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

Project Report

**Emergency Evacuation**

Waldburger Dominik & Zehnder Matthias

Zurich
May 2010
Eigenständigkeitserklärung


Waldburger Dominik          Zehnder Matthias
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.

Waldburger Dominik        Zehnder Matthias
Contents

1 Individual contributions 6

2 Introduction and Motivations 6

3 Description of the Model 7
   3.1 Introduction . 7
   3.2 The auditoriums 7
      3.2.1 Our auditoriums 7
   3.3 Time descretisation 10
   3.4 The decision pyramid 10
   3.5 Step one 11
   3.6 Step two 11
   3.7 Step three 11

4 Implementation 12
   4.1 Introduction . 12
   4.2 No subdiscretization 12
   4.3 Input . 12
   4.4 Ground structure 13
      4.4.1 waym 13
      4.4.2 studm 14
      4.4.3 studc 15
      4.4.4 studl 15
   4.5 Stumble and decision making 15
      4.5.1 studt 15
      4.5.2 studp 16
      4.5.3 wayt 16
   4.6 Visualisation and statistics 16
      4.6.1 p 17
      4.6.2 stats 17
   4.7 Simulation.m 17
   4.8 preperation.m 19
   4.9 pressure.m 19
   4.10 border.m 21
   4.11 dulr.m udrl.m 21
   4.12 picture.m 21
   4.13 stumble.m 21
   4.14 waytime.m 21
5 Simulation Results and Discussion
  5.1 Results ........................................... 28
  5.2 Discussion ....................................... 34
    5.2.1 Influence of exit door distribution ............. 34
    5.2.2 Influence of Aisle width ....................... 35
    5.2.3 Influence of different room layouts ............ 35
    5.2.4 Model weaknesses ............................. 35

6 Summary and Outlook ............................... 36

7 References .......................................... 37

8 Code .................................................. 38
  8.1 Simulation.m .................................. 38
  8.2 preperation.m .................................. 40
  8.3 step1.m ....................................... 41
  8.4 step2.m ....................................... 44
  8.5 step3.m ....................................... 46
  8.6 alternivway.m .................................. 48
  8.7 border.m ....................................... 49
  8.8 dulr.m ......................................... 49
  8.9 udrl.m ......................................... 49
  8.10 stumble.m ..................................... 49
  8.11 picture.m ..................................... 50
  8.12 statistic.m .................................... 50
  8.13 directway.m .................................... 51
  8.14 laststep.m ..................................... 52
  8.15 pressure.m ..................................... 53
  8.16 pictureshow.m .................................. 54
  8.17 maps.m ......................................... 55
1 Individual contributions

At the beginning of our project we worked a lot together to bring our ideas down to a model. We defined in detail how our model should work and simulate our problem. We defined our parameters which we wanted to investigate in detail.

In the further working Dominik Waldburger was responsible for the implementation. Especially he took care that the interfaces between the different functions were functionable. We both worked on the different functions.

Matthias Zehnder was at the end of the project responsible for the report. Dominik Waldburger contributed the part of the implementation.

2 Introduction and Motivations

Emergency evacuation in a building like the ETH Hauptgebäude is a very important topic. We all hope that there will never be a situation where an emergency evacuation is necessary.

We wanted to investigate the dynamics of students in case of an emergency, when they have to leave the auditorium in a hurry. We wanted to analyze the influence of different arrangements of seats, doors and escape routes.

For us it was interesting to discuss how a model could evaluate the fastest way to the exit. Also in this problem we asked us how the model can determine if a detour is better than the direct way out. We placed special emphasis on modeling this decision making process in cases where more than one person is demanding the same empty space on the evacuation route or in cases where a detour would be faster than the direct way out.

Various parameters have been investigated like the width of the aisles, the placements of the exits and a seat arrangement other than straight rows. A special safety aspect we looked into was the pressure behind exiting people build up by the length of the queue.

A further aspect we considered was optimizing the time discretisation by elimination of the influence of the implementation itself.

We always have been fully aware of the fact that the model would never be able to represent the reality. We tried to brake down our model to a level detailed enough to allow us to transfer the conclusions into real world. We especially tried to further develop the approach shown in the lecture to get a better understanding in modeling such problems.
3 Description of the Model

3.1 Introduction

Basically, for our model we take the idea of a Cellular Automata like the one we discussed in the lecture. However, unlike in the lecture, where the decision was based solely on the neighbor cells, in our model, it is based on the empty spaces, the demands for them and the pressure for the demands as well as the benefit to move to a particular space.

3.2 The auditoriums

The auditoriums in our Model are two dimensional matrices, where each point in the matrix describes either a person, a desk, a wall, an exit or a free space. We do not distinguish between walls and desks, so it is not possible to climb desks. Allowed exit routes lead only through free spaces. To be able to compare various room layouts, we always choose the same room area, meaning the same matrix dimensions. This means that the number of people vary from room to room. Basic layouts of the model rooms derives from the auditoriums in the HPH and in the HG.

3.2.1 Our auditoriums

Legend: grey = wall  white = door  green to yellow = people w. dif. pressures

Auditorium one:
Auditorium two:

Auditorium three:
Auditorium four:

Auditorium five:
3.3 Time discretisation

In order to avoid influence of the implementation on the time discretisation we prohibit that the checking order influences the decision of the people. The model fulfills that by comparing the demand weights for multiple allocations for a free space. Furthermore, the steps are taken only after all comparisons have been done and the priorities have been set.

3.4 The decision pyramid

Where is the nearest exit? Which one is the shortest way? Would a detour be faster? Do I have a free space to move to?

The decision for the next move of a person is based on these aspects. To take this decision our model works in three steps. In the first step the decision is sought based on free spaces, the closeness to the exit, demands and the pressure of others. If no clear decision can be taken in a a second step the model takes a random decision. In a third step the model checks for blocked people which could benefit from a detour. Our model also takes care of the possibility that a person can stumble in which case a possible move cannot be taken.
3.5  Step one

In the first step a person looks at the four places around him. He checks if the places are free and whether they bring him closer to the exit. If a place fulfills these requirements, it checks if there are others demanding the same free space. If there is no such demand it reserves this space. If there are other demands on a particular free space the model compares the pressure, that is the queue length behind the people, and makes the reservation based on the highest pressure. If no clear decision can be taken no one gets the space in step one.

3.6  Step two

In the second step the model decides among the candidates with the same pressure which one gets the reservation. In our model we use a random generator for this decision.

3.7  Step three

For all people with adjacent free spaces but without reservations the model checks the availability and the benefit of a possible detour. To do that the model checks whether the person is already on a detour or not.

If not he looks for a detour and the benefit of taking it, based on the increasing length of the way and the projected time on the direct way. If there is a positive match it makes the reservation.

If he is on a detour it checks whether the detour is still available and still would be a benefit, then he remains on the detour, if not he goes back to the direct way.
4 Implementation

4.1 Introduction

In this chapter we describe the implementation of our model in MATLAB illustrated by some selected parts of the code. The full code can be found in the chapter Code.

4.2 No subdiscretization

As mentioned we like to avoid a subdiscretization. That’s why we can’t save just the position of all students in the room. For each student we have to save the actual position and the position he’d like to go in the next step. To implement this in MATLAB we save the information in an s (s = number of students) x 2 or s x 4 matrices with alternate access:

\[ t = \text{the global discretization time variable} \]

<table>
<thead>
<tr>
<th>Function</th>
<th>t=1</th>
<th>t=2</th>
<th>t=3</th>
<th>t=4</th>
<th>t=odd</th>
<th>t=even</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{mod}(t+1,2)+1 )</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>( \text{mod}(t,2)+1 )</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( 2 \times \text{mod}(t+1,2)+1 )</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( \text{mod}(t+1,2)+2 )</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

4.3 Input

The input for the simulation is an n x m integer-matrix of a room with the information about the walls (= -1), the exits (= 1) and the places of the students (= 2). The border-cells have to be a wall or an exit and for every student there have to be a way to an exit. For better handling the room-matrix get split in two matrices waym and studm and the students get numbered.
4.4 Ground structure

The ground structure is provided by the four files: waym, studm, studc, studl

4.4.1 waym

Waym stands for way-map. It’s an n x m integer-matrix with the same dimension as the given room. It contains the information about walls (= -1) and exits (= 1). In the other cells, the floor cells, is the number of steps written for the shortest way to an exit. This allows a fast request if a field is on the way to the nearest exit.
An example of a waym-matrix:

<table>
<thead>
<tr>
<th></th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Studm stands for student-map. It’s an n x m integer-matrix with the same dimension as waym. The position of each student is saved and reserved by his number in the corresponding cell. So fast answers to the questions if there’s a student on this field or which student is on this field is provided without searching the coordinates of all students.

An example of a studm-matrix:

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4.4.2 studm

Studm stands for student-map. It’s an n x m integer-matrix with the same dimension as waym. The position of each student is saved and reserved by his number in the corresponding cell. So fast answers to the questions if there’s a student on this field or which student is on this field is provided without searching the coordinates of all students.
4.4.3 studc

Studc stands for student-coordinates. It’s an s x 4 integer-matrix with the actual coordinates and the last or next coordinates of all students saved.

\[
\text{studc}(i, 2^*\text{mod}(t+1,2)+1) = \text{actual x-coordinate of the i-th student}
\]

\[
\text{studc}(i, 2^*\text{mod}(t+1,2)+2) = \text{actual y-coordinate of the i-th student}
\]

\[
\text{studc}(i, 2^*\text{mod}(t,2)+1) = \text{last or next x-coordinate of the i-th student}
\]

\[
\text{studc}(i, 2^*\text{mod}(t,2)+2) = \text{last or next x-coordinate of the i-th student}
\]

4.4.4 studl

Studl stands for student left in the room. It’s an s x 2 boolean-matrix and saves the information if a student is still in the room and if he had already chosen his next step.

\[
\text{studl}(i, \text{mod}(t+1,2)+1) = 1 \quad \text{i-th student is in the room and doesn’t have chosen his next step}
\]

\[
\text{studl}(i, \text{mod}(t+1,2)+1) = 0
\]

\[
\text{studl}(i, \text{mod}(t,2)+1) = 1 \quad \text{i-th student is in the room and has chosen his next step}
\]

\[
\text{studl}(i, \text{mod}(t,2)+1) = 0
\]

\[
\text{studl}(i, \text{mod}(t,2)+1) = 0 \quad \text{i-th student left the room}
\]

4.5 Stumble and decision making

To simulate different running speeds or accidents we let the students randomly waiting for some rounds. For the decision-making-process we must be aware of the pressure on the student and the possibility of a faster detour for every student in the room. This information is saved in the variables: studt, studp and wayt

4.5.1 studt

Studt stands for student-time. Its is a s x 3 integer-matrix saving the time the student have to wait because of stumbling, the time the student has waited and if the student is on an alternative way.
studt(i, 1) = time to wait of the i-th student
studt(i, 2) = time waited of the i-th student
studt(i, 3) = 1 i-th student is on alternative way (detour)
studt(i, 3) = 0 i-th student is not on alternative way (detour)

4.5.2 studp

Studp stands for student-pressure. The pressure from up, down, right, left on the
student is saved in a s x 4 double-matrix.

studp(i, 1) = the pressure on the i-th student from up
studp(i, 2) = the pressure on the i-th student from down
studp(i, 3) = the pressure on the i-th student from right
studp(i, 4) = the pressure on the i-th student from left

4.5.3 wayt

Wayt stands for way-time. It’s a s x 6 double-matrix saving the shortest way-time
on a direct and an alternative way. For the shortest way-time on the direct way it
sums up the time the students on the direct way to the exit have to wait. For the
shortest alternative way-time it sums up the time the students on the alternative
way have to wait and adds the step backwards which must be taken.

wayt(i, 1) = the shortest way-time for the i-th student on a direct way to the exit
wayt(i, 2) = the shortest way-time for the i-th student on an alternative way to the exit
wayt(i, 2) = -1 there exists no alternative way for the i-th student
wayt(i, 3:6) = 1 there’s a free field up, down, right, left on the way of the shortest
alternative way
wayt(i, 3:6) = 0 there’s no free field

4.6 Visualisation and statistics

For the visualisation and the statistics we save the current situation in the variables:
p and stats
4.6.1 \textbf{p}

P stands for picture. It’s an \( n \times m \times t \) (\( t \) number of discretization steps) 3-dimensional-double-matrix.

\[ p(:, :, t) = \text{a picture of the actual situation of the room} \]
\[ p(i, j, t) = 1 \text{ wall} \]
\[ p(i, j, t) = 2 \text{ floor} \]
\[ p(i, j, t) = 3 \text{ exit} \]
\[ p(i, j, t) = 4+ \text{ student plus his pressure} \]

4.6.2 \textbf{stats}

Stats stands for statistics and is a \( s \times 2 \times t \) 3-dimensional-double-matrix.

\[ \text{stats}(i, 1, t) = \text{pressure of the i-th student to the time t} \]
\[ \text{stats}(i, 2, t) = \text{time to wait of the i-th student to the time t} \]
\[ \text{stats}(i, :, t) = [-1 -1] \text{ i-th student left the room} \]

4.7 \textbf{Simulation.m}

The Simulation.m is the main m-file which coordinates the different functions. First it deletes all potentially existing variables which could disturb the simulation and runs the preperation.m function.
Now the decision-making, stumble, visualisation and statistic variables get computed and the students walk as long as every student has left the room. In the end uninteresting variables get deleted.

```matlab
% Simulate an emergency evacuation
% choose a map 0-6
mapn = 1;

% clears possible existing files
clear waym wayt studm studc studt studl studp p bool t stats

% preparation
[waym, studm, studc, studt, studl] = preperation(maps(mapn));
% waym = map with the shortestway to the doors
% studm = who is where
% studc = coordenates of each student
% studp = pressure of each student from up, down, right, left
% studt = time the student have to wait
% studl = which students are left in the room
% stats = pressure, time to wait for every student over time
```
4.8 preperation.m

The preperation.m splits the rawmap into the waym and the studm. For every free field of the waym it computes the number of steps for the shortest way to an exit and the students in the studm get numbered. For every student it reserves a line in studc, studt and studl.

4.9 pressure.m

For all students in the room and for all directions (up down left right),
it looks field per field. Is there a student for which the field in my direction is on the direct way? On direct way means that the number of the field in waym is smaller than the number of the actual position.

Yes: It adds 1 divided by the number of field on the way of the other student to the pressure in this direction

No: It stops.
4.10  border.m
A simple function which returns if (x, y) is in the matrix.

4.11  dulr.m udrl.m
Two short functions containing two polynomials to have access to the neighbour fields in a for-loop.

4.12  picture.m
This function makes a picture of the actual situation as described in chapter 4.6.1.

4.13  stumble.m
Every student who hasn’t to wait gets randomly a time to wait. The probability of stumble is:

\[
\frac{\text{pressure} - \text{time waited}}{e^{\frac{20}{\text{pressure}}}}
\]

The intensity of stumble is a random number between 1 and \( \min\left(\frac{\text{pressure}}{2} + 1, 20\right) \)

4.14  waytime.m
For every student it computes with the functions directway.m and alternivway.m the direct way-time and the alternative-way-time with the connected direction.

4.15  directway.m
Directway.m is a recursive function which sums up the students and their time to wait on the direct way to the exit and returns the shortest way.

4.16  alternivway.m
Alternivway.m is also a recursive function similar to directway.m but now it looks through the alternatives ways and add additionally the step backwards which must be taken.

4.17  statistic.m
This creates the stats variable.
4.18 step1.m

Step1.m checks for every not waiting student if there’s a free adjacent field on the direct way without flip and checks the neighbour fields of the adjacent field for the possible combatant. Flip means that the student flips between the actual and the last position.

```matlab
function [studm, studo, studp, studd] = step1(waym, studm, studo, studp, studd, studl, t)  
    quant = size(studl); % Anzahl Studenten
    for steps = 1:t
        for i = 1:quant
            x = studl(i,1); y = studl(i,2);  
            if studm(x+1,y) == 0 && studo(x-1,y) == 0  
                Stud im Raum, nicht wartend
                if x + y == 1 || x + y == 2
                    study(x,y) = 1; % gleiches Feld
                    field = zeros(1,4);  
                    if x + y == 1
                        field(1,1) = 1;  
                    else
                        field(1,1) = 2;  
                    end
                    if x + y == 2
                        field(1,1) = 1;  
                    else
                        field(1,1) = 2;  
                    end
                    field(1,4) = 1;  
                    field(1,2) = 1;  
                    for j = 1:4
                        [dx, dy] = udir(j);  
                        if x + dx >= 0 && x + dx <= waym &&  
                            y + dy >= 0 && y + dy <= waym
                            field(1,j) = 1;  
                        end
                    end
                    if field(1,1) == 1  
                        if sum(field(1,:)) == 1
                            % Fehle
                            field(1,1) = 0;  
                        else
                            % Mehrere Studenten
                            field(1,1) = 0;  
                        end
                    end
                else
                    % Distanz
                    if x + y == 1
                        field(1,1) = 1;  
                    else
                        field(1,1) = 2;  
                    end
                    field(1,4) = 1;  
                    field(1,2) = 1;  
                    for j = 1:4
                        [dx, dy] = udir(j);  
                        if x + dx >= 0 && x + dx <= waym &&  
                            y + dy >= 0 && y + dy <= waym
                            field(1,j) = 1;  
                        end
                    end
                    if field(1,1) == 1  
                        if sum(field(1,:)) == 1
                            % Fehle
                            field(1,1) = 0;  
                        else
                            % Mehrere Studenten
                            field(1,1) = 0;  
                        end
                    end
                end
            end
            if studp(x+1,y) == 0  
                % Distanz
                if x + y == 1
                    field(1,1) = 1;  
                else
                    field(1,1) = 2;  
                end
                field(1,4) = 1;  
                field(1,2) = 1;  
                for j = 1:4
                    [dx, dy] = udir(j);  
                    if x + dx >= 0 && x + dx <= waym &&  
                        y + dy >= 0 && y + dy <= waym
                        field(1,j) = 1;  
                    end
                end
                if field(1,1) == 1  
                    if sum(field(1,:)) == 1
                        % Fehle
                        field(1,1) = 0;  
                    else
                        % Mehrere Studenten
                        field(1,1) = 0;  
                    end
                end
            end
        end
    end
end
```

Are there adjacent fields without combatants?

```matlab
if sum(field) == 0  
    % Sieht aus wie ein leerer Raum  
    % Wird das Feld ausgelöst
    field(1,1) = 0;  
end
```

Yes: Reserve the field, if there are multiple choose randomly.
No: Does the student have the higher pressure then the combatant to a field?

If yes reserve the field.
This runs four times. So it's sure that no student has longer any pressure advantage to a field.

4.19 step2.m

Step2 is similar to step1, but now no student has longer any pressure advantage. So the student now is allowed to reserve a field is randomly chosen.
This runs as long as every student who could walk has done his reservation.

4.20 step3.m

In step3.m the students on direct way and on alternative way are handled differently. First it checks if the first field of the alternative way fields from wayt are still free.

```
field(1,:) = (sum(field2)) > 0; % freies Feld
for i=1:4
    if field(1,i)
        field(1,i) = round((sum(field2(i,:))+1)*rand(1)+0.5) == 1;
    end
end
```

For the students on the alternative way it checks if the detour still exists and if it's still profitable. So he walks or wait or he goes back on the direct way and if the last field is free he returns to it.
For the students on direct way it checks wayt for an existing profitable detour. If one exists he reserves it, otherwise he waits.
4.21 laststep.m

The function laststep.m handles all students which couldn’t walk in this turn. For the students on the alternativeway (studt(s, 3) = 1) it exchanges the old and new coordinates with the effect, that they can’t walk on the direct way in the next turn because of the no flip in step1.m and step2.m. The other still standing students take the old coordinates as new ones. And all students get set moved.
In the second part the function redraws the map and clears the student on the exit from studl.

```matlab
function [studm, studc, studt, studl] = laststep (waym, studm, studc, studt, studl, t)
    quant = size(studl);
    for s = 1:quant(1,1) % alle nicht bewegten wartend setzen
        if studl(s,mod(t+1,2)+1) % kein Schritt gemacht
            if studl(s,3)
                ax = studc(s,2*mod(t,2)+1); % Koordinaten wechseln
                ay = studc(s,2*mod(t,2)+2);
                studc(s,2*mod(t,2)+1) = studc(s,2*mod(t+1,2)+1);
                studc(s,2*mod(t,2)+2) = studc(s,2*mod(t+1,2)+2);
                studc(s,2*mod(t+1,2)+1) = ax;
                studc(s,2*mod(t+1,2)+2) = ay;
            else
                studc(s,2*mod(t,2)+1) = studc(s,2*mod(t+1,2)+1); % Koordinaten übernehmen
                studc(s,2*mod(t,2)+2) = studc(s,2*mod(t+1,2)+2);
                studt(s,2) = studt(s,2)+1; % Warten ++
            end
        end
        studl(s,mod(t+1,2)+1) = 0; % gesogen
        studl(s,mod(t,2)+1) = 1; % noch im Raum
    end
end
```

In the second part the function redraws the map and clears the student on the exit from studl.

```matlab
studm = zeros(size(studm)); % studm neu einzeichnen
for s = 1:quant(1,1) % noch im Raum
    if studl(s,mod(t,2)+1) % noch im Raum
        if waym(studc(s,2*mod(t,2)+1),studc(s,2*mod(t,2)+2)) == 1 % Exit
            studl(s,:) = [0,0]; studt(s,:) = [-1,t,0]; % Studt verlässt Raum
        else % nicht auf Ausgang
            if studl(s,1) % time to wait -1
                studt(s,1) = studt(s,1)+1;
            end
            studm(studc(s,2*mod(t,2)+1),studc(s,2*mod(t,2)+2)) = s; % einzeichnen
        end
    end
end
```

### 4.22 pictureshow.m

This function sets the colormap to our costume colormap and animates the pictures saved in p to a movie.
5 Simulation Results and Discussion

5.1 Results

In the simulations three parameters have been evaluated in the six different auditoriums:

- Time to leave the room for each person
- Medium pressure on each person during exiting
- Medium risk for stumbling

Auditorium one:

![Histogram of time to exit](image1)

![Histogram of pressure](image2)

![Histogram of stumble](image3)
Auditorium two:

- Time to exit. $\bar{\theta} = 52.0602$
- Pressure. $\bar{\theta} = 0.86971$
- Stumble. $\bar{\theta} = 0.38643$
Auditorium three:

- Time to exit: $\bar{\varphi} = 35.9011$
- Pressure: $\bar{\varphi} = 0.75423$
- Stumble: $\bar{\varphi} = 0.38424$
Auditorium four:

- **Time to exit**: $\bar{\theta} = 32.1195$
- **Pressure**: $\bar{\theta} = 0.63266$
- **Stumble**: $\bar{\theta} = 0.38009$
Auditorium five:

- Time to exit: \( \bar{\theta} = 29.5231 \)
- Pressure: \( \bar{\theta} = 0.48767 \)
- Stumble: \( \bar{\theta} = 0.36592 \)

![Graphs showing distributions of time to exit, pressure, and stumble for 32 students.](image-url)
Auditorium six:

- Time to exit: \( \bar{Y} = 26.5629 \)

- Pressure: \( \bar{Y} = 0.40003 \)

- Stumble: \( \bar{Y} = 0.35468 \)
Comparison of the different auditoriums:

5.2 Discussion

5.2.1 Influence of exit door distribution

The comparison between rooms with doors in the back only and with rooms with doors in the back and in the front shows that the evacuation time can be drastically reduced by a wider distribution of exit doors. We can see that the average exit time drops from room one (two doors) to room three (four doors) from 60 to 36 time units. The same result we can see in the rooms two (two doors) and four (four doors) with exit time 52 time units to 32 respectively.

Conclusion: This outcome was more or less expected, but it shows, that it is essential that all doors are accessible and operational for a fast evacuation. Since the model is assuming that everyone knows the nearest exit. It is imperative that also in reality everyone is familiar with all possible exit routes. During the discussion of this result we recognized that both of us are not yet familiar with the exit routes in our lecture rooms.
5.2.2 Influence of Aisle width

Our simulations show that by increasing the aisle width by taking out 10 percent of all seats the exiting time over proportionally was reduced by 13.5 percent. Our model is not taking in to account that the increase of the aisle width by one chair will free more space than for one exiting person. Therefore in reality the gain would be even higher.

Conclusion: The contrary of increasing the aisle width would be an overload of the auditorium with more people than available seats. In this case we would expect a drastic increase of the exiting time. It might be worthwhile to consider weather in some auditoriums a couple of seats could be abandoned without loosing much capacity.

5.2.3 Influence of different room layouts

In our simulation of the auditorium number five and six we can see that a larger number narrower exiting roots reduces the exiting time distinctly. But we also see that this reduces also the capacity significantly.

Conclusion: For the design of new auditoriums omitting seat rows would be of interest but the reduced capacity might turn out problematically.

5.2.4 Model weaknesses

We are fully aware of the fact that our model is not depicting the reality. For example we assigned in model the same space for a seat as for an exiting person. In reality the density of people in the aisles is much higher than the density of seated people. Furthermore in reality people act more egoistically than in our model. We assumed that people with higher pressure would get priority independent of there personalities. An other weakness in our model is stumbling of exiting people. The model implies that a stumbling person gets up again, where in reality a stumbling person might gets furthers to stumble or could even cause panic. Despite the weaknesses of the model we believe that the drawn conclusions held some value.
6 Summary and Outlook

We explored with a model implemented in Matlab the emergency evacuation of an auditorium. It showed us the influences of different room layouts, exit door distributions, and aisle width for the evacuation. It is important that an auditorium has enough and distributed exit doors for an evacuation to be fast and safe. With a slight reduction in seat numbers leading to wider aisles the evacuation time can be reduced overproportionally. Increasing the number of exit ways while reducing their width is also reducing the evacuation time but reduces the capacity significantly.

To improve our model one could think of implementing also different person densities and different behaviors (personalities) of exiting people.
7 References

8 Code

8.1 Simulation.m

```matlab
% Simulate an emergency evacuation
% choose a map 0-6
mapn = 1;

% clears possible existing files
clear waym wayt studm studc studt studl studp p bool t stats

% preparation
[waym, studm, studc, studt, studl] = preparation(maps(mapn));
% waym - map with the shortest way to the doors
% studm - who is where
% studc - coordinates of each student
% studp - pressure of each student from up, down, right, left
% studt - time the student have to wait
% studl - which students are left in the room
% stats - pressure, time to wait for every student over time

% Loop until nobody rests in the room
bool = true;
t = 1;
while(bool)
    % pressure
    studp = pressure(waym, studm, studc, studt, studl, t);
    % picture
    p(:, :, t) = picture(waym, studm, studp);
    % stumble
    studt = stumble(studp, studt, studl, t);
    % waitime
    wayt = waitime(waym, studm, studc, studt, studl, t);
    % statistic
    stats(:, :, t) = statistic(studp, studt, studl, t);
```
% step 1
[studm, studc, studt, studl] = step1(waym, studm, studc, studp, studt, studl, t);
% step2
[studm, studc, studt, studl] = step2(waym, studm, studc, studt, studl, t);
% step unweg
[studm, studc, studt, studl] = step3(waym, studm, studc, studt, studl, t);
% make final step: restlicher twaited t1, twait -i, ausgang löschen
[studm, studc, studt, studl] = laststep(waym, studm, studc, studt, studl, t);

t=t+1;
% noch wer da?
bool = sum(studl(:, mod(t+1, 2)+1));
end
% stats auswerten
stats(:, :, t) = statistic(studp, studt, studl, t);
clear i t bool mapn
8.2 preparation.m

```matlab
function [waym, studm, studo, studt, studl] = preparation(mapraw)

% separate the waymap and the studmap informations
waym = mod(mapraw, 2) .* mapraw;
studm = (mapraw - waym) ./ 2;

% finds the shortest way to a exit
i = 1;
bool = true;
dim = size(waym);

while bool
    for l = 1:dim(1,1)
        for r = 1:dim(1,2)
            if waym(l, r) == 1
                if border(l-1, r, dim) && waym(l-1, r) == 0 % auf der Karte und gleich 0
                    waym(l-1, r) = i+1;
                    bool = false;
                end
                if border(l+1, r, dim) && waym(l+1, r) == 0
                    waym(l+1, r) = i+1;
                    bool = false;
                end
                if border(l, r-1, dim) && waym(l, r-1) == 0
                    waym(l, r-1) = i+1;
                    bool = false;
                end
                if border(l, r+1, dim) && waym(l, r+1) == 0
                    waym(l, r+1) = i+1;
                    bool = false;
                end
            end
        end
    end
end
end
```
8.3 step1.m

```matlab
function [studm, studc, studp, studt] = step1(waym, studm, studc, studp, studt, studl, t)
    quant = size(studl); % Anzahl Studenten
    for steps = 1:4
        for i = 1:quant(1,1)
            if studl(i, mod(i+2,2)+1) == 0 % Stud im Raum, nicht wartend
                x = studm(i,2*mod(i+2,2)+1); % t ungerade 1; gerade 3
                y = studm(i,2*mod(i+2,2)+2); % t ungerade 2; gerade 4
                field = zeros(1,4); % mögliche Mittarbeiter down up left right
                for i = 1:4
                    [dx, dy] = sdir(i); % down up left right
                    if not(studm(x+dx,y+dy)) && waym(x+dx,y+dy) == 0 % weg(i,x,y) && weg(i,x+dx,y+dy) == 0 && not(weg(i,x+dx,y+dy))) && studm(x+dx,y+dy) == 0
                        x = studm(i,2*mod(i+2,2)+1) && y = studm(i,2*mod(i+2,2)+2); % t ungerade 1; gerade 3
                        y = studm(i,2*mod(i+2,2)+2) && x = studm(i,2*mod(i+2,2)+1); % t ungerade 2; gerade 4
                        field(i,1) = 1;
                        for j = 1:4
                            [dxx, dyy] = sdir(j); % down up left right
                            if i == j && border(x+dxx,y+dyy) && studm(x+dxx,y+dyy) == 0
                                x = studm(i,2*mod(i+2,2)+1) && y = studm(i,2*mod(i+2,2)+2); % t ungerade 1; gerade 3
                                y = studm(i,2*mod(i+2,2)+2) && x = studm(i,2*mod(i+2,2)+1); % t ungerade 2; gerade 4
                                field(i,1) = 1;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
```
end

if sum(field) % gibt es freie Felder
  for i = 1:n % Gibt es Felder ohne Konkurrenz
    if field(i,i) % Feld frei?
      if sum(field(1:i)) % there are other students
        field(i,i) = 0;
      end
    end
  end
end

if sum(field) % es gibt solche Felder ohne Konkurrenz
  chosen = round(sum(field)*rand(1)+0.5); % random auswahl
  i = 1;
  while(chosen)
    if field(i,i)
      if chosen == 1
        [dx,dy] = du1(i);
        studt(s,2) = 0;
        studt(s,3) = 0;
        studc(s,2*mod(t,2)+1) = x+dx;
        studc(s,2*mod(t,2)+2) = y+dy;
        studl(s,mod(t+1,2)+1) = 0;
        studl(s,mod(t,2)+1) = 1;
        studm(studc(s,1),studc(s,2)) = s;
        studm(studc(s,3),studc(s,4)) = s;
        chosen = chosen-1;
      else
        chosen = chosen-1;
      end
    end
    i = i+1;
  end
else % gibt es Felder wo grösster Druck?
field(1,:) = (sum((field2')) > 0); % freies Feld
for i = 1:4 % größerer Druck
    if field(1,i)
        [dx,dy] = duiz(i);
        for j = 1:4
            if field2(1,j) % steht da jemand
                [ddx,ddy] = udrl(1);
                field(1,i) = field(1,i)*(studp(s,1) >...
                studp(studm(x+ddx,y+ddy-dy),1)));
                % = 0, wenn nicht größerer Druck als alle andern
            end
        end
end
if sum(field) % es gibt ein solches Feld
    chosen = round(sum(field)*rand(1)+0.5); % random auswahl
    i = 1;
    while(chosen)
        if field(1,i)
            if chosen == 1
                [dx,dy] = duiz(1);
                studp(s,2) = 0;
                studp(s,3) = 0;
                studp(s,2*mod(t,2)+1) = x+dx;
                studp(s,2*mod(t,2)+2) = y+dy;
                studl(s,mod(t+1,2)+1) = 0;
                studl(s,mod(t,2)+1) = 1;
                studm(stud(s,1),stud(s,2)) = s;
                studm(stud(s,3),stud(s,4)) = s;
                chosen = chosen-1;
            else
                chosen = choosen-1;
            end
        end
    end
end
8.4 *step2.m*

```matlab
function [studm, studc, studt, stud1] = step2 (waym, studm, studc, studt, stud1, t)

quant = size(stud1);
bool = true;

while (bool)
    for a = 1:quant(1,1)
        if studc(a, mod(t+1,2)+1) && not(studc(a,1)) && im Raum, nicht warten
            x = studc(a, 2*mod(t+1,2)+1); % t ungerade 1; gerade 3
            y = studc(a, 2*mod(t+1,2)+2); % t ungerade 2; gerade 4
            field = zeros(1,4); % mögliche Felder DOWN LEFT RIGHT
            field2 = zeros(1,4); % mögliche Mitbesterer DOWN LEFT RIGHT
            for i = 1:4
                [dx, dy] = durl(i); % DOWN LEFT RIGHT
                if not(studm(x+dx, y+dy)) && waym(x+dx, y+dy) < waym(x, y) &&...
                    waym(x+dx, y+dy) == -1 && not(x+dx == studc(a, 2*mod(t,2)+1))...
                    && y+dy == studc(a, 2*mod(t,2)+2))
                    % gibt es Felder frei, auf Weg, kein Flip
                    field(i,1) = 1;
                    bool = false; % es gibt noch ein mit möglichem freiem Feld
                    for j = 1:4
                        [ddx, ddy] = durl(j); % DOWN LEFT RIGHT
                        if 1 == j && border(x+dx+ddx, y+dy+ddy, size(waym))
                            % nicht zugängliches Feld, noch im Raum
                            if studm(x+dx+ddx, y+dy+ddy) &&...
                                studc(studm(x+dx+ddx, y+dy+ddy), mod(t+1,2)+1) && not(studc(studm(x+dx+ddx, y+dy+ddy), mod(t+1,2)+1))...
                                && waym(x+dx+ddx, y+dy<waym(x+dx+ddx, y+dy+ddy))
                                % Mitschüler?, noch nicht gelaufen, nicht warten, auf Weg
                                field2(i, j) = 1; % anderer
                        end
                    end
                end
            end
        end
        if sum(field) % es gibt ein freies Feld
            for i = 1:4 % Gibt es Felder ohne Konkurrenz
                if field(i,1) % Feld frei?
                    if sum(field2(i, 1)) % there are other students
                        field(i, 1) = 0;
                    end
                end
            end
        end
        if sum(field) % es gibt solche Felder ohne Konkurrenz
            chosen = round(sum(field)*rand(1)+0.5); % random auswahl
            if chosen
                if field(i,1)
                    if chosen == 1
                        [dx, dy] = durl(i);
                        studc(a, 1) = 0;
                        studc(a, 3) = 0;
                        studc(a, 3*mod(t+1,2)+1) = x+dx;
                        studc(a, 3*mod(t+1,2)+2) = y+dy;
                        studc(a, mod(t+1,2)+1) = 0;
                        studc(a, mod(t+1,2)+2) = 1;
                        studc(studm(a,1), studc(a,2)) = 0;
                        studm(studc(a,3), studc(a,4)) = 0;
                    end
                end
            end
        end
    end
end
```
chosen = chosen-1;
else
  chosen = chosen-1;
end

i = i+1;

else % wer darf ziehen?
  field(1,:) = (sum(field2') > 0); % freies Feld
  for i=1:9
    if field(1,i)
      field(1,i) = (round((sum(field2(1,:))+1)*rand(1)+0.5) == 1);
    end
  end

  if sum(field) <= 5 gibt ein solches Feld
    chosen = round(sum(field)*rand(1)+0.5); % random auswahl
    i = 1;
    while(chosen)
      if field(1,i)
        if chosen == 1
          [dx,dy] = dulu(1);
          stude(s,2) = 0;
          stude(s,3) = 0;
          stude(s,2*mod(t,2)+1) = x+dx;
          stude(s,2*mod(t,2)+2) = y+dy;
          stude(s,mod(t,2)+1) = 0;
          stude(s,mod(t,2)+1) = 1;
          stude(s,mod(t,2)+1) = stude(stude(s,1),stude(s,2));
          stude(stude(s,3),stude(s,4)) = s;
          chosen = chosen-1;
        else
          chosen = chosen-1;
        end
      end
      i = i+1;
    end
  end

  bool = not(bool);
end
for $s = 1:quant(1,1)$
    if stud(s,mod(t+1,2)+1) && not(stud(s,1)) % im Raum, nicht wartend
        $x = stud(s,2*mod(t+1,2)+1)$; % t ungerade 1: gerade 3
        $y = stud(s,2*mod(t+1,2)+2)$; % t ungerade 2: gerade 4
        for $i = 1:4$ % Felder immer noch frei?
            if wayt(s,2+i)
                [dx,dy] = durl(i);
                wayt(s,2+i) = not(studm(x+dx,y+dy));
            end
        end
    end
    if stud(s,3) % schon auf Umweg
        if sum(wayt(s,3:6)) % es gibt Umweg, erstes Feld frei
            chosen = round(sum(wayt(s,3:6))*rand(1)+0.5); % random auswahl
            i = 1;
            while(chosen)
                if wayt(s,2-i)
                    if chosen == 1
                        [dx,dy] = durl(i);
                        studt(s,3) = 1;
                        studt(s,2*mod(t,2)+1) = x+dx;
                        studt(s,2*mod(t,2)+2) = y+dy;
                        studt(s,mod(t-1,2)+1) = 0;
                        studt(s,mod(t,2)+1) = 1;
                        studt(studt(s,1),studt(s,2)) = s;
                        studt(studt(s,3),studt(s,4)) = s;
                        chosen = chosen-1;
                    else
                        chosen = chosen-1;
                    end
                end
                i = i+1;
            end
        end
    else
        if wayt(s,2) == -1 || (wayt(s,1)+studt(s,2)) <= wayt(s,2)
            % Umweg gibt es nicht mehr, Umweg lohnt sich nicht mehr
            studt(s,3) = 0;
            if not(studm(studt(s,2*mod(t,2)+1),studt(s,2*mod(t,2)+2)))
                % letzte Position noch frei
                studt(s,2) = 0;
                studt(s,mod(t+1,2)+1) = 0;
                studt(s,mod(t,2)+1) = 1;
                studt(studt(s,1),studt(s,2)) = s;
                studt(studt(s,3),studt(s,4)) = s;
            end
        end
    end
end
else % noch nicht auf Umweg
    if sum(wayt(s,3:6)) &lt; (wayt(s,1)+stud(s,2)) &gt; wayt(s,2)
        % es gibt Umweg, erstes Feld frei, Umweg lohnt sich
        chosen = round(sum(wayt(s,3:6))/rand(1)+0.5); % random Auswahl
        i = 1;
        while(chosen)
            if wayt(s,2+1)
                [dx,dy] = dulr(1);
                studt(s,3) = 1;
                studc(s,2*mod(t,2)+1) = x+dx;
                studc(s,2*mod(t,2)+2) = y+dy;
                stuld(s,mod(t+1,2)+1) = 0;
                stuld(s,mod(t,2)+1) = 1;
                studm(studc(s,1),studc(s,2)) = s;
                studm(studc(s,3),studc(s,4)) = s;
                chosen = chosen-1;
            else
                chosen = chosen-1;
            end
            i = i+1;
        end
    end
end
8.6 alternivway.m

```matlab
function [tmin, nw] = alternivway(x, y, waym, studm, studt)
    % ungefährezeit auf einem kürzesten alternativen weg
    tmin = -1;
    nw = 0; % 0 kein Weg
    field = (-1)*ones(1, 1);
    for i = 1:4
        [dx, dy] = udrl(1);
        if border(x+dx, y+dy, size(waym)) && waym(x+dx, y+dy) == -1 && i == j
            % im Raum keine Wand und kein Flip
            if waym(x, y) < waym(x+dx, y+dy) % auf Umweg
                [t, nw] = alternivway(x+dx, y+dy, round(i/2)*2-mod(i+1, 2), waym, studm, studt);
                % Permutation 1<->2, 3<->4
                if nw && (not (studm(x+dx, y+dy))) || studt(studm(x+dx, y+dy), 3))
                    % Keine Sackgasse, kein Student der nicht auf Umweg
                    nw = 1;
                    if studm(x+dx, y+dy) && studt(studm(x+dx, y+dy), 3) && Student auf umweg
                        field(1, 1) = 1+stud(studm(x+dx, y+dy), 1)+waym(x+dx, y+dy)-waym(x, y)+c;
                        tmin = field(1, 1);
                    else
                        field(1, 1) = waym(x+dx, y+dy)-waym(x, y)+t;
                        tmin = field(1, 1);
                    end
                end
            else
                tmin = 0; % Umweg beendet, es ist keine Sackgasse
                nw = 1;
                return
            end
        end
        if tmin == -1
            for i = 1:4 % kleinstes t
                if field(1, i) == -1
                    if field(1, i) < tmin
                        tmin = field(1, i);
                    end
                end
            end
        end
    end
end
```
8.7  border.m

```matlab
function [bool] = border(x, y, dim)
% X,Y in the matrix -> true
% else -> false
  bool = true;
  if x < 1 || x > dim(1,1) || y < 1 || y > dim(1,2)
     bool = false;
end
```

8.8  dulr.m

```matlab
function [dx,dy] = dulr(i)
% down up left right
  dx = -2/3*i^3+11/2*i^2-83/6*i+10;
  dy = 2/3*i^3-9/2*i^2+83/6*i-5;
end
```

8.9  udrl.m

```matlab
function [dx,dy] = udrl(1)
% up down right left
  dx = 2/3*i^3-11/2*i^2+83/6*i-10;
  dy = -2/3*i^3+9/2*i^2-53/6*i+5;
end
```

8.10  stumble.m

```matlab
function [studt] = stumble(studp,studt,studl,t)
  quant = size(studl);
  for s = 1:quant(1,1) % all students
     if studl(s,mod(t-1,2)+1) < 0
        if not(studl(s,2)) % not in exam
           if sum(studp(s,1)) > 30
              smax = 20;
           else
              smax = sum(studp(s,1))/2+1;
           end
        end
        studt(s,1) = round(smax*rand(1)+5)*(rand(1) <= exp((sum(studp(s,1))-studl(s,2))/(20/exp(1)));
     end
  end
end
```
8.11  picture.m

```matlab
function [p] = picture(waym, studm, studp)
    dim = size(waym);
    p = zeros(dim);
    for l = 1:dim(1,1)
        for r = 1:dim(1,2)
            if abs(waym(l,r)) == 1 % Wand oder Ausgang
                p(l,r) = waym(l,r);
            elseif studm(l,r) == 0; % Student
                p(l,r) = 2+sum(studp(studm(l,r),:));
            end
        end
    end
    p = p+2; % 1 Wand, 2 Boden , 3 Türe, 4-9 Studenten
end
```

8.12  statistic.m

```matlab
function [stats] = statistic(studp, studt, studl, t)
    quant = size(studl); % Anzahl Studenten
    stats = zeros(quant);
    for s = 1:quant(1,1)
        if not(studl(s,mod(t+1,2)+1)) % nicht mehr im Raum
            stats(s,:) = [-1 -1];
        elseif
            stats(s,1) = sum(studp(s,:));
            stats(s,2) = studt(s,1);
        end
    end
end
```
function [tmin] = directway(x,y,waym,studm,studt) % ungefähre Zeit auf direktem Weg

tmin = -1;
if waym(x,y) == 1
  tmin = 0;
  return
end
field = (-1)*ones(1,4);
for i = 1:4
  [dx,dy] = dult(i);
  if waym(x+dx,y+dy) < waym(x,y) && waym(x+dx,y+dy) ~= -1 && ~waym(x+dx,y+dy)
    field(i,1)=0;
  end
end
for i = 1:4
  [dx,dy] = dult(i);
  if not(field(:,1)) \ % auf Weg
    if studm(x+dx,y+dy)
      field(1,1) = (studm(x+dx,y+dy) +1+directway(x+dx,y+dy,waym,studm,studt));
      tmin = field(1,1);
    else
      field(1,1) = directway(x+dx,y+dy,waym,studm,studt); \ %min = field(1,1);
    end
  end
end
for i = 1:4 \ % kürzester Weg
  if field(1,1) == -1
    if tmin > field(1,1);
      tmin = field(1,1);
    end
  end
end
end
8.14 laststep.m

```matlab
function [studm, studc, studt, studl] = laststep (waym, studm, studc, studt, studl, t)
    quant = size(stud); %
    for s = 1:quant(1,1) % alle nicht bewegten wartend setzen
        if studl(s,mod(t+1,2)+1) % kein Schritt gemacht
            if studt(s,3)
                ax = studc(s,2*mod(t,2)+1); % Koordinaten wechseln
                ay = studc(s,2*mod(t,2)+2);
                studc(s,2*mod(t,2)+1) = studc(s,2*mod(t+1,2)+1);
                studc(s,2*mod(t,2)+2) = studc(s,2*mod(t+1,2)+2);
                studc(s,2*mod(t+1,2)+1) = ax;
                studc(s,2*mod(t+1,2)+2) = ay;
            else
                studc(s,2*mod(t,2)+1) = studc(s,2*mod(t+1,2)+1); % Koordinaten übernehmen
                studc(s,2*mod(t,2)+2) = studc(s,2*mod(t+1,2)+2);
                studt(s,2) = studt(s,2)-1; % Warten --1
            end
            studl(s,mod(t+1,2)+1) = 0; % gezogen
            studl(s,mod(t,2)+1) = 1; % noch im Raum
        end
    end
    studm = zeros(size(stud)); % studm neu einzeichnen
    for s = 1:quant(1,1)
        if studl(s,mod(t,2)+1) % noch im Raum
            if waym(studc(s,2*mod(t,2)+1),studc(s,2*mod(t,2)+2)) == 1 % Exit
                studl(s,:) = [0,0]; studt(s,:) = [-1,t,0]; % Stud verlässt Raum
            else % nicht auf Ausgang
                if studt(s,1) % time to wait --1
                    stud(l,s) = studt(s,1)-1;
                end
                studm(studc(s,2*mod(t,2)+1),studc(s,2*mod(t,2)+2)) = s; % einzeichnen
            end
        end
    end
end
```
function [studp] = pressure(waym, studm, studc, studi, t)
    quant = size(studi); % quantity of students
    studp = zeros(quant(1,1),4); % set p to 0
    for s = 1:quant(1,1) % for all students
        if studl(s, mod(t+1,2)+1) % t ungrade 1; grade 2 % for all students left
            for i = 1:4 % for all directions
                x = studc(s,2*mod(t+1,2)+1); % t ungrade 1; grade 3
                y = studc(s,2*mod(t+1,2)+2); % t ungrade 2; grade 4
                [dx,dy] = udrl(i); % up down right left
                bool = true;
                while(bool) % until there's nobody in this direction
                    if waym(x+dx,y+dy) == -1 % wall: break
                        break
                    else if not(waym(x+dx,y+dy)) % no person: break
                        break
                    else if (waym(x,y) - waym(x+dx,y+dy)) >= 0 % student doesn't want to go to this field
                        break
                    else
                        n = 0;
                        for j = 1:4 % number of field student could go
                            [ddx,ddy] = udrl(j);
                            if waym(x+ddx,y+ddy) == -1 % keep Wand
                                n = n+(waym(x+ddx,y+ddy) - waym(x+ddx,y+ddx)) > 0);
                            end
                        end
                        studp(s,i) = studp(s,i)+1/n;
                        end
                        x = x+dx;
                        y = y+dy;
                        end
        end
    end
end
8.16  pictureshow.m

```matlab
function[] = pictureshow(p)
    steps = size(p);
    steps(1,3);  %steps = steps(1,3);
    steps=1;
    i = 1;
    axis off;
    grid off;
    cmap = zeros(64,3);
    cmap(1:29,:) = [0.4000 0.4000 0.4000;
                    0.8000 0.8000 0.8000;
                    1.0000 1.0000 1.0000;
                    0 1.0000 0;
                    0.2500 1.0000 0;
                    0.5000 1.0000 0;
                    0.7500 1.0000 0;
                    1.0000 1.0000 0;
                    1.0000 0.8500 0;
                    1.0000 0.7000 0;
                    1.0000 0.5500 0;
                    1.0000 0.4000 0;
                    1.0000 0.3000 0;
                    1.0000 0.2000 0;
                    1.0000 0.1000 0;
                    1.0000 0 0;
                    0.9000 0 0;
                    0.8000 0 0;
                    0.7000 0 0;
                    0.6000 0 0;
                    0.5400 0 0;
                    0.4800 0 0;
                    0.4200 0 0;
                    0.3600 0 0;
                    0.3000 0 0;
                    0.2400 0 0;
                    0.1800 0 0;
                    0.1200 0 0;
                    0.0600 0 0];
    colormap(cmap);
    while(true)
        image(p(:,:,mod(i,steps)+1))
        i = i+1;
        pause(.2);
    end
end
```
We disclaim here to plot the map matrices. In the chapter 4.3 you can see how we implemented the different maps.