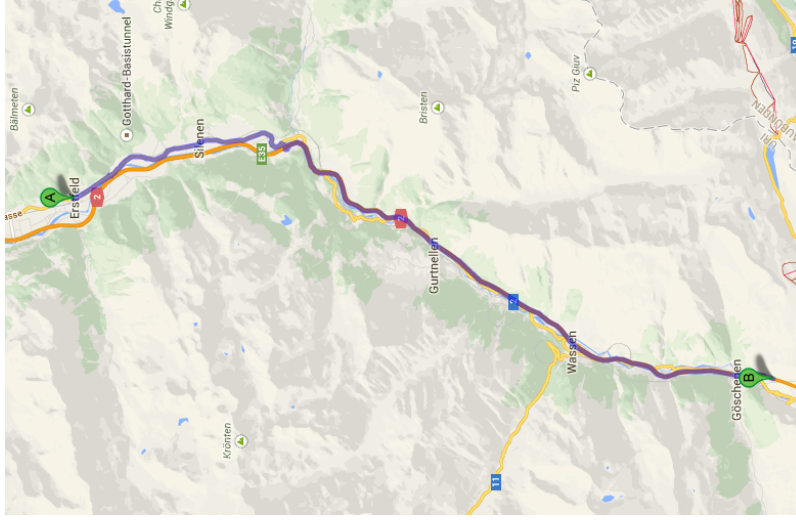


Figure 3: map of Uri, A: Erstfeld, B: Gotthard tunnel at Göschenen

5 Implementation

5.1 Correlation between reality and model

Item	Reality	Model
Time	1s	1 iteration
Maximum speed	120 km/h (we defined v_{max} as 100 km/h to get even values)	5 (only 6 levels)
Length of an average car (Skoda Octavia)	4.5 m	1 cell
Distance Erstfeld - Göschenen	19'000 m	4222 cells (19'000 m / 4.5 m, rounded)

Figure 4: Units

Variable name	Meaning	Values
moveProb	Probability for a car to move forward	0...1
moveCorr	Modify moveProb for a given time (in order to improve congestion length prediction)	-1...1
laneChange	Probability for a car to change its lane	0...1
-	Value of laneChange as a function of distance to changeCell (for locations where cars have to change from left to right)	Constants in equation
redLight	Illuminate a dropcounter redlight in front of tunnel (it turns on if congestion length > 2 km and redLight is set to 1)	0, 1
dropCounter	Length of a period when redlight is ON	0...Inf
-	Maximum speed just in front of tunnel	0...5

Figure 5: Variables

5.2 Inflow

The inflow is given in $\#cars / 180\text{ s}$. For every iteration in our model, the cars are spawned in Erstfeld (starting position) on the two lanes with a total probability $p = \#cars/180$. As there are normally slightly more vehicles on the right lane, we multiply p for the right lane with a factor of 1.1 and the p for the left lane with a factor of 0.9.

5.3 Average speed

First we convert the average speed from km/h (0...120) to “Nagel-Schreckenberg-Speed” (0...5). If the speed is more than 5, we round down to 5. Example: Average Speed is 94 km/h. Converting into Nagel-Schreckenberg-Speed results in 4.7. Our model would in this case spawn 70% of the cars with a speed of 5, and 30% with a speed of 4.

5.4 Lane Change and red-light

As cars change lane often in reality, it should be also a part of the model. Furthermore, it is necessary to change the lane due to lane reduction in front of the tunnel. The probability to change the lane from left to the right is higher than the other way around. That yields more cars on the right lane. In front of the tunnel the probability for a lane change increases, due to lane reduction.

Red-light is activated if the congestion length exceed the length of one Kilometre. It’s acting approximately 40 meters in front of the lane reduction. Every second one car can pass through the tunnel.

5.5 Congestion Length

The length of a congestion is given in Kilometres. First we calculate the length of the congestion in our simulation and then multiply the length (unit: cells) by the length of a cell (4.5 m) to enable a comparison between the virtual (calculated) congestion with the real (measured) congestion.

Measuring the congestion length in our model: To obtain a stable output, we divide the highway into blocks with a length of 50 cells. If a block fulfils the two following criteria, it is declared a congestion block: high density and low average speed. We count the congestion blocks starting in Göschenen heading northbound to Erstfeld, and we only accept one uncongested block between any two blocks, otherwise we consider the congestion as terminated at the point where two or more blocks do not fulfil the congestion criteria anymore.

5.6 Processing the datasets delivered by ASTRA

The attached extract shows among other things that we have information about the inflow values and average speeds in 3 minutes intervals. We also have the length of the congestion at specific times (random times and course measurements). These are the three values that we used.

Datum	Fahrzeuge pro 3Min	Durchschnittsgeschwindigkeit	Fahrzeuge pro 3Min Vergleichstage Fr 25.7.2014 - Sa 26.7.2014	Durchschnittsgeschwindigkeit Vergleichstage Fr 25.7.2014 - Sa 26.7.2014	Fahrzeuge pro 3Min Vergleichstage Fr 1.8.2014 - Sa 2.8.2014	Durchschnittsgeschwindigkeit Vergleichstage Fr 1.8.2014 - Sa 2.8.2014
2014-07-18 00:00-00:03	30	114	22	110	33	110
2014-07-18 00:03-00:06	22	116	15	98	37	107
2014-07-18 00:06-00:09	22	115	25	116	42	115

Figure 6: Inflow values with average speeds

Datum	Uhrzeit	zwischen		und	AS	Name	Stau- km	Warte- zeit	Verkehrs-information	Empfehlung
		AS	Name							
02.08.2014	05:10	39	Wassen	40	Göschenen				stockender Verkehr	keine
02.08.2014	05:30	39	Wassen	40	Göschenen	2	20		Stau, Verkehrsüberlastung	keine
02.08.2014	06:11	39	Wassen	40	Göschenen	3	30		Stau, Verkehrsüberlastung, Einfahrt Gö gesperrt.	keine

Figure 7: Congestion data

6 Simulation Results and Discussion

The reference data for congestion measurement is provided by ASTRA (Bundesamt für Strassen). There are four datasets:

- Dataset 1: measurement of 2014 July, 18. - 19.
- Dataset 2: measurement of 2014 August, first
- Dataset 3: measurement of 2014 July, 25. - 26.
- Dataset 4: measurement of 2014 August, second

In a first step the model will be trained by dataset 1 and 2. The authors wrote a program which calculates specified values of a specified variable (moveProb and moveCorr) for a given number of times. With other words, it finds the optimal parameter values that minimize the error of measured congestion. In a second step, the datasets 3 and 4 are used to evaluate the previous trained model.

6.1 Preparing dataset – curve fitting

The timestamps for the measurement are irregular. In order to compare the congestion from the model to the reference data, the breakpoints will be interpolated. The resulting curve can be used to generate regular timestamps for measurement. The authors have opted for a linear interpolation. That means that every point will be connected by a straight line with its neighbour, see on the figures below. A linear interpolation is reasonable because the congestion length between two breakpoints cannot change very quickly.

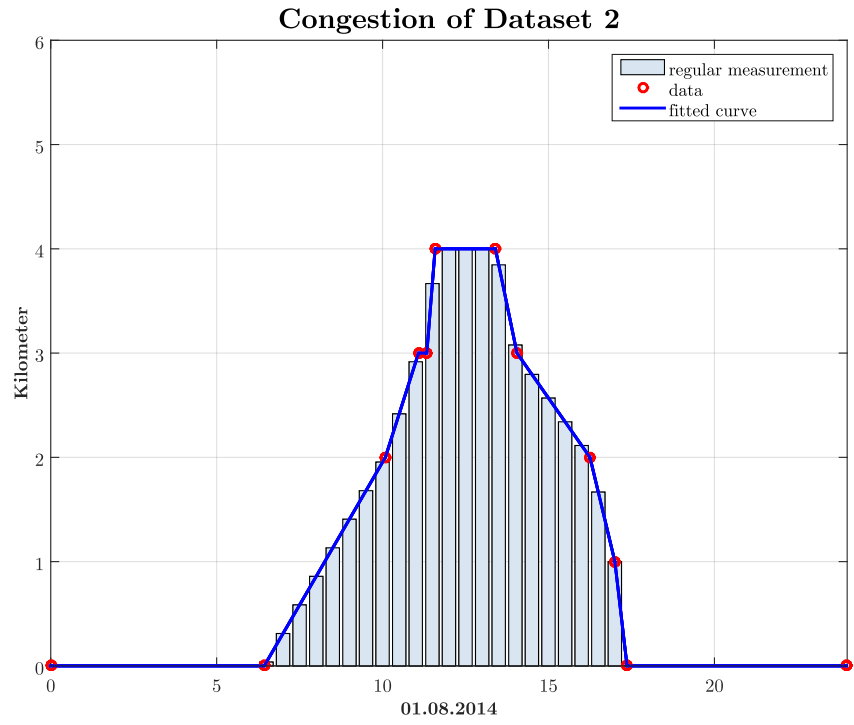
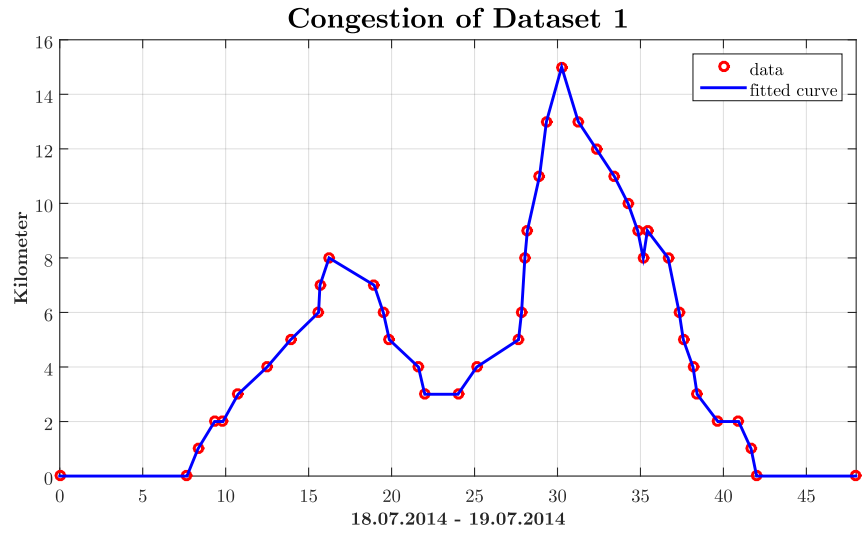


Figure 9: Dataset 2, added regular measurements for comparison

6.2 Train model

There are two parameters to fit the model:

- moveProb: describes how accurate cars moving forward
- moveCorr: increases moveProb in case of congestion

The program optiFinder was written to find the best setting of these two parameters. Important here by are the quantities *prediction* and *precision*. The latter describes the relative ratio between the area outside the intersection and the area of the reference data itself. One can imagine two portions A and B and its intersection $A \cap B$. The precision would be $\text{precision} = 1 - (A \cup B - A \cap B)/A$, where A is the area from the reference data.

Prediction is the ratio between all x-values that are inscribed by the intersection of the two curves and the reference curve. The big difference to precision is the way of counting. It would be very beneficial for the prediction value, if there is a lot of congestion. The outcome of this would be a poor precision value. That's why precision and prediction are used to find the optimal parameter. Because precision is more important, the weight for it is three times greater than for prediction.

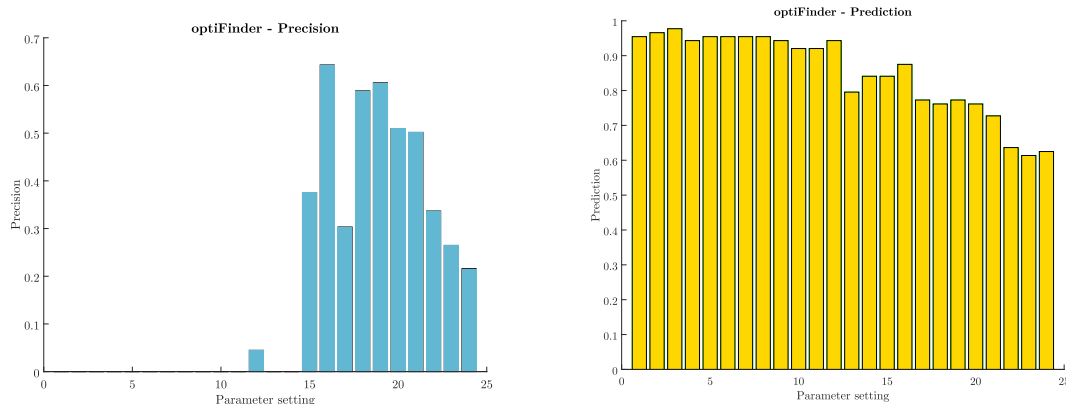


Figure 10: left: precision, right prediction for Dataset 2

The weighting of prediction and precision lead to the values for moveProb and moveCorr.

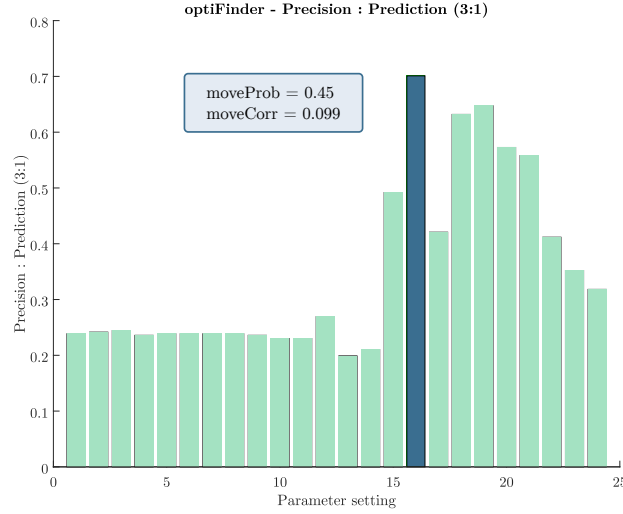


Figure 11: resulting values from weighting prediction (1x) and precision (3x)

7 Summary and Outlook

The authors created a model based on the Nagel-Schreckenberg model. It was extended to fit the situation at the Gotthard-Strassentunnel Nordportal between Erstfeld and Göschenenen. Its purpose is to predict traffic congestions in front of the tunnel.

The model was firstly trained with two datasets (24h and 48h), which were both provided by ASTRA.

The authors soon realised that there were some uncasualities in the congestion plot diagrams, which lead back to measuring mistakes or inaccuracy of the dataset. They had to adjust these abnormalities in order to obtain appropriate output.

At the very end, the finished model had to prove its accuracy by predicting traffic congestions with two other datasets (24h and 48h, also provided by ASTRA), which have not been used before.

7.1 Outlook

Both of the authors invested much time and effort in this project, and they really liked working on it. It is indisputable that some misleading measurements from the datasets distorted the results, but the results...

Without a doubt systems of this nature will play an important role in the future.

8 Appendix

References

- [1] Andreas Schadschneider. *Traffic flow modelling*. <http://www.thp.uni-koeln.de/~as/MyPage/traffic.html>. April 2000
- [2] Torsten Held, Stefan Bittihn. *Cellular automata for traffic simulation – Nagel-Schreckenberg model*. March 2011
- [3] K. Nagel and M. Schreckenberg. *A cellular automaton model for freeway traffic*. 1992

List of Figures

1	Nagel-Schreckenberg model – Symbols	7
2	Sketch of model	8
3	map of Uri, A: Erstfeld, B: Gotthard tunnel at Göschenen	9
4	Units	9
5	Variables	10
6	Inflow values with average speeds	11
7	Congestion data	12
8	Dataset 1	13
9	Dataset 2, added regular measurements for comparison	13
10	left: precision, right prediction for Dataset 2	14
11	resulting values from weighting prediction (1x) and precision (3x) . .	15

Code

Listing 1: interpolate

```
1 function [Y_congestion, data] = interpolate(dataset, interval, showPlot)
2 %% parameters
3 % INPUT:
4 %   dataset:      choose one of 4 datasets, {1,2,3,4}
5 %   interval:     interval of measured congestion in minutes
6 %   showPlot:     show interpolated curve, {0=false, 1 = true}
7 % OUTPUT:
8 %   Y_congestion: length of congestion, #measures depend on interval
9 %   data:         vector, contains time data
10
11 %% Prepare data (ASTRA)
12
13 switch dataset
14     case 1
15         time = {'18.07.2014 00:00' '18.07.2014 07:40' '18.07.2014 08:20' '
16                 18.07.2014 09:20' '18.07.2014 09:50' '18.07.2014 10:45' '18.07.2014
17                 12:30' '18.07.2014 13:55' '18.07.2014 15:36' '18.07.2014 15:41' '
18                 18.07.2014 16:14' '18.07.2014 18:57' '18.07.2014 19:30' '18.07.2014
19                 19:52' '18.07.2014 21:37' '18.07.2014 22:00' '19.07.2014 00:00' '
20                 19.07.2014 01:08' '19.07.2014 03:39' '19.07.2014 03:50' '19.07.2014
21                 04:02' '19.07.2014 04:11' '19.07.2014 04:55' '19.07.2014 05:20' '
22                 19.07.2014 06:15' '19.07.2014 07:15' '19.07.2014 08:20' '19.07.2014
23                 09:25' '19.07.2014 10:15' '19.07.2014 10:50' '19.07.2014 11:10' '
24                 19.07.2014 11:25' '19.07.2014 12:40' '19.07.2014 13:20' '19.07.2014
25                 13:35' '19.07.2014 14:10' '19.07.2014 14:25' '19.07.2014 15:40' '
26                 19.07.2014 16:55' '19.07.2014 17:40' '19.07.2014 18:00' '19.07.2014
27                 23:59'};
28         congestion = [0 0 1 2 2 3 4 5 6 7 8 7 6 5 4 3 3 4 5 6 8 9 11 13 15 13 12
29                       11 10 9 8 9 8 6 5 4 3 2 2 1 0 0];
30     case 2
31         time = {'01.08.2014 00:00' '01.08.2014 06:26' '01.08.2014 10:05' '
32                 01.08.2014 11:05' '01.08.2014 11:20' '01.08.2014 11:35' '01.08.2014
33                 13:24' '01.08.2014 14:03' '01.08.2014 16:15' '01.08.2014 17:00' '
34                 01.08.2014 17:22' '01.08.2014 23:59'};
35         congestion = [0 0 2 3 3 4 4 3 2 1 0 0];
36     case 3
37         time = {'25.07.2014 00:00' '25.07.2014 09:55' '25.07.2014 10:10' '
38                 25.07.2014 11:45' '25.07.2014 11:55' '25.07.2014 12:00' '25.07.2014
39                 12:40' '25.07.2014 13:35' '25.07.2014 16:45' '25.07.2014 17:10' '
40                 25.07.2014 18:30' '25.07.2014 19:14' '25.07.2014 20:48' '25.07.2014
41                 21:50' '25.07.2014 23:44' '26.07.2014 00:00' '26.07.2014 01:57' '
42                 26.07.2014 02:38' '26.07.2014 03:45' '26.07.2014 04:20' '26.07.2014
43                 04:36' '26.07.2014 04:55' '26.07.2014 05:30' '26.07.2014 05:40' '
44                 26.07.2014 07:50' '26.07.2014 09:00' '26.07.2014 10:15' '26.07.2014
45                 12:00' '26.07.2014 13:00' '26.07.2014 13:45' '26.07.2014 14:30' '
46                 26.07.2014 14:55' '26.07.2014 15:10' '26.07.2014 16:33' '26.07.2014
47                 16:47' '26.07.2014 17:01' '26.07.2014 23:59'};
48         congestion = [0 0 1 3 4 5 6 7 6 5 4 3 4 3 4 4 4 4 5 6 7 8 9 11 13 15
49                       13 10 8 6 4 3 2 1 0 0];
50     case 4
51         time = {'02.08.2014 00:00' '02.08.2014 05:10' '02.08.2014 05:30' '
52                 '02.08.2014 05:50' '02.08.2014 06:10' '02.08.2014 06:30' '02.08.2014
53                 06:50' '02.08.2014 07:10' '02.08.2014 07:30' '02.08.2014 07:50' '
54                 02.08.2014 08:10' '02.08.2014 08:30' '02.08.2014 08:50' '02.08.2014
55                 09:10' '02.08.2014 09:30' '02.08.2014 09:50' '02.08.2014 10:10' '
56                 02.08.2014 10:30' '02.08.2014 10:50' '02.08.2014 11:10' '02.08.2014
57                 11:30' '02.08.2014 11:50' '02.08.2014 12:10' '02.08.2014 12:30' '
58                 02.08.2014 12:50' '02.08.2014 13:10' '02.08.2014 13:30' '02.08.2014
59                 13:50' '02.08.2014 14:10' '02.08.2014 14:30' '02.08.2014 14:50' '
60                 02.08.2014 15:10' '02.08.2014 15:30' '02.08.2014 15:50' '02.08.2014
61                 16:10' '02.08.2014 16:30' '02.08.2014 16:50' '02.08.2014 17:10' '
62                 02.08.2014 17:30' '02.08.2014 17:50' '02.08.2014 18:10' '02.08.2014
63                 18:30' '02.08.2014 18:50' '02.08.2014 19:10' '02.08.2014 19:30' '
64                 02.08.2014 19:50' '02.08.2014 20:10' '02.08.2014 20:30' '02.08.2014
65                 20:50' '02.08.2014 21:10' '02.08.2014 21:30' '02.08.2014 21:50' '
66                 02.08.2014 22:10' '02.08.2014 22:30' '02.08.2014 22:50' '02.08.2014
67                 23:10' '02.08.2014 23:30' '02.08.2014 23:59'};
```

```

                '02.08.2014 06:11' '02.08.2014 06:40' '02.08.2014 07:25' '02.08.2014
                07:50' '02.08.2014 08:05' '02.08.2014 08:45' '02.08.2014 08:56' '
                02.08.2014 10:48' '02.08.2014 10:53' '02.08.2014 11:30' '02.08.2014
                12:08' '02.08.2014 12:51' '02.08.2014 13:42' '02.08.2014 14:05' '
                02.08.2014 14:15' '02.08.2014 14:30' '02.08.2014 14:45' '02.08.2014
                15:03' '02.08.2014 16:00' '02.08.2014 16:17' '02.08.2014 23:59'};
25         congestion = [ 0 0 2 3 4 5 6 7 8 9 11 13 11 10 9 8 7 6 5 4 3 2 0 0];
26     otherwise
27         disp(['dataset ' num2str(dataset) ' does not exist. Available datasets:
                training(1, 2); evaluation(3,4)'])
28         return
29     end
30
31     % generate date vector
32     data = datevec(time, 'dd.mm.yyyy HH:MM');
33
34     % compute #seconds since 00:00 of first day
35     seconds = zeros(1, length(time));
36     first_date = datenum(data(1,1:3));
37     for i = 1:length(time)
38         curr_date = datenum(data(i,1:3));
39         if curr_date == first_date
40             seconds(1,i) = data(i,4)*3600 + data(i,5)*60;
41             xMax = 24;
42         else
43             seconds(1,i) = data(i,4)*3600 + data(i,5)*60 + 24*3600;
44             xMax = 48;
45         end
46     end
47
48     % data for curve fitting
49     X = seconds / 3600;
50     Y = congestion;
51
52     %% Curve Fitting
53     [xData, yData] = prepareCurveData(X, Y);
54
55     % Set up fitttype and options.
56     ft = 'linearinterp';
57     opts = fitoptions(ft);
58     opts.Normalize = 'on';
59
60     % Fit model to data.
61     [fitresult] = fit(xData, yData, ft, opts);
62
63     if showPlot
64         % Plot fit with data.
65         figure('Name', ['Dataset ' num2str(dataset)]);
66         p = plot(fitresult, '-b', xData, yData, 'or');
67         set(p, 'LineWidth', 2)
68
69         % Label axes
70         ylim([0 16]);
71         if mod(dataset,2)
72             xlabel([datestr(data(1,:), 'dd.mm.yyyy') ' - ' datestr(data(end,:), 'dd.mm.
                    yyyy')], 'fontweight', 'bold', 'fontsize', 11);
73         else
74             xlabel(datestr(data(1,:), 'dd.mm.yyyy'), 'fontweight', 'bold', 'fontsize', 11);

```

```

75     if dataset == 2
76         ylim([0 6]);
77     end
78 end
79 xlim([0 xMax]);
80 ylabel('Kilometer', 'fontweight','bold','fontsize',11);
81 t = title(['Congestion of Dataset ' num2str(dataset)]);
82 set(t,'fontweight','bold','fontsize', 18);
83 grid on
84 end
85 %% Generate congestion data for error-evaluation
86 Y_congestion = fitresult(interval/60:interval/60:xMax);
87 end

```

Listing 2: optiFinder

```

1 function [moveProb, moveCorr] = optiFinder(dataset)
2 % Optimum Finder for:
3 %   - moveProb
4 %   - moveCorr
5
6 %% moveCorr
7 mc_start = 0;
8 mc_stop = .099;
9 mc_step = .033;
10
11 %% moveProb
12 mp_start = .3;
13 mp_stop = .55;
14 mp_step = .05;
15
16 isAnimated = 0;
17 rounds = 4;
18 numberMC = round((mc_stop-mc_start)/mc_step+1);
19 numberMP = round((mp_stop-mp_start)/mp_step+1);
20 evaluation = zeros(2,numberMC * numberMP);
21 precision_tot = 0;
22 prediction_tot = 0;
23 v = 0;
24
25 set(0,'DefaultFigureVisible','off') % suppress bar graph output
26 for moveProb = mp_start:mp_step:mp_stop
27     for moveCorr = mc_start:mc_step:mc_stop
28         v = v+1;
29         for j = 1:rounds
30             [precision,prediction] = NaSch_Datasets_v1(dataset, moveProb, isAnimated
31                 , moveCorr);
32             precision_tot = precision_tot + precision;
33             prediction_tot = prediction_tot + prediction;
34         end
35         evaluation(1,v) = precision_tot/rounds;
36         evaluation(2,v) = prediction_tot/rounds;
37         precision_tot = 0;
38         prediction_tot = 0;
39     end
40 end
41 % diagram of different errors
42 set(0,'DefaultFigureVisible','on') % do not suppress following bar graph output
43 figure()
44 hold on;
45 title('optiFinder - Precision')
46 x = 1:length(evaluation);
47 y = evaluation(1,:);
48 xlabel('Parameter setting');
49 ylabel('Precision');
50 bar(x,y, 'EdgeColor','g', 'FaceColor','g')
51
52 figure()
53 hold on;
54 title('optiFinder - Prediction')
55 x = 1:length(evaluation);
56 y = evaluation(2,:);

```

```

56 xlabel('Parameter setting');
57 ylabel('Prediction');
58 bar(x,y, 'EdgeColor','g', 'FaceColor','g')
59
60 figure()
61 hold on;
62 title('optiFinder - Precision : Prediction (3:1)')
63 x = 1:length(evaluation);
64 y = (3*evaluation(1,:)+evaluation(2,:))/4;
65 xlabel('Parameter setting');
66 ylabel('Precision : Prediction (3:1)');
67 bar(x,y, 'EdgeColor','g', 'FaceColor','g')
68
69 [Max, Index] = max(y);
70 T = zeros(numberMC, numberMP);
71 T(Index) = 1;
72 [value, location] = max(T(:));
73 [R,C] = ind2sub(size(T),location);
74 moveProb = mp_start + (C-1) * mp_step;
75 moveCorr = mc_start + (R-1) * mc_step;
76 end

```

Listing 3: model.m

```

1 function [precision, prediction] = NaSch_Datasets_v1(dataset, moveProb,
    isAnimated, moveCorr)
2
3 % values
4 % NaSch_Datasets_v1(1,.525,0,0,0,.055): perfectly symmetric
5 % NaSch_Datasets_v1(2,.525,0,0,0,.05): less error
6
7 %% parameters
8 % INPUT:
9 %   dataset: choose one of 4 datasets, {1,2,3,4} (don't use 3,4:evaluation)
10 %   moveProb: the probability for a car to move forwards, 0..1
11 %   isAnimated: start program with animation, boolean 1=true 0=false
12 % OUTPUT:
13 %   error_tot: total
14 %
15 % EXAMPLE:
16 %   NaSch_lC_Stats_v1(2, .5, 0 , 1, 0)
17
18 % parameters for comparison model - reality
19 lC = 4.5; % length of each cell (average length of cars => Skoda
    Octavia)
20 lR = 19000; % length in Reality (Erstfeld - Goeschenen, 13min)
21 N = round(lR / lC); % 4222 cells
22 measureInterval = 30; % measure every #min
23 [I, S] = Datasets(dataset);
24 nIter = length(I)*180; % number of iterations; datasets: #cars/180s => we want
    1 iteration / s
25 q = 0; % running variable for reading outSet
26 redLight_act = 1; % activate redLight
27
28 % set parameter values
29 conv = 1000/lC; % "convert", #cells that matches 1km
30 cC = N-22; % "change cell", where cars have to change the lane
31 vmax = 5; % maximal velocity of the cars (vmax = 5 = 100 km/h)
32 L = 11; % length of lane where cars can change (in front of cC)
33 vmax_L = 3; % maximal velocity in L
34 a = 0.2; % min probability that car changes lane at cC - L
35 b = 0.6; % max probability that car changes lane at cC
36 laneChange = .1; % the probability for a car to change the lane, 0..1
37 dropCounter = 2; % dropCounter: #seconds a car can pass the redlight, redlight
    is active
38 % after 2km congestion. Use '1' to turn off redlight. Do NOT
39 % use odd numbers.
40 redLight = 0; % automatic redLight, do NOT turn on
41 inflowCounter = 0; % count cars
42
43 % use quadratic increments for the probability between a and b ( $p = k \cdot x^2 + d$ )
44 k = (a - b)/(L*L-2*cC*L); % for  $x=cC-L$  is  $p = a$ 
45 d = b - cC*cC*(a - b)/(L*L-2*cC*L); % for  $x=cC$  is  $p = b$ 
46
47 % set statistical variables
48 vSum = 0; % sum of speeds
49 nCars = 0; % #cars on road
50
51 % define variables in a block (2 x bL)
52 bL = 50; % block length: length of a block // 225m

```

```

53 bD = 0;           % block density: density of cars in a block, 0..1
54 bV = 0;           % block velocity: average velocity in a block, 0..vmax
55 bC = 0;           % block counter: counts number of blocks (congestion), 0..(N %
    bL)
56 bE = 0;           % empty counter: counts number of blocks (no congestion),
    {0,1,2}
57
58 % congestion length in each round
59 if mod(dataset,2)
60     divider = 48*60 / measureInterval;
61 else
62     divider = 24*60 / measureInterval;
63 end
64 congLength = zeros(1,nIter/divider);
65 congPlot = zeros(1,divider);
66 currentCongestion = 0;
67 first = 1;          % congestion optimization
68 opt_act = 0;
69 congStart = 0;
70
71 % define road (-1 = no car; 0..vmax = car, value represents velocity)
72 X = -ones(2,N);
73
74 % take average inflow (every 2 hours)
75 if mod(dataset,2)
76     inflow = zeros(1,24);
77     for i = 0:23
78         inflow(1,i+1) = sum(I(1,1+i*40:40+i*40)) / 40;
79     end
80 else
81     inflow = zeros(1,12);
82     for i = 0:11
83         inflow(1,i+1) = sum(I(1,1+i*40:40+i*40)) / 40;
84     end
85 end
86 p=1;
87
88 %% main loop, iterating the time variable, t
89 for t = 1:nIter
90     %% NaSch and laneChange
91     % cars change lane with given probability laneChange
92     for i = 1:cC-L
93         % left to right --> probability laneChange
94         if X(1,i) ~= -1 && X(2,i) == -1 && rand < laneChange
95             X(2,i) = X(1,i);
96             X(1,i) = -1;
97         end
98         % right to left --> probability 0.95*laneChange
99         if X(2,i) ~= -1 && X(1,i) == -1 && rand < 0.95*laneChange
100             X(1,i) = X(2,i);
101             X(2,i) = -1;
102         end
103     end
104
105     % acceleration (NaSch -- RULE 1) =====
106     for i = 1:2*N
107         if X(i) ~= -1 && X(i) < vmax
108             X(i) = X(i) + 1;

```

```

109     end
110 end
111
112 % cars have to change lane due to lane reduction
113 for i = cC-L:cC
114
115     % reduce velocity of lanes in front of lane reduction
116     if X(1,i) > vmax_L
117         X(1,i) = vmax_L;
118     elseif X(2,i) > vmax_L
119         X(2,i) = vmax_L;
120     end
121
122     % lane change on road
123     if X(1,i) ~= -1 % change lane if possible
124         if i == cC && X(2,i) == -1
125             X(2,i) = X(1,i) + 2; % accelerate after lane change
126             X(1,i) = -1;
127         elseif X(2,i) == -1 && rand < (k*i*i + d)
128             X(2,i) = X(1,i) + 2; % +2 accelerate
129             X(1,i) = -1;
130         elseif X(1,i)+i > cC % avoid overrunning changeCell
131             X(1,i) = cC-i;
132         end
133     end
134 end
135
136 % red light
137 if X(1,cC-10) ~= -1 && redLight
138     X(1,cC-10) = 0;
139     if mod(t+dropCounter/2,dropCounter) == 0
140         X(1,cC-10) = 2;
141     end
142 end
143
144 if X(2,cC-10) ~= -1 && redLight
145     X(2,cC-10) = 0;
146     if mod(t,dropCounter) == 0
147         X(2,cC-10) = 2;
148     end
149 end
150
151 % slowing down (NaSch -- RULE 2) =====
152 for row = 1:2
153     for i = 1:N
154         if X(row,i) ~= -1
155             for j = 1:X(row,i)
156                 if i+j <= N
157                     if X(row,i+j) ~= -1
158                         X(row,i) = j-1;
159                     break
160                 end
161             end
162         end
163     end
164 end
165 end
166

```



```

167 % randomization (NaSch -- RULE 3) =====
168 for i = 1:2*N
169     if X(i) > 0
170         X(i) = X(i) - (rand > moveProb);
171     end
172 end
173
174 % car motion (NaSch -- RULE 4) =====
175
176 %% inflow
177 % update positions X(1..N)
178 Xold = X;
179 for row = 1:2
180     for i = 1:N
181         if Xold(row,i) > 0 && i + Xold(row,i) <= N
182             X(row,i+X(row,i)) = X(row,i);
183             X(row,i) = -1;
184         elseif Xold(row,i) > 0 && i + Xold(row,i) > N
185             X(row,i) = -1;
186         end
187     end
188 end
189
190 if t > q*180 % datasets: #cars / 180s => every 180s we take new inflow value
191     from dataset
192     q = q + 1;
193
194     % average inflow
195     if mod(q,40) == 0
196         p = p + 1;
197     end
198
199     if mod(dataset,2)
200         if p > 24
201             p = 24;
202         end
203     else
204         if p > 12
205             p = 12;
206         end
207     end
208
209     rateI = inflow(1,p)/(2*180);
210
211     rateS_m = S(1,q); % mean speed
212     rateS_m = rateS_m/(3.6*5.55555); % convert km/h into NaSch-units
213     if rateS_m > 5
214         rateS_m = 5;
215     elseif rateS_m < 2
216         rateS_m = 2;
217     end
218
219     rateS = ceil(rateS_m) - (rand < (ceil(rateS_m)-rateS_m));
220
221 % update position X(1,1) left lane (inflow left)
222 % calculate inflow rate per second, divide by 2 because
223 % the 2 rows, multiply by .95 because left lane
224 rate = rateI*.95;

```

```

224
225     if rand < rate && X(1,1) == -1
226         X(1,1) = rateS; % all cars enter with speed of rateS
227         inflowCounter = inflowCounter + 1;
228     end
229
230     % update position X(2,1) right lane (inflow right)
231     rate = rateI*1.05;
232
233     if rand < rate && X(2,1) == -1
234         X(2,1) = rateS; % all cars enter with speed of rateS
235         inflowCounter = inflowCounter + 1;
236     end
237
238     %% statistics
239     % average speed (only with animation)
240     if isAnimated
241         for row = 1:2
242             for i = 1:cC
243                 if X(row,i) ~= -1
244                     vSum = vSum + X(row,i);
245                     nCars = nCars + 1;
246                 end
247             end
248         end
249
250         vAverage = vSum / nCars;
251         % reset vSum for next round
252         vSum = 0;
253         nCars = 0;
254     end
255
256     % congestion length
257     for i = cC:-bL:1
258         % compute density of cars and average velocity in a block
259         for j = 2*i:-1:2*(i-bL) + 1
260             if X(j) ~= -1
261                 bV = bV + X(j);
262                 bD = bD + 1;
263             end
264         end
265         if bD == 0
266             bD = 1;
267         end
268         bV = bV / bD;
269         bD = bD / (2*bL);
270
271         % test if block satisfy conditions for a congestion
272         % count only connected congestion, gaps are allowed.
273         % gap is #gaps allowed between two 'congestion-blocks'
274         gap = 1;
275         if bV < 1 && bD > .48
276             bC = bC + 1;
277             bE = 0;
278         elseif bE >= gap
279             bC = bC - gap;
280             break
281         else

```

```

282         bE = bE + 1;
283         bC = bC + 1;
284     end
285     % reset variables
286     bV = 0;
287     bD = 0;
288 end
289
290 k = measureInterval * 60;
291 congLength(1,1+mod(t-1,k)) = bC * bL;
292 % measure congestion for interval[min], store mean in congPlot
293 if mod(t,k) == 0
294     congPlot(1,t/k) = sum(congLength/conv) / k;
295     %congPlot(1,t/k) = round((sum(congLength/conv) / k)/.5)*.5;    % round
        half up
296
297     % "traffic optimization" (faster congestion growth at beginning)
298     if congPlot(1,t/k) > 1 && first == 1
299         moveProb = moveProb + moveCorr;    % slow down moveProb when
            congestion begins
300         first = 0;
301         congStart = t;
302         opt_act = 1;
303     end
304
305     % enable redlight, if congestion longer than 1km
306     if congPlot(1,t/k) > 1 && redLight_act
307         redLight = 1;
308     else
309         redLight = 0;
310     end
311     currentCongestion = congPlot(1,t/k);
312 end
313
314 % reset moveProb after x*3600 seconds
315 if t >= congStart + 2*3600 && currentCongestion < 2 && opt_act
316     moveProb = moveProb - moveCorr;
317     first = 0;
318     opt_act = 0;
319 end
320
321 % reset counters
322 bC = 0;
323 bE = 0;
324
325 %% animation
326 if isAnimated
327     %clf;
328     hold on;
329     xlim([N-100 N+1])
330     ylim([-20 20])
331     plot(N-100:cC+1, 0.5*ones(1,length(N-100:cC+1)), 'Color', [.75 .75 .75],
        'LineWidth', 12)
332     plot(N-100:N+1, -0.5*ones(1,length(N-100:N+1)), 'Color', [.75 .75 .75],
        'LineWidth', 12)
333     plot(N-100:cC+1, 0*(N-100:cC+1), '--', 'Color', [.95 .95 .95], '
        LineWidth', .8)
334     title([' Iterationsschritt: ' num2str(t), ' Congestion: ' num2str(

```

```

        currentCongestion), ' Average Speed: ' num2str(vAverage), ' inFlow:
        ' num2str(rateI)])
335
336     for row = 1:2
337         for i = N-100:N
338             if X(row,i) ~= -1
339                 draw_car(i, 1.2*(1.5-row), 0.8, 0.2);
340             end
341         end
342     end
343     pause(0.01)
344 end
345 end
346 %% END OF MAIN LOOP
347
348 disp(['inflow and outflow are equal to ' num2str(inflowCounter)]);
349
350
351 % diagram of comparison model - reality
352 [Y_dataset, data] = interpolate(dataset, measureInterval, 0);
353 figure
354 hold on;
355 title(' Comparison Model - Reality ')
356 xval = measureInterval / 60;
357 bar(xval:xval:nIter/3600, congPlot);
358 bar(xval:xval:nIter/3600, Y_dataset, .4, 'FaceColor', [.8, .85, .9])
359
360 ylim([0 16]);
361 if mod(dataset, 2)
362     xlabel([datestr(data(1,:), 'dd.mm.yyyy') ' - ' datestr(data(end,:), 'dd.mm.
363         yyyy')]);
364 else
365     xlabel(datestr(data(1,:), 'dd.mm.yyyy'));
366     if dataset == 2
367         ylim([0 6]);
368     end
369 ylabel('Kilometer');
370 if mod(dataset, 2)
371     xlim([0 48]);
372 else
373     xlim([0 24]);
374 end
375 grid on;
376 hold off;
377
378 %% error evaluation
379 area_dataset = sum(Y_dataset);
380 error = 0;
381 for i = 1:length(congPlot)
382     error = error + abs(congPlot(i)-Y_dataset(i));
383 end
384 error = error / area_dataset;
385 if error > 1
386     error = 1;
387 end
388 precision = 1 - error;
389

```

```

390 intersect = 0;
391 soloCounter = 0;
392 for i = 1:length(congPlot)
393     if Y_dataset(i) > 0
394         soloCounter = soloCounter + 1;
395         if congPlot(i) > 0
396             intersect = intersect + 1;
397         end
398     end
399 end
400 prediction = intersect / soloCounter;
401
402 end
403
404 %% datasets
405 function [InFlow, Speed] = Datasets(inSet)
406 switch inSet
407     case 1
408         InFlow = csvread('Dataset.csv', 2, 1, 'B3..B962');
409         Speed = csvread('Dataset.csv', 2, 2, 'C3..C962');
410     case 2
411         InFlow = csvread('Dataset.csv', 2, 5, 'F3..F482');
412         Speed = csvread('Dataset.csv', 2, 6, 'G3..G482');
413     case 3
414         InFlow = csvread('Dataset.csv', 2, 3, 'D3..D962');
415         Speed = csvread('Dataset.csv', 2, 4, 'E3..E962');
416     case 4
417         InFlow = csvread('Dataset.csv', 482, 5, 'F483..F962');
418         Speed = csvread('Dataset.csv', 482, 6, 'G483..G962');
419 end
420 end

```