Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

D I P L O M A R B E I T

Comparison of Mathematical Models and Development of a Hybrid Approach for the Simulation and Forecast of Influenza Epidemics within Heterogeneous Populations

ausgeführt am Institut für

Analysis und Scientific Computing

der Technischen Universität Wien

unter Anleitung von Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

durch

Štefan Emrich

Parkulica, 32

A-7304 Großwarasdorf / Veliki Borištof

Datum

Unterschrift

Zusammenfassung

Im Rahmen dieser Diplomarbeit wird ein mathematisches Modell zur Simulation und Vorhersage von Influenza Epidemien entwickelt werden.

Die klassischen Methoden für die Modellierung und Lösung solcher Aufgabenstellungen sind Differentialgleichungssysteme. Diese erweisen sich in mancher Hinsicht jedoch leider als eingeschränkt. Vor allem bei der Betrachtung von heterogenen Populationen und räumlichen Komponenten werden diese Systeme extrem kompliziert und über die Maße komplex. Am Anfang dieser Arbeit werden deshalb die Potentiale von alternativen Ansätzen – im Speziellen zellulären Automaten und agentenbasierten Systemen – untersucht.

Die Analyse dieser Methoden gliedert sich in zwei Teile. Einerseits in den theoretischen Teil, in welchem die Methoden verglichen und ihre Stärken bzw. Schwächen ermittelt werden. Andererseits in den praktischen Teil, einschließlich der Untersuchungen des Verhaltens der Implementierungen beider Methoden. Zu diesem Zweck wurde eine einfache SIR-Epidemie sowohl mit der einen als auch der anderen Methode modelliert.

Bestärkt von den Ergebnissen der Evaluierung der Ansätze, wurde das hybride Modell aufgesetzt. Da die genaue Parametrisierung des Modells gute und reale Daten voraussetzt, und aufgrund der Tatsache, dass die weitere Entwicklung des Modells sehr von der Qualität dieser vorhandenen Daten abhängt, wird auch die Datenlage genauer betrachtet. Am Ende werden Experimente, durchgeführt am neu entwickelten Modell, analysiert.

Die Ergebnisse dieser Experimente unterstützen die ursprüngliche Motivation des hybriden Ansatzes und ermutigen zu weiteren Untersuchungen desselben. Schließlich wird ein Ausblick auf die mögliche Entwicklung von und notwendigen Bedingungen für so einen Modellansatz gegeben.

Abstract

In the course of this thesis a hybrid mathematical model for the simulation and prediction of influenza epidemics is going to be established.

The classic methods applied for modelling such epidemics used to be ODE-Systems but unfortunately these systems are limited in some respect. They become particularly complicated and complex beyond limit when observing heterogeneous populations and spatial components. Thus the potential of alternative approaches – namely cellular automata and agent based systems – is analyzed in the beginning of this work.

Analysis of these methods was split into two major parts. The first one being the theoretical one in which the methods were compared in order to locate their respective strengths and weaknesses. The second part being the practical analysis including behavior of the implementations, for this a simple SIR epidemic was modelled with each approach.

Backed by the findings of the analysis of the methods, the final hybrid model was set up. Since accurate parametrization of models requires reliable and authentic data for validation purposes, and due to the fact that a further development of the model strongly depends on the quality of this data, this issue is also covered in this thesis. Finally the experiments with the newly created model are analyzed.

The outcome of the experiments backs the basic motivation to use a hybrid approach and encourages further investigation of it. Thus an outline for future possibilities of and necessities for such modelling approaches is sketched.

Contents

List of Figures

Zahvalne riči

Po skoro sedam ljeti študijuma na Bečkom tehničkom sveučilišću se kanim gonc jako zahvalit kod mojih starji ki su mi ovo uopće omogućili. Naravno i kod mojih bratov, družicov i tovarušev ki su mi pokratili čas u dalekom Beču. Zvanatoga se kanim i srdačno zahvalit kod mojega profesora Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felixa Breiteneckera, a posebno kod Günthera Zaunera i Nikia Poppera za njev trud i njevu pomoć pri pisanju ovoga djela.

Lipa hvala!

Am Ende meines Studiums an der TU Wien will ich mich bei meinen Eltern die mir dies überhaupt ermöglicht haben ganz herzlich bedanken. Danke auch meinen Brüdern, Freundinnen und Freunden, die sich darum gesorgt haben, dass mir während der Zeit nicht langweilig wurde. Und natürlich auch herzlichen Dank an meinen Professor Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker und vor allem meinen zwei Betreuern Günther Zauner und Niki Popper für ihre Mühe und Hilfe beim Verfassen dieser Arbeit.

Danke!

At the end of my studies at the Vienna University of Technology I want to say "thank you" to my parents for offering this opportunity to me. Of course I also want to thank my brothers and friends for making this time worth wile. And last but not least a big "thank you" to my professor Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker and especially my two tutors Günther Zauner and Niki Popper for their effort and help at this thesis.

Thank you!

Chapter 1

Introduction

1.1 Motivation for Modelling Diseases [1]

Looking at the effects, diseases and particularly epidemic or even pandemic outbreaks did have on mankind throughout history, it may be hard to grasp the full scale of pain, tragedy and suffering that they carried along. The toll of human lives is astronomic, easily surpassing that of all wars. The figures for historic epidemics are estimates and of course vary, but even the conservative assumptions are shocking.

The black death epidemic of the 14^{th} century can be held accountable for about 20-25 million deaths in Europe alone. Europe at that time had a population of estimated 80-100 million, meaning that up to a third of the population was carried off by it. Although latest studies assume that it was not only the plague but a combination of a few diseases that were circulating at that time.

Only diseases that the human immune system cannot cope with are potentially life threatening. This was the case for the plague in middle ages and it was also discovered fast by the military. The first known case of biological warfare being the siege of Genoa in 1346, when infectious corpses were thrown over the city walls to cause death and panic among the city's population.

But even seemingly "harmless" diseases proved to be extremely lethal. The Aztecs did not have any anti bodies against smallpox. Accidentally introduced by the Spanish army, smallpox killed approximately half of the Aztecs population, whereas the invaders did not suffer noteworthy.

In modern times specialists for chemical warfare were and are struggling to create new or mutated viruses against which humans do not have any immunity protection.

[1] Main references for this section: [L4, L8, L24]

This seems increasingly irrational if looking at the efforts that were undertaken to eradicate the natural threats to humans and with the recent threat that such diseases might fall in the hands of terrorists.

Although some diseases that used to be common causes for death have been extinct[2], some others were just pushed back beyond the frontiers of western countries and forgotten there. A famous example being malaria which used to be common also in the Mediterranean parts of Europe but was conquered by simply draining swamps and thus destroying the malaria mosquitoes habitat. In underdeveloped regions on the other hand, it is still cause of 1.5 - 2.7 million deaths per year, with estimated 90% of the sickened living in Africa.

Yet another example for usually harmless diseases capable of devastating global pandemics is influenza. Its most famous – and lethal – outbreak being the "Spanish flu" at the beginning of the last century (1918 to 1920). With estimates varying from 20 up to 40 million its death toll is astonishing. Although the low hygiene standards, food shortages and poor economic situation at the end of World War I were contributing to its effects, this strain was remarkably dangerous. Unlike normal strains it mainly killed adults and older children instead of elderly people.[3]

Since the Spanish flu two pandemic outbreaks of influenza have occurred, namely the "Asian flu" (1957 to 1958) and the "Hong Kong flu" (1968 to 1969) pandemics, killing an estimate of 1 - 1.5 million and 0.75 - 1 million humans respectively. A known pattern for future outbreaks of influenza pandemics does not exist. Not even the necessary conditions for a virus to become pandemic are know. It is uncertain when and what kind of pandemic strain we will face. What we can be certain of is that a pandemic outbreak will occur. The challenge we face is to be prepared for this case as good as possible.[4]

The economic damage of such pandemic outbreaks, even if nonlethal, is tremendous. It not only sums up enormous costs for medical treatment, but also brings along massive loss of working time and thus high disease related follow-up costs. And even pandemics not affecting humans but animals, do have the potential to severely bash our economy, e.g. through necessary mass killing of poultry due to avian influenza.

In contrast to the social circumstances a 100 years ago, people from developed countries and regions of the world, nowadays live in solid hygienic conditions and have access to medication as well as sufficient food. But this by far does not mean that the

[2]With the most famous extinct disease being smallpox.

[3][L8, L27]

[4][L13, L24]

danger of a global pandemic is banned – on the contrary! The process of globalization and increased mobility are making it possible for diseases to spread as fast as never before in human history, as shown by SARS and the avian flu (H5N1).

Therefore national administrations, as well as international organizations need to have efficient plans to respond to outbreaks and pandemic diseases as fast as possible. And all scientific help available should be used to support these efforts, including biologists, medical scientists, epidemiological research workers and mathematicians.

In Austria the ministry for health and women is in charge of the creation and maintenance of such plans[5]. These kind of pandemic plans are created in cooperation with the WHO[6]. This is necessary because of the global nature of pandemics.

The next section is going to give an overview of the work in this field done by mathematicians so far.

1.2 The Historical Development of Modelling[7]

The wish to understand diseases and epidemics is probably as old as humans are facing them. The first known effort to mathematically describe diseases dates back to ancient Greece, and Hippocrates[8] who described illnesses and epidemics.

In order to successfully describe or model the pattern of diseases, certain levels of mathematical methods as well as of understanding the disease itself, are necessary. This is the reason why, from Hippocrates on, it took centuries before progress was made. Due to lack of understanding the diseases and their causes this progress was primarily made by statistical analysis of medical data. John Graunt and William Petty were one of the first when they analyzed the London Bills of Mortality in the 17^{th} century.

After this beginning it again took another 200 years until at the end of the 19^{th} century adequate mathematical methods have been developed and mankind started to explore bacteriology and understand the physical basis. With it the necessary basis for the development of mathematical theories for the spread of epidemics was established.

Although already before this some remarkable progress was made by analysis of spatial and temporal spread of epidemics. By such an analysis for example John Snow was able to locate one specific water pump as the source for the 1854 cholera epidemic in London. At that time this was a remarkable finding, since it was believed that foul

[5] see cite.[L8]

[6] World Health Organization

[7] Main references for this section: [L2, L4, L9, L11, L12, L14]

[8] 460-370 BC, source [W11]

air was the cause of cholera – instead of polluted water!

Another very important step in epidemic research was made by the efforts to understand the waxing and decline of epidemics, as done by William Farr in 1840. The outcome was then used to effectively fit normal curves to existing epidemic data, in Farrs case smallpox. Later he tried to predict the outbreak of rinderpest with such a model, although with only moderate success. Only the shape of both the predicted and the observed epidemic were similar.

Further improvement of prediction methods came at the beginning of the 20^{th} century, with a deeper understanding of the epidemic mechanisms through bacteriological research. Hamer realized that the course of an epidemic strongly depends on the number of susceptible and infected individuals as well as on the contact rate between these groups. The outcome were deterministic models which allowed to predict the number of infected at any given time by knowing the initial settings of susceptible and infected individuals plus the specific attack- and recovery-rates. With these models it was possible to understand the periodic patterns of epidemic occurrences.

These deterministic models were refined and made famous by Kermack and McKendrick. Often literature does refer to the ODE-system[9]

$$\begin{aligned} \dot{S}(t) &= -\beta S(t)I(t) \\ \dot{I}(t) &= \beta S(t)I(t) - \gamma I(t) \\ \dot{R}(t) &= -\gamma I(t) \end{aligned} \tag{1.1}$$

as Kermack and McKendrick model. This is not correct, since system (1.1) is only "derived from special case $\bar{A} = \beta e^{-\gamma\tau}$ ..." of the Kermack McKendric model "...

$$\dot{S}(t) = S(t) \int_0^\infty \bar{A}(\tau)\dot{S}(t-\tau)d\tau \tag{1.2}$$

with $S(t)$ denoting the (spatial) density of susceptibles at time t and $\bar{A}(\tau)$ equals the expected infectivity of an individual that became infected τ units of time ago".[10] Nevertheless the ODE-system in (1.1) is used very often for classical modelling of SIR epidemics[11].

Subsequently it was noticed that all these deterministic models exhibit damped harmonic waves, whereas disease data, although oscillating, does not show this effect.

[9] abbr. of Ordinary differential equation system

[10] cite. [L10]

[11] Alternative approaches for modelling such a SIR epidemic will be presented in chapter 3.

The next generation of models was connected to better and more available data. With the focus partially shifting from large scale models towards smaller groups the need for models somehow influenced by probability became increasing. Such a model was first presented by McKendrick in 1926 but did not find much attention. A fully stochastic model, that did receive the necessary attention, was presented five years later (1931) by Greenwood. The models of this time started to integrate latent and incubation periods of the disease pattern and lead to chains of binomial distributions.

From the 1940ies on the development started to pick up speed, at the same time some influential advances were made. Approaches were combined, for example using deterministic models under some circumstances and changing to stochastic ones to examine striking details exhibited by the deterministic models. To get a feeling for the speed of development it is interresting to look at the published work of that time.

At the end of the 1950ies "the total number of references to mathematical work in the literature was at about 100"[12]. This number doubled in the following 10 years and another 8 years later it did lie at 500. The further development was supported by the dawn of computers that set in during the 1950ies and 60ies. In the late 60ies control of infectious diseases, to keep the disease levels at a "tolerable"[13] level, came into focus and studies were conducted with methods derived from operations research.

1.3 New Ways - A Hybrid Approach

Generally the above mentioned methods have been improved and specialized in many ways. Methods were found to simulate multiple diseases or the spatial spread of diseases. Nevertheless all methods do have in common that they are based on differential equations (PDE[14] or ODE systems) and modelling idealized (e.g. homogeneous) populations. When trying to apply the methods to inhomogeneous populations (e.g. different age groups with varying attack rates) or spatial distributions (e.g. different densities for rural and urban areas), complexity explodes in a way that it cannot be handled.

[12] cit. [L4]

[13] In this context "tolerable" is a very delicate definition. Since the specification of this level of course depends on the point of view. From a financial point of view it can mean to aim for the least costs; from an affected persons perspective it will most likely mean the minimum of suffering. For most nations this usually means best protection of the population combined with highest possible cost efficiency.

[14] PDE – Partial differential equation

Thus, in order to simulate epidemics with respect to spatial structures and inhomogeneous populations, other approaches need to be found. Two alternative methods that are of interest for such tasks are agent based systems and cellular automata. These methods will be presented and thoroughly covered in chapter 2 of this thesis. A possible application of them will be presented in chapter 3 by applying them to a SIR epidemic, that is usually described by ODE-systems of the form (1.1).

As one will see in the chapters 2 and 3, cellular automata prove to be very efficient tools when correctly implemented on computers. Whereas agent based systems offer a very flexible structure, desirable for describing complex constructs (e.g. a mixed population consisting of different age groups with specific social behavior and specific infection risk).

The tools and approaches described in section 1.2 do rely only on little data – usually the number of infected and healthy (susceptible) individuals within a population. They further need infection/recovery rates to describe the development of the system from the current point on. These rates can of course not be measured in nature and thus estimation, calculation or combination of both is necessary. In addition the outcome of this approach is based on the assumption of homogeneous populations.

The methods that we will look at are different in this respect. Since at the present time a wide range of data is available (e.g. demographic data, medical data, statistics on the housing situation, etc.) one can take advantage of this and try using the data to set up simple, local rules in order to describe the spread of diseases. A model based on such rules would have the advantage, that it can be fitted with data collected and used to forecast the likely development of an epidemic. In addition the information received from the model is based on a more realistic demographic structure.

Thus a combination of cellular automata and agent based systems seems promising for modelling epidemics — with respect to an inhomogeneous demographic structure, spatially distributed and moving individuals, while still remaining computationally efficient. Such a hybrid model[15] will be presented in chapter 4 of this thesis. Finally the outcome of the hybrid approach as well as of the simulation with the model will be presented in part 4.3.

[15]The created model is capable of simulating a 100-day period for a population of 100,000 inhabitants in less than 9 hours on a standard desktop computer.

Chapter 2

The Modelling Concepts Used

This chapter deals with the modelling tools used in the subsequent model. At first cellular automata (abbreviation CA) will be introduced together with a short historical overview of the development. We will then focus on a special kind of CA namely Lattice Gas Cellular Automata (abbr. LGCA).

Agent-based systems (abbr. ABS) are also referred to as multi-agent systems (MAS) and will be covered in 2.2. The end of the chapter is marked by a short summary and a comparison of the two modelling approaches.

2.1 Cellular Automata - CA

Cellular automata were first introduced in the late 1940ies by John von Neumann and Stanislaw Ulam, thus are not a completely new method. The idea was to simulate biological processes such as reproduction and evolution in a spacial environment, by simple means – the cells.[1] The difference between cellular automata and the methods previously used for modelling purposes lies within the difference of the approach.

Differential equation models (ODE models) can be classified as "top-down" approach. This means one tries to describe the system as a whole by describing the global processes and transitions. Whereas cellular automata on the other hand are classified as "bottom-up" approach. "Bottom up" characterizes systems that are aiming to model the global behavior, in other words the *macroscopic* point of view through description of the microscopic correlations.

The idea behind cellular automata was partially influenced by the Turing machine. A Turing machine is an automaton with only three operations (read, write, move on)

[1] see [L15, L19]

that is capable of manipulating information stored on a tape, manipulation is based on simple rules. All basic arithmetic operations can be implemented on these machines and with them other, more complex operations and programs can be produced.

Originally developed in only one dimension CA soon also became two-dimensional. One-dimensional cellular automata amongst others have been used for simulation of traffic and traffic jams. Later three dimensional Ca's proved to be interresting for simulation of gas- or liquid-distributions in space. Since the CA used in the subsequent models will be solely two-dimensional only those will be the covered here[2].

A major boost to the popularity of cellular automata was the publication of John Conway's game "Life" (1970) and succeeding publications (in the journal *Scientific American*) dealing with "Life".[3]

The definition of classical two-dimensional cellular automata is a very simple one. The following four points are sufficient to describe a CA:

1. Geometry of cells:
   The CA consists of equal (geometrical) cells which are arranged in a (regular) lattice.

2. Neighborhood definition
   A cell neighborhood is defined for the automaton. The neighborhood determines which cells are influencing each other. This neighborhood is valid for all cells during the whole runtime.

3. Cell states
   A finite number of (discrete) states is defined. Every cell can assume one of those states. The state of the cell is subject to change. The transition between states is determined by the transition rules.
   The states of the cells are updated synchronously (after discrete time steps) for all cells.

4. Transition rules
   These rules describe how the cells change its states. The rules only depend on the state of the neighboring cells and on the state of the cell itself. One set of transition rules is valid for the whole CA over the whole runtime.

[2]For further reading on especially one dimensional CA see [L35].
[3]see [L17, L34]

The definition of cellular automata may vary throughout literature. This definition was adopted from [L34] and is not the only possible definition.

The cells of classical cellular automata usually have a rectangular lattice and square cells. For different reasons, e.g. simulation of liquid-/gas-dynamics, other structures may be used. In case of the mentioned examples a hexagonal lattice is used. We will describe this kind of CA more detailed in Section 2.1.

The above mentioned criteria are not always fulfilled – there are exceptions in which it makes sense to use (slightly) different criteria. For example the transition rules may also contain a probabilistic component.

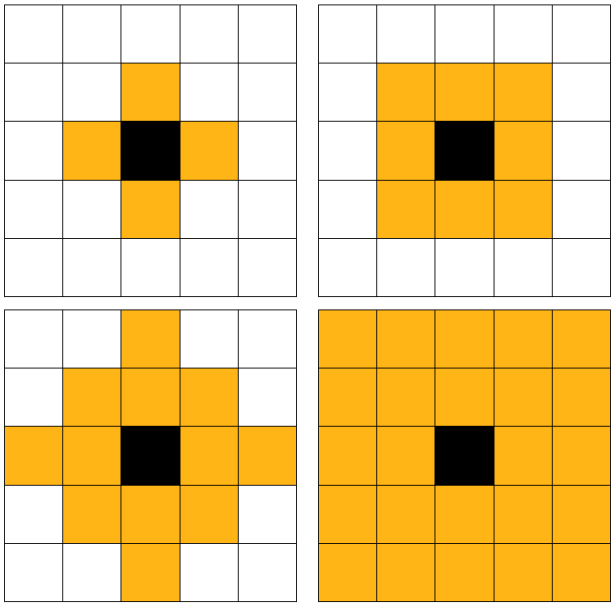

Figure 2.1: *von Neumann (left) and Moore (right) neighborhoods of range 1 and 2*

Let us assume a cellular automaton with Moore neighborhood (see Fig.2.1) of range 1. And let n be the number of possible states that a cell can take on in accordace to our transition rule Φ. As stated above Φ is updating the state of every cell at every time step. The new value of the cell $c_{i,j}$ with location i, j only depends on the state of the cell itself and the states of its neighbors. Thus we can express the change of $c_{i,j}$ after one timestep $t \to t+1$ through

$$c_{i,j}^{(t+1)} = \Phi\left(c_{i,j}^{(t)}, c_{i-1,j-1}^{(t)}, c_{i-1,j}^{(t)}, c_{i,j+1}^{(t)}, c_{i,j-1}^{(t)}, c_{i,j+1}^{(t)}, c_{i+1,j-1}^{(t)}, c_{i+1,j}^{(t)}, c_{i+1,j+1}^{(t)}\right). \tag{2.1}$$

To give an example we will look at Conway's famous cellular automaton based game "Life". It is based on a rectangular lattice of which every cell can assume one of two

states: 'dead' or 'alive'. The transition rules are defined as follows (by neighbor we mean a live neighboring cell):

- A live cell with two or three neighbors will stay alive, else it will die.
- A dead cell with exactly three neighbors becomes alive.

Despite their simple definition cellular automata are capable of producing very complex systems. And due to their local structure they are able to represent geographic distributions. They can be used to describe ecological-, social- as well as immune-systems (e.g. pollution of groundwater, segregation in neighborhoods, growth of cancer)[4] only to name a few.

A very early example for the use of cellular automata were the findings of Thomas C. Schelling[5]. Although exclusively done by hand he showed how slight personal preferences or attitudes can change a whole society in a very dramatic way. The concrete example was the segregation of neighborhoods. With cellular automata he simulated different demands that people have regarding their neighbors skin color. One would think, that if a person would demand only little more than a third of his neighbors being of the same color mixed neighborhoods would form. Schelling showed that the opposite happens, the system still ends in segregation!

Another big advantage of cellular automata lies within their computational simplicity. Since they consist of only a few simple rules and a very rigid structure they can easily be implemented very efficiently in computer systems. Although they of course need a certain amount of memory to save the cell related information[6]. Because of their locally bound transition rules they are appropriate for parallel computation. Thus making them more and more interesting with parallelization rapidly developing in today's (personal) computers. A very handy property is, that they are "unconditionally numerically stable by construction"[7].

A disadvantage of classical cellular automata is the fixed cell. Making it hard to model travelling individuals/particles. In Schellings model described above, a cell changes its state depending on the number of neighbors of same/different color. The state of the cell represents the color of the inhabitant. Thus the population is not

[4] [L3, L29, L23]

[5] see [L29]

[6] The consumed memory to store a 10 x 1,000,000 matrix with double precision entries in MATLAB lies at approx. 75 MB (amount can be reduced by using other data types).

[7] cit. [L34], page 36

really travelling, but only jumping from one cell to another one that better suits the demands.

If we want to model individuals that are truly travelling in space an enhancement to our cellular automata needs to be found. In the following section we will be looking at lattice gas cellular automata which do prove to be a usefull extension for our purposes.

Lattice Gas Cellular Automata - LGCA

Because of its features and properties CA are also of interrest for the simulation of physical problems and models. This was the basic idea out of which lattice-gas cellular automata evolved in the early 1970ies.

To simulate large systems or structures by using a smaller scale model one needs to obey the law of dynamic similarities (e.g. wind tunnels). The law also needs to be applied when simulating real flows within lattice-gas cellular automata.

One can describe the flow of incompressible liquids by the Navier-Stokes [8] equation

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\nabla)\mathbf{u} = -\nabla P + \nu\nabla^2\mathbf{u} \tag{2.2}$$

with the continuity equation $\nabla\mathbf{u} = 0$ and ∇ being the nabla operator, $\mathbf{u}$ the flow velocity, P the kinematic pressure, ν the kinematic shear viscosity and ∇^2 the laplacian operator.

The Navier-Stokes equation is derived from the conservation laws of energy, mass, momentum and angular momentum. It is nonlinear in the velocity $\mathbf{u}$ and thus in general analytically not solveable – save for a few exceptions. This question is strongly connected to the question of boundary conditions.

To compute the results for the equation numeric methods need to be applied. The area that is engaged with finding such numerical solutions for the Navier-Stokes equation is called computational fluid dynamics (CFD). Lattice-gas cellular automata and lattice Boltzmann methods are, amongst others, a possible way to simulate the behavior of flows in fluids or gases.

In order to simulate the Navier-Stokes equation with cellular automata it is necessary, that they "hold corresponding conserved quantities"[9]. The first proposal for simulation of liquid flows with cellular automata was made by Hardy, de Pazzis and

[8] Named after Claude-Louis Navier (1785-1836) and George Gabriel Stokes(1819-1903).

[9] cit. [L34], page 36

Pomeau in 1973, their CA named HHP-LGCA after their initials. Unfortunately the CA did not yield the Navier-Stokes equation.

After this attempt it took another 13 years, before Frisch, Hasslacher and Pomeau discovered that a third condition needs to be fulfilled: In addition to conservation of mass and momentum the cellular automata also needs to hold sufficient symmetry. The demand for symmetry was met by a triangular lattice, the cellular automaton named "FHP lattice gas cellular automata" after its developers.

The main features of the FHP-LGCA can be summed up by following 8 points [10]:

1. It has a regular lattice with *hexagonal symmetry*.

2. *Nodes* (*sites*) are linked to its six nearest neighbors (located all at the same distance) by edges/vectors.

3. The vectors $\mathbf{c}_i$ linking nearest neighbor nodes are called *lattice vectors* or *lattice velocities*

$$\mathbf{c}_i = \left(\cos\frac{\pi}{3}i, \sin\frac{\pi}{3}i\right), \quad i = 1, \ldots, 6$$

   with $|\mathbf{c}_i| = 1$ for all i.

4. A cell is associated with each link at all nodes.

5. Cells can either be empty of occupied by at most one particle (*exclusion principle*).

6. All particles have the same mass and are indistinguishable.

7. The evolution in time proceeds by an alternation of *collision* $\mathcal{C}$ and *streaming/propagation* $\mathcal{S}$:

$$\mathcal{E} = \mathcal{S} \circ \mathcal{C},$$

   where $\mathcal{E}$ is called *evolution operator*.

8. The collisions are strictly *local* -only particles of a single node are involved.

With these properties one can better understand the difference between FHP-LGCA and classical CA.

[10] cit. [L34], chapter 3.2

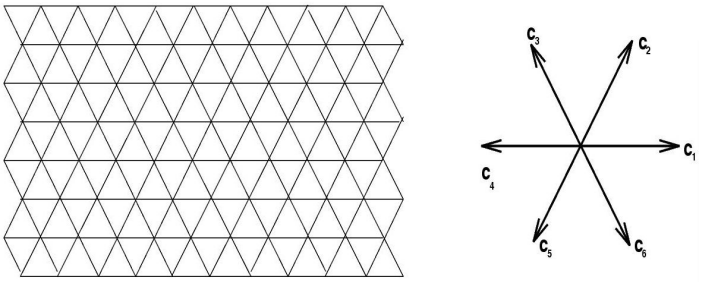

Figure 2.2: *The lattice (left) and lattice vectors (right) of an FHP LGCA*

As stated before, the FHP has an underlying triangular lattice and thus possesses hexagonal symmetry. Within this lattice all nodes are connected with its six nearest neighbors via edges/vectors (see Fig. 2.2). And the cells are not defined by surrounding edges, but are located at each node at all links. In addition a *lattice vector* $\mathbf{c}_i$ is assigned to every cell. The vector $\mathbf{c}_i$ connects the node with its nearest neighbor in direction i. Since the time steps in CA are (usually) always 1 we can also refer to $\mathbf{c}_i$ as *lattice velocity*. The particles all do have the same mass, in order for the collisions to be mass and momentum conserving. This lets us interpret our *lattice vector/lattice velocity* $\mathbf{c}_i$ in yet another way: as the momentum of the particle.

As mentioned, collisions play an essential role in LGCA. According to the set of rules used, one can distinguish between three FHP variants, namely *FHP-I*, *FHP-II* and *FHP-III*. The smallest set of rules consists of 2 and 3 particle collisions (without spectator-/rest-particle) and distinguishes the *FHP-I* variant (for this set of collisions see Fig. 2.3). The other two types consist of additional collision rules (more colliding particles and a possible spectator-/rest-particles).

The CA used in the subsequent model is a slight modification of the *FHP-I*. Since the model is not aiming to reproduce physical dynamics but a social structure the amount of interaction/collisions is sufficient. In addition the *FHP-I* is also faster in terms of runtime than a *FHP-II* or *FHP-III* with a full implementation of all collision-rules. A difference to the FHP-LGCA described above, is the uniqueness of the particles. In order to represent the (unique) individuals of the system it is necessary that they are distinguishable within the automaton. To realize this requirement each particle represents an individual via its individual key. The boundaries of the automaton are set to be periodic -allowing a better mixing of the individuals than reflexive border conditions.

If we take a looking at the properties of cellular automata and accept modifications

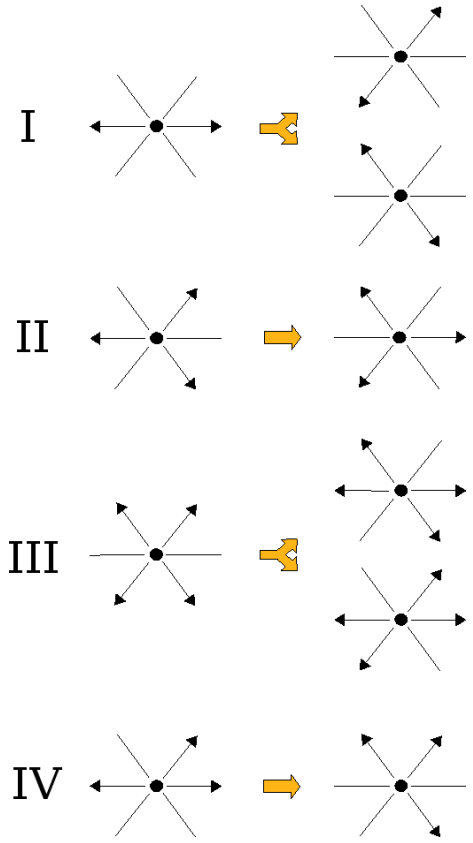

Figure 2.3: *FHP collision rules without rest particle. Arrows represent the occupied cells, lines the cells empty. Left hand side constellations before collision, right hand side after. Only rules I through III are used for FHP-I automata.*

of their strict rules, one can guess, that they do have hidden potential. Especially if considering their local nature and the kind of data available at the present time it seems promising to use CA for modelling inhomogeneous systems.

Another promising method for modelling of complex systems is the agent based approach which we will deal with in the following section.

2.2 Agent-based Systems

Agent-based systems, or multi-agent systems (abbr. ABS, MAS) are fairly new modelling tools. They evolved out of research into the field of distributed artificial intelligence (DAS), a side branch of artificial intelligence research. Like cellular automata the ABS approach is categorized "bottom up". The focus of agent-based systems lies on individual agents, as the name implies. Because of their complexity, MAS opened a

broad field of new possibilities on the modelling sector. Some even say, that they were "the breakthrough in computer modelling in social sciences"[11]. But at the same time their complexity is also a limitation, since simulation runs require a high amount of computing power.

Nevertheless MAS are currently being used for a wide variety of applications; server-client simulations, socio-economic models or simulations of transport robots in factories only to name a few. It may at first be quite surprising to find such a broad variety in a discipline this young. But after taking a closer look one can see, that the flexible definition of agent-based systems encourage such diverse implementations.

A shortcoming, resulting from the youth of this approach, is the inconsistency or even lack of common definitions. For example there is no common answer to the question "What exactly is an agent?".

A constantly present danger when talking about or describing agents is to use terms normally associated with humans. Examples are words like *intelligence*, *memory* or *autonomy* which are tempting to overestimate or misjudge the capabilities of these computer systems. Keeping this in mind we can take a shot at sketching agent-based systems. Looking at it from the ABS perspective we can use following definition[12]:

An agent is a computer system *situated* in an environment. In addition it has the capabilities to *flexibly* and *autonomously* act in this environment in order to reach its (predefined) objectives/goals.

This leaves us with three terms that need to be refined:

- *Situated* in our case means that the agent is interacting with its environment. The agent is capable to receive input from the surrounding (generally via sensors) and can also manipulate it to some extent.

- The definition of *autonomy* needs to be handled with care, as explained above -we are talking about a pre-programmed computer system. Thus it is satisfactory if the agent can reach decisions without (human) interaction.

- *Flexibility* is required in multiple ways. Firstly one demands that the system is operating and acting in reasonable time. Secondly the agents are not to be solely reactive but goal-oriented or in the best case anticipating. And thirdly agents

[11] cit. [L18]
[12] adapted from [L22]

may have the capability to communicate or interact with other agents and/or real humans.

Of course there may be differing definitions of the term *agent*. And naturally these definitions are subject to change - different implementations may require different emphases. What still needs to be defined is the multi agent system.

MAS are characterized by:

- agents with a limited point of view (incomplete information[13] and/or problem solving capabilities),
- the absence of global system control,
- decentralized data and
- asynchronous computation of the agents.

Judging by this we can come to a similar conclusion for agent based systems as for cellular automata. Both simulation methods do represent the bottom-up approach. The agents within MAS are usually locally anchored. Thus information and data describing local conditions (e.g. medical data from physicians, sizes of child care facilities, etc.) can be very well implemented by the use of agents. Whereas the classical top-down approach does not leave room to utilize this kind of data.

2.3 Comparison of the Methods

Both of the presented approaches of course do have their strengths and flaws. Some of them may have been already mentioned, some not. In this section we will try to get a concluding overview of their similarities and differences, their strengths and weaknesses.

The Smallest Unit

As we already know CA are based on cells, or in case of LGCA on travelling particles. These cells or particles do not hold a big amount of information and are very limited in terms of movement as well. During one time step they can usually advance only one distance unit.

[13] Incompleteness with respect to the whole system with all related information, e.g. total number of population, knowledge of conditions elsewhere, etc.

Agents within MAS on the other hand, can hold many properties and are physically not bound. They can "jump" within the system from one place to another without (major) limitations. Further, agents can have a list of "partners", "friends", or other kinds of "contacts". Depending on the implementation they may even have the ability to communicate with each other.

Lapse of Time

Time in CA is a strictly discrete factor – all cells are updated synchronously, thus all changes happen synchronously. This also means, that within cellular automata there are times with no changes happening (e.g. after an update step has been executed). Such a moment may prove to be beneficial if trying to save the content of the automata at given points in time.

In agent-based simulations time may either be discrete or virtually continuous, depending on the implementation and the desired qualities of the system. Changes are usually triggered by events and thus may happen at any time during the simulation. Compared to cellular automata this means that event handling within MAS requires much more attention, and thus computing power. In addition to the overhead produced the risk of errors and mistakes is increasing.

Spatial structure

Cellular automata are built of cells which are usually identical and symmetrical, although at least one study[14] exists, which shows that the shape of the cells is not essential for the outcome of some simulations. This might be interesting since many sociologists are of the belief that complex social structures cannot be modelled by simple and symmetrical grids.

Agent-based systems do not necessarily need a spatial grid or structure, although most of them do have one. Just as before MAS are also more flexible when it comes to the underlying grid. Often coordinates are assigned to agents, by which the agent then is positioned within the system. It is of course possible to create a structure within MAS that resembles a cellular grid and can be used to emulate cellular automata[15].

[14] see [L20]

[15] This can be done by using fixed agents, positioned on a lattice and defining a neighborhood that corresponds to the CA neighborhood.

State Transitions

Within cellular automata the state transitions are controlled by fixed rules. They are applied simultaneously to all cells once every time step. Thus the automaton changes its face step by step.

Agent-based models do not have universally valid rules, changes are triggered by events. These events depend on certain parameters which are continuously monitored. If their value reaches a critical point the event is launched and a change takes place. And even these critical points may be subject to change, depending on time, spatial position, property of agent, etc.

Sphere of Influence

The sphere of influence within CA is equivalent to the defined neighborhood, thus it is of a very local nature. Only very few cells of the direct surrounding are within reach.

Agents in multi-agent systems usually also do have a quite manageable sphere of influence, but unlike in CA it is not limited to the direct surrounding. For example an agent can have contacts all over the system – which can even represent a whole country. With every "visit" of its contacts the agent is affecting only one individual – but in very distant regions of the system.

2.4 Summary

Altogether one could interpret multi-agent systems as an extension of cellular automata[16]. Cellular automata are much more standardized and have stricter rules than multi-agent systems, leaving more tolerance when programming ABS. But this freedom comes at a price. The more possibilities a system offers, the more parameters it has to use. The process of parameter optimization becomes increasingly harder with increasing number of parameters to define, as well as instabilities within the system are much more difficult to locate.

To simulate more realistic, although still simplified, populations and their complex socio-economic behavior, ABS will be the modelling tool of choice. On the other hand there are efforts to loosen up the strict definition of CA to bring them closer towards MAS – with interesting effects![17]

[16] This is for example done in [L26].

[17] see [L28] for details

But there are still many areas in which a simple structure is sufficient to describe the system's behavior, and with it cellular automata as modelling tools (e.g. diffusion of gases or liquids in three dimensional space). In this case it does not make sense to use unnecessarily complicated systems for modelling.

To put it in the words of CA-luminary Stephen Wolfram:

> "Cellular automata are sufficient simple to allow detailed mathematical analysis, yet sufficient complex to exhibit a wide variety of complicated phenomena." *Wolfram, 1983*

2.5 The Simulation Environment

2.5.1 MATLAB

In chapter 4 a hybrid system for the modelling of influenza epidemics will be set up. This modell is programmed in MATLAB®, which is a software package and programming language for numerical computing. Often referred to as computer algebra package.

Its name derives from MATrix LAboratory and this already describes the classic strenghts of the language: matrix and vector manipulation. It was originally developed to provide easy access to matrix software in the late 1970ies. It features a wide variety of add-ons called *toolboxes* that enhance its capabilities. An interresting toolbox for simulation and modelling purposes would be the state event toolbox.

Over the years MATLAB evolved into a simulation environment. A big contribution to this was the add-on Simulink (firstly released in 1990), which nowadays is a graphical block-tool for modelling, simulation and analysis of dynamic systems.

Nowadays MATLAB is capable of handling a wide spectrum of applications (e.g. signal and image processing, communications, control design, test and measurement, financial modelling and analysis, and computational biology). It is possible to write programs, functions and subfunctions in order to fit ones needs. The latest versions of MATLAB are even capable of symbolic computation – and thus of solving ODE systems.

The program does offer numerous interfaces in order to link it with other programming languages such as C, C++, Java, Fortran, etc. This in combination with the optimization algorithms allows MATLAB to be used as optimization tool for external as well as for internal models. It also offers a wide variety of 2D and 3D graphic

functions for visualization purposes.

A question that might arise is why MATLAB was used for programming the model and not an object oriented language. MATLAB in its current versions did extremely improve matrix manipulation in terms of runtime. Several features are even written in assembler code. Compared to not optimized C++ code MATLAB is even faster for matrix operations. Since the most operations used within the subsequent program are matrix operations and MATLAB is capable of handling them very conveniently, it has been the program of choice.

With all its features and several runtime enhancements MATLAB currently is a fully-fledged simulation and development environment capable of solving most mathematical and modelling problems.

2.5.2 AnyLogic

In chapter 3 we will use the simulation environment AnyLogic® to set up an epidemic model using the agent based approach. AnyLogic 5.5 is an object oriented programming tool that provides a wide variety of features used in multi agent systems, such as message passing, state charts and -events, setting up of object classes and many more. AnyLogic is based on JAVA and by this offers users the possibility to add JAVA code at any place of the program.

On one hand this is a very comfortable feature, including all benefits of object oriented programming, on the other hand it brings along the "classic" JAVA flaw of poor runtime. Another positive aspect of the JAVA based environment is the possibility to export models as JAVA applets for easy use and/or distribution and integration on web sites.

Besides agent based modelling AnyLogic can also be used to program and combine Discrete Events and System Dynamics approaches. In addition it also offers an easy possibility to add animation to models. Further it is possible to modify model properties during runtime (e.g. change infection rates, speed of agents, etc.).

In AnyLogic it is very difficult to export data from simulations and experiments for further analysis and interpretation. This turns out to be a big disadvantage of AnyLogic, especially because the program does not offer much possibilities for analysis.

AnyLogic does offer, similar as MATLAB, various toolboxes to expand the possibilities of the program. These also include some visualization possibilities, although they are by far not as extensive as MATLAB's. The storage of data is also by far not as convenient as with MATLAB's matrices that allow numerous possibilities for analysis.

Chapter 3

A Simple SIR-Epidemic

The final model is aiming to simulate an epidemic outbreak of influenza within a realistic population. This means that there are numerous things that need to be considered, starting from influenza-related information (e.g. mean duration of illness) over social factors (e.g. demographic structure) up to counter measures that shall be tested.

For the final model to be efficient and provide realistic results all these factors need to be summed up for a successful implementation. We will start with a simple SIR model to study some general effects and properties of the system as well as of the methods. These findings are going to be helpful when setting up the full scale model in chapter 4.

3.1 The CA Approach

Since we want to model a population of moving (human) individuals we are going to use a slightly modified *FHP-I* version. In a SIR-model individuals can either be susceptible, infected or recovered.[1] Only a susceptible individual can become infected, infected individuals recover with a given probability and recovered individuals are immune to infection for the rest of the simulation. Later we will also look at a variation that allows the recovered to go back to susceptible state. Let $S(t)$ be the number of susceptible individuals, $I(t)$ the number of infected and $R(t)$ the number of recovered individuals at time t.

Further we are going to make a few assumptions in order to keep the model simple. Our population size is stable during the simulation meaning no deaths or births occur.

[1] Often the SIR model is extended onto a SEIR model in which individuals are not immediately infectious. They pass through a phase in which they are infected but not yet infectious.

Individuals become infective at the time of infection and stay so until recovery. Infection and recovery are implemented by fixed rates (infection r and recovery a) that are applied after every time step.

Alltogether this is quite far from being realistic, it is sufficient to start with. In chapter 4 we will refine most of these settings to receive an improved model.

3.1.1 Basic Implementation

The cellular automaton is implemented as matrix. The first and second dimension resemble the lattice (the rows and the nodes in every row), the "third dimension" is reserved for the content of the cells at each node. Thus the matrix is only 6 entries deep. The boundary condition is periodic to allow better mixing of the particles. For easier implementation of the boundary conditions the length and height of the lattice are equal l. Thus one can visualize the automaton as a cube (actually a hexahedron) with edges of lengths l, l and 6.

In MATLAB one can produce such a matrix (with all zero entries) easily by using following command:

```
% ca_size being the length ''l'' of one side of the automaton
ca_size = l;
ca_matrix = zeros(ca_size,ca_size,6,'int32');
```

The population information is stored in a look up table – a two dimensional matrix. The first row representing the unique key of the individual, the second its health status[2]. The cellular automata is filled randomly with particles, which are the unique keys of the modelled particles (which will later on become agents). All two and three particle collision rules[3] (without spectator) are applied to the CA.

The infection routine is called up at every time step t after propagation and collision of the CA. The routine checks all nodes (i, j) , $i, j = 1 \dots l$ for infected $I_{i,j}(t)$ and susceptible $S_{i,j}(t)$ particles at time t. If at least one infected and one susceptible particle are found at a node ($I_{i,j}(t) > 0$ and $S_{i,j}(t) > 0$) the susceptible particles are put at risk of infection with probability r.

The recovery routine is executed subsequently. First a search for all infected particles is carried out, thereafter they are cured with probability a. This procedure is not realistic, for example it does allow for a particle to become infected and recover during one single time step. But changing the order of the procedures would bear the

[2]This realization was chosen because of the expansions that will be applied. For the goals of the current implementation it would be sufficient if the states of the individuals would serve as particles.

[3]See Fig. 2.3 for an illustration.

probability of the initial infection dying out before even making a single step. With this implementation the duration of the disease is determined purely probabilistic. Theoretically allowing a particle to stay infected until infinity.

This is a matter that needs to be changed for the final implementation.

3.1.2 Stability Analysis

In the current model one can already identify a few parameters that can be modified and experimented with. Naturally these are the infection and recovery rates r and a. Secondly these are the initial sizes of the sub-populations of susceptible $S(t = 0) = S_0$, infected I_0 and recovered R_0 individuals. And the last parameter left for modification being the density of individuals within the automaton.

One should also keep in mind that fairly small automata – with only few particles inside – offer only poor results. The reason for such a behavior is easy to guess: If one particle represents a relatively big part of the population, random effects have much more weight and thus more influence on the total behavior. These effects are shown nicely in Fig. 3.1.[4]

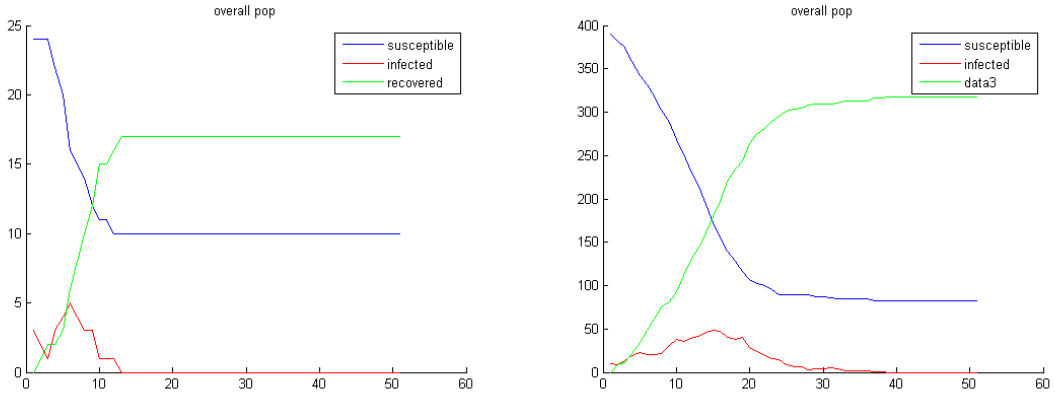

Figure 3.1: *Standard runs with CA size 4 x 4 (left) and size 15 x 15 (right); (density being approx. 30% leading to 27 and 400 particles; x-axes: time, y-axes number of particles)*

For the following analysis we will run several experiments with our automaton. The initial standard configuration of the system is displayed in the following table:

[4]Simulation configurations: standard config. with 27 particles (3 infected) respectively 400 (10 infected).

Standard parameters used for subsequent CA simulations		
Infection rate	...	0.6
Recovery rate	...	0.3
Susceptibles	...	990
Infected	...	10
Recovered	...	0
Density approx.	...	30%

The approximate nature of density derives from the fact, that the

$$\text{side length} = \sqrt{\frac{\text{number of particles}}{6 * \text{density}}} \, .$$

Hence (usually) only an approximate density can be achieved within automata, since the side length needs to be an integer. Further all MATLAB simulations are run ten times and averaged in order to reduce the deviation caused by the stochastic nature of infection, recovery and movement. This method of averaging is called "Monte Carlo" method.[5] The effects and necessity for such an averaging are shown in more detail in section 3.4.

In the first experiments we will vary the infection rate from 20 to 100% (see Fig. 3.2). When looking at these results one must keep in mind that an infection rate of 100% means that all susceptible particles that are at a node together with an infectious neighbor will be infected in that time step. If such a neighbor does not exist infection cannot occur.

This also explains the difference in the graphs when changing the recovery rate (as done in Fig. 3.3). As one can easily notice the recovery rate has a stronger influence on the overall system than the infection rate[6], at 40% recovery the infection already dies out very fast.

A very interresting method to analyze the mutual effects of two parameters on a system is to run several (hundred) simulations with varying settings and compare the results. In this case we will gradually change the infection and recovery rate from 0 to 100%.

In Fig. 3.4 following plots are displayed the time until the epidemic dies out (top left), the number of infections per simulation (top right), maximum of infected particles

[5] see [L6]

[6] This can easily be explained by the fact that no neighbor is required for recovery.

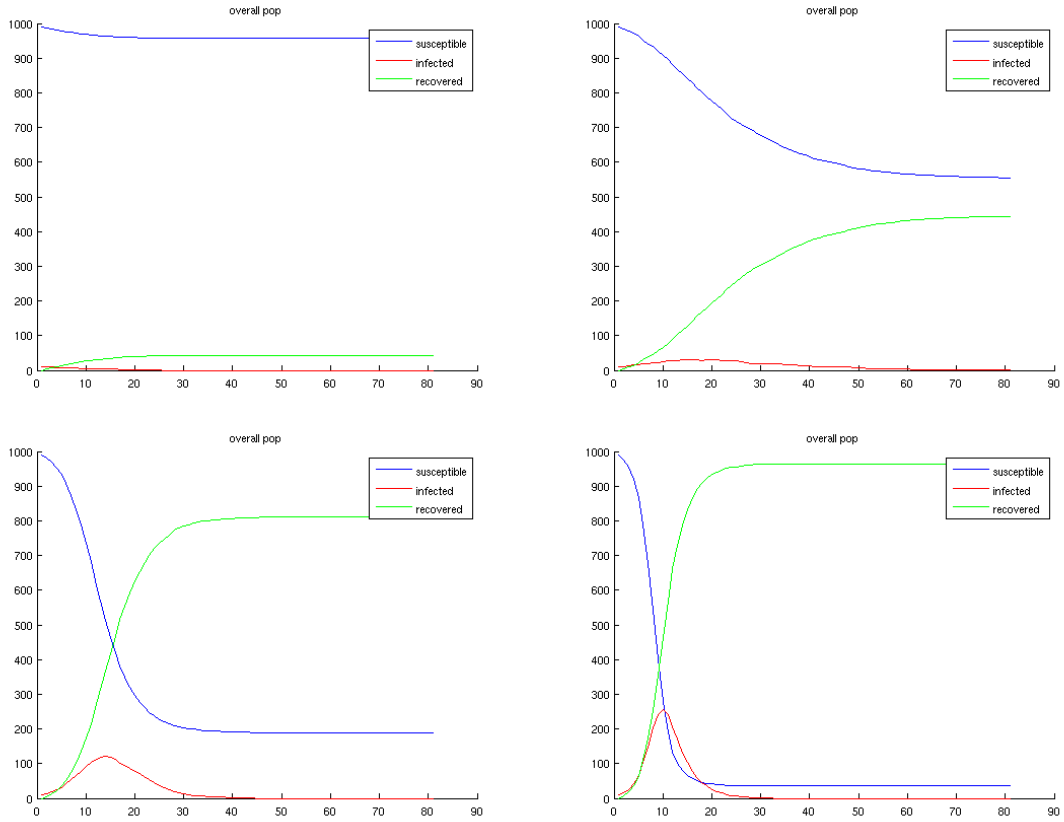

Figure 3.2: *First row: infection rate 20% and 40%, second row: standard simulation = 60% and 100% (x-axis: units of time; y-axis: number of particles)*

during one timestep (bottom left) and the number of susceptible particles at the end of the simulation (bottom right). The runtime of the simulations was limited to 50 units, since most epidemics die out until then. Much longer runtimes only appear very rarely (e.g. for extreme settings) and would disturb the color coding.[7] Every setting was averaged over 10 runs. Unfortunately this method only works for two parameters and thus is not of much use for large systems with more parameters. For systems with 10, 20 or even several hundred parameters[8] it is simply impossible to try and run simulations with all possible combinations.

Another handicap of this method is the consumed runtime. In this case the two rates were raised in 4% steps from 0 to 100, meaning that $26^2 = 676$ initial settings

[7] In addition the simulation was broken of in case that the epidemic died out before reaching 50 time units – to save runtime.

[8] A normal CA is not going to have such an amount of parameters, but it might be the case for different methods.

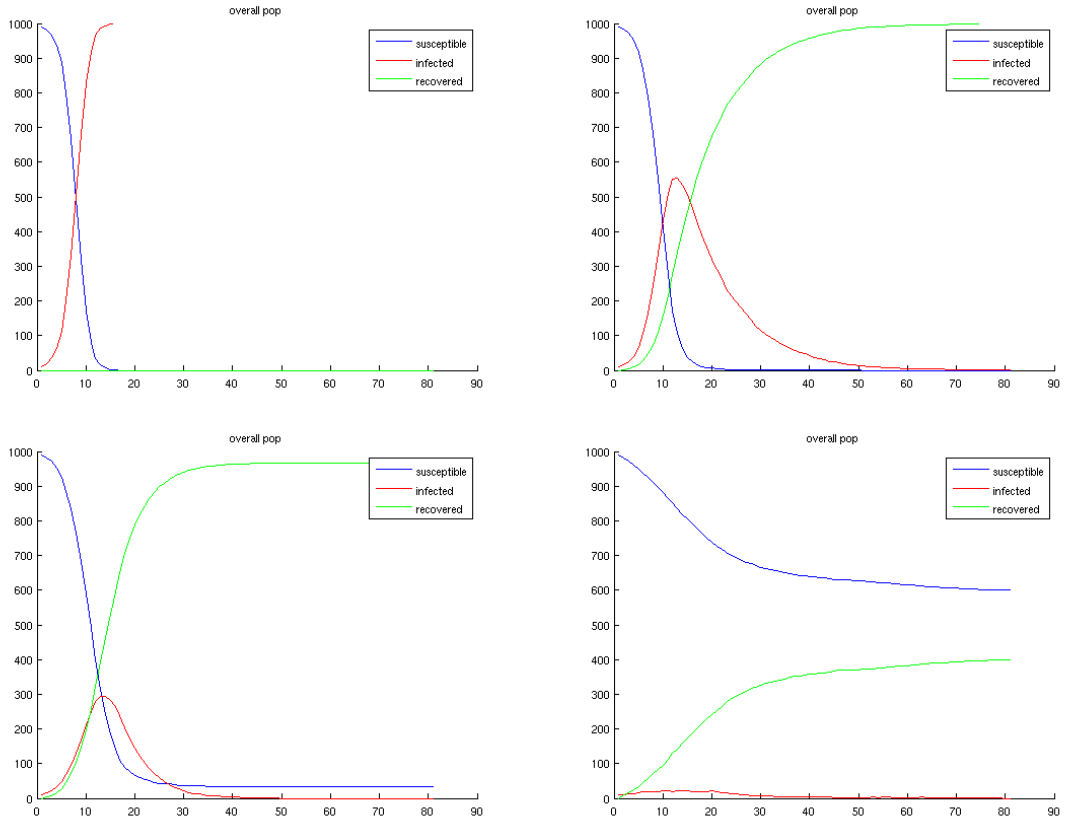

Figure 3.3: *First row: recovery rate 0 and 10%, second row: 20% and 40%, (see Fig. 3.2 for standar simulation = 30%, x-axis: units of time; y-axis: number of particles)*

needed to be computed. By averaging each setting over 10 runs this required 6,760 runs of the model[9]. If this method of running consecutive simulations with varying parameters is applied to complex models with long runtimes, this approach is extremely time consuming.

When looking at the outcome of this experiment, two graphs show to be particularly interresting: the development of the duration of the epidemic and the number of unaffected particles at the end of it. With the former changing its behavior at an infection rate greater than 50% and recovery rate over 20%. And the latter presenting only a very tiny corridor in which it is possible to escape infection. Regarding the graph "maximum no. of infected (particles per timestep)" one can easily understand the high level of infections for recovery rate 0%, in order to show details for a wider area the color coding maximum was set to 700[10].

[9]Some runtime can be saved by breaking off the simulation after the infection dies out.

[10]On the very left of the graph the actual number reaches the maximum of 1000 particles.

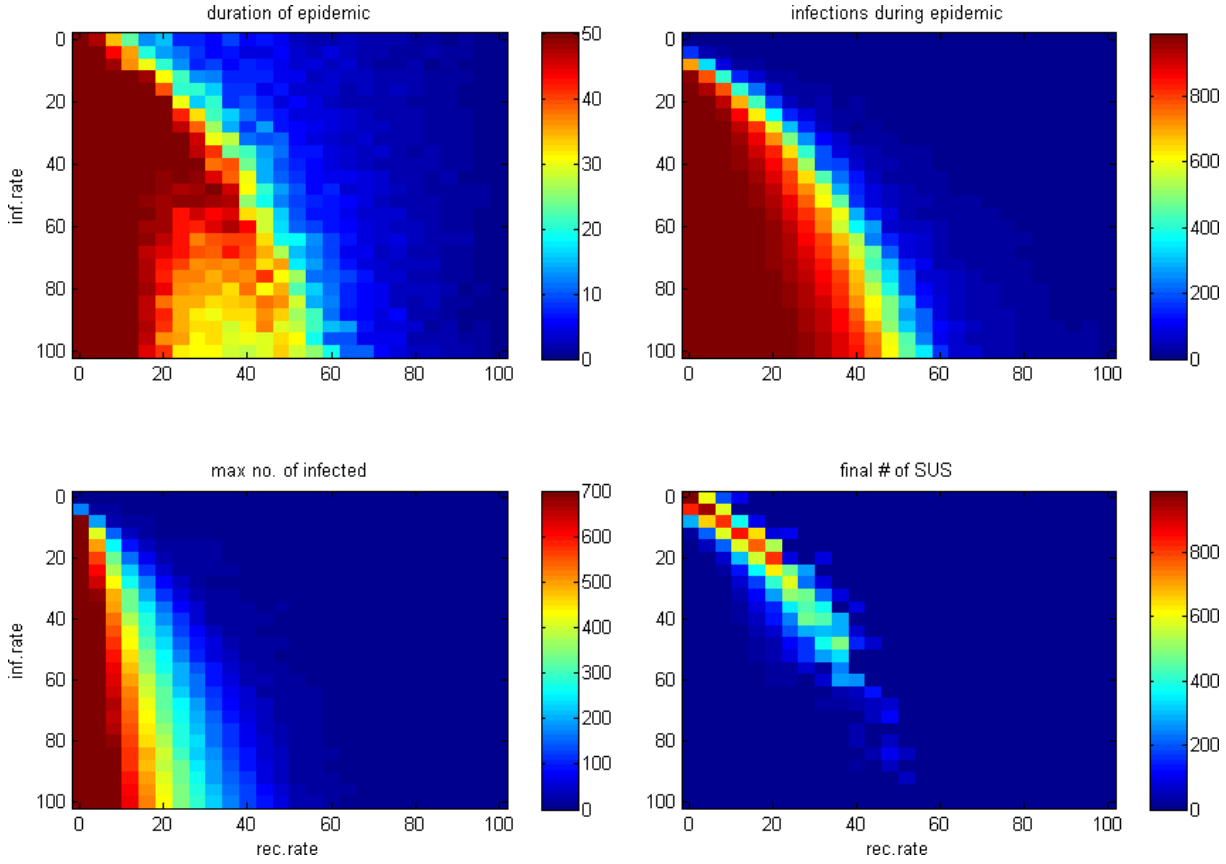

Figure 3.4: *First row: time until epidemic dies off (left), number of infections per simulation (right); second row: maximum of infected particles at one timestep (left), number of susceptible particles at the end of the simulation (left). (Axes: x-recovery rate, y-infection rate in percent)*

When replacing several of the susceptible particles with infected ones, system's behavior does not change drastically. The higher the initial number of infected particles, the smaller the raise of the peak and the earlier the infection dies out. This can be seen in Fig. 3.5 where an initial configuration of 50 respectively 300 infected particles is presented. The impression is amplified if one compares the results with the standard simulation (initially 10 infected particles). The results so far have not been too much of a surprise. An experiment that is already more interresting is the manipulation of density, meaning the ratio of particles in the system divided by available spots. It is obvious that if there are hardly any particles in the system (in relation to free spots) than the chance for contact and thus for risk of infection is minimal.[11] This risk is of

[11] A possible interpretation would be the spread of diseases in rural and urban areas (low vs. high population density).

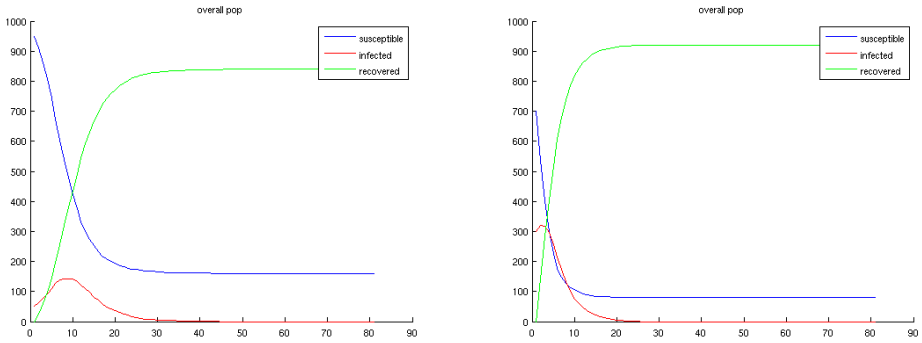

Figure 3.5: *Course of epidemic outbreak with starting configurations of 50 and 300 infected particles (x-axis: units of time; y-axis: number of particles).*

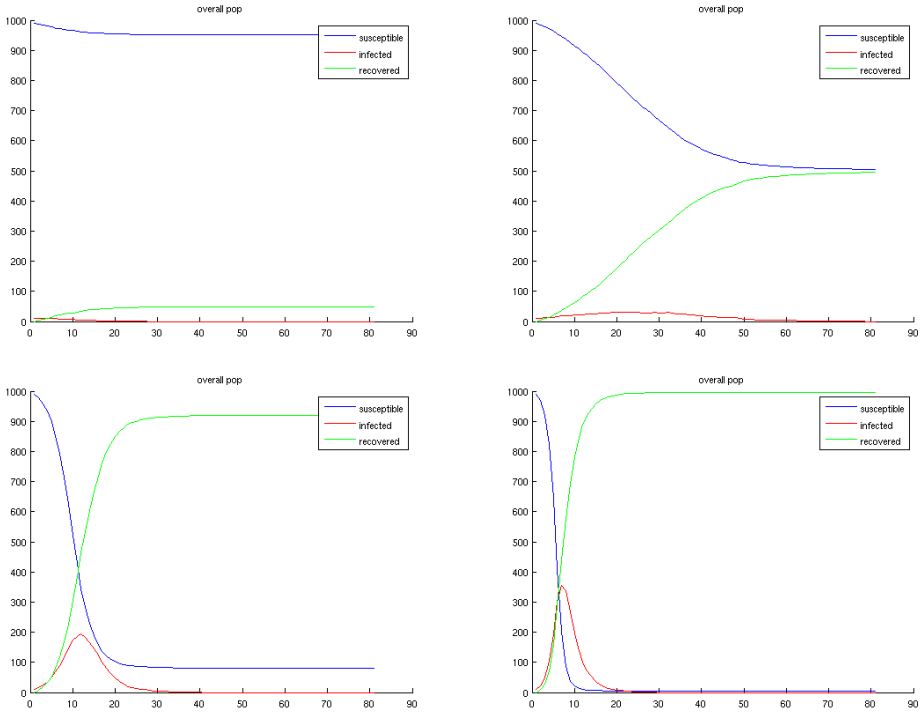

Figure 3.6: *First row: 10% and 20% of spots taken by particles, row two: 40% and 80% of spots taken (These numbers are approximate due to the fact that the number of spots in the automaton is* $6 * LengthOfAutomaton^2$ *(x-axis: units of time; y-axis: number of particles))*

course increasing with the density within the automaton as can be seen in Fig. 3.6.

3.1.3 Countermeasures - Immunization

We are now capable to conduct a first, simple countermeasure. If the fraction of susceptible particles decreases the infection does not find hosts as easily. This can for example be achieved by vaccination and thus immunization of particles and is very easy to implement in our model. All we have to do is to change the initial number of recovered particles. These particles cannot become infected any more and thus are already immune to the disease. As one can see in Fig. 3.7 this countermeasure is having quite some impact on the course of the disease.

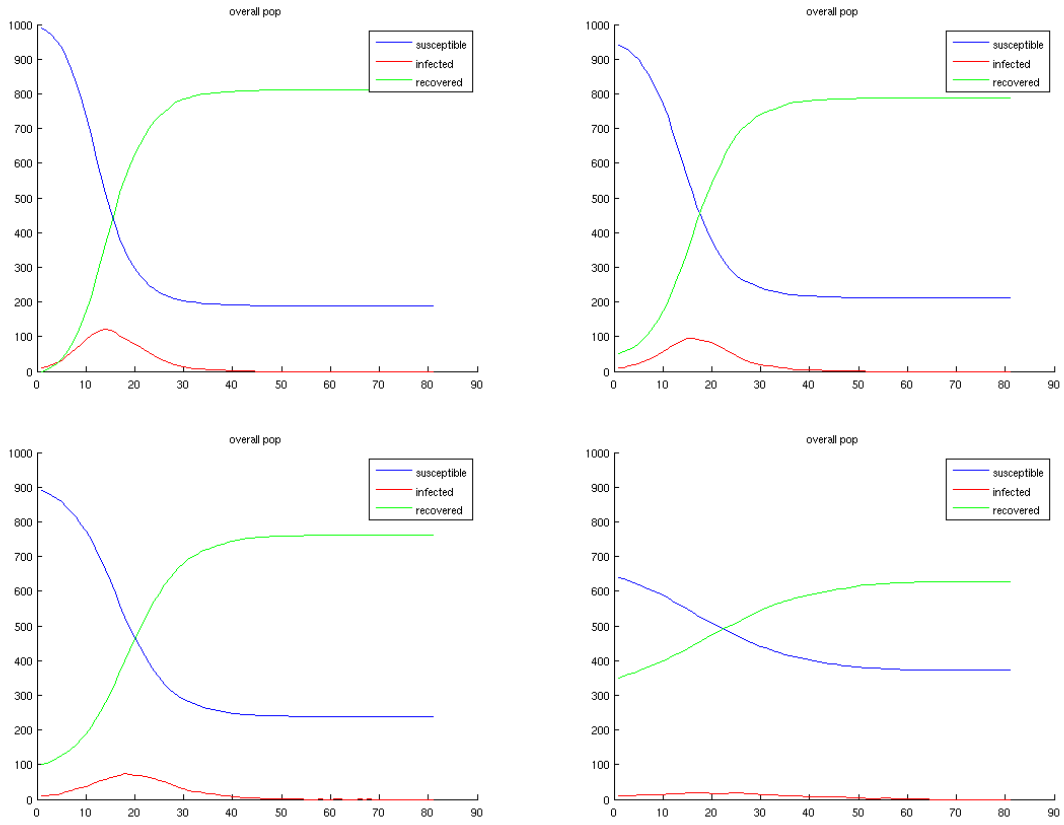

Figure 3.7: *First row: Standard run with no immunized particles and simulation with 50 vaccinated particles, row two: initial configurations with 100 and 350 immunized particles (x-axis: units of time; y-axis: number of particles).*

It is evident, that already a relatively small number of vaccinated particles does change the course of the epidemic. In the simulation with initial configuration of 350 immunized particles the epidemic is not even breaking out. This is rather surprising, even though the majority of the population is not immunized the vaccination of the

minority does prevent an outbreak of the infection. On the other hand this is prolonging the time until the infection dies out, which can become critical if for example immunity would only last for 30 units!

When following the idea of vaccination one can understand the importance of protecting so-called hubs in society that are crucial for the spread of diseases, such as hospitals, medical personal, etc.[12]

3.1.4 Transition to SIRS-Epidemic

Until now we did model a disease that can affect every individual only once. This is true for several real diseases, the most famous being the childhood diseases like measles, mumps, chickenpox, etc. But for many other diseases this does not apply. Thus our model will now be enhanced in order to simulate the loss of immunity. This means that our particles need to change their status from recovered to susceptible. In order to achieve this we could either add another random procedure or we have to expand our existing model. Choosing latter possibility we are going to add a time variable to our particles in which we will store the moment of recovery, allowing the particle to become susceptible after a given phase of immunity. By this we are actually leaving the strict concept of cellular automata[13]. Our model is now capable of simulating epidemic waves by transforming recovered individuals into susceptible ones after a certain amount of time.

In Fig. 3.8[14] we can see the effects which different immunization periods do have on our system. The effect does of course depend on the length of the immunity period. The main frequency for the simulations are approx.: 27.8, 35.7 and 50 days (from left to right).[15] If the immunity period is becoming too long the infection will die off. These effects are shown in more detail for the AB implementation by Fig. 3.14 through Fig. 3.16 of section 3.2.

[12]a) The national pandemic plans therefore usually strongly recommends to protect such personal by vaccination or other prophylaxis.

b) An interresting countermeasure experiment, the intentional placement of vaccinated populations within the automaton, was conducted in [L21].

c) Another kind of hubs could be places like harbors and airports. A study on the effects on disease spread via seaways has been conducted in [L30].

[13]Here the implementation of storing the particle information in a look-up table comes in handy. We now only need to add another row to our matrix in which the time of recovery will be stored.

[14]It is necessary to keep in mind, that these graphs represent the average of 30 simulation runs.

[15]These figures are calculated by dividing 1 through the “frequency per day” (which is represented by the x-axis).

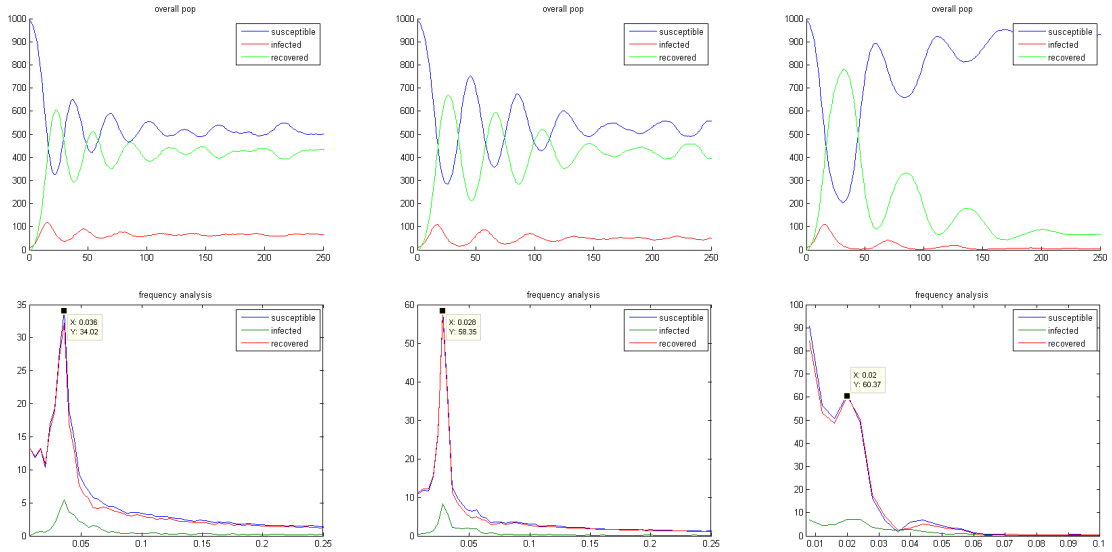

Figure 3.8: *First row: Our standard model with loss of immunity after 15, 20 and 30 time steps (x-axis: units of time; y-axis: number of particles); second row: fft-analysis of above results (x-axis: frequency per day; y-axis: unscaled).*

This new variable for the time-stamp that we just introduced, can be used not only for keeping track of the time of immunization. Since the health-states are mutually exclusive the variable can also be used to keep track of the disease length. This would allow an individual to recover after a specified time (average disease duration) – as will be done in the subsequent model.

3.2 Simulation with AB-Model

Of course it is also possible to model a SIR-type epidemic with multi agent systems. For this example we will use AnyLogic®, a softwaretool that claims being capable to deal with System Dynamics, Discrete Events, ODE-solving and Agent Based approaches. It is based on JAVA which on one hand is allowing the user to manually add additional code. On the other hand JAVA is not necessarily the fastest programming language, which proves to be a drawback for large simulations.

We will keep our assumptions the same as for the previous simulations (constant population, immediate infectiousness, etc.) and start by defining our smallest unit the agent. Since all agents are "equal" we just need to create one and then reproduce this prototype.

Our agent of course needs a starting position within the system and velocity to move. We thus assign two variables to represent the x- and y-coordinate and two more for the x- and y-component of the velocity. This is done by a picking two random numbers form a normal distribution in each case. In addition the agent does hold a (logical) variable which controls whether a contact/partner is available or not (starting value 0).

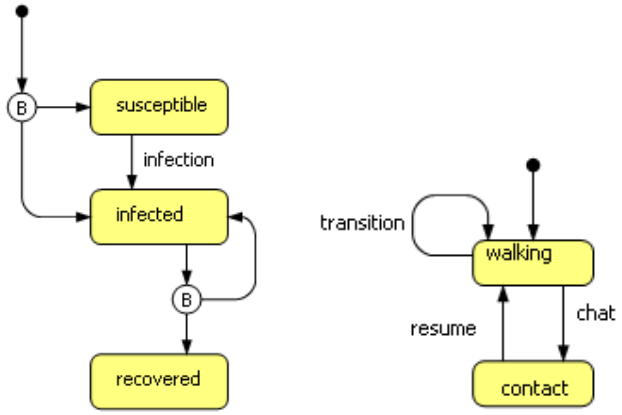

Figure 3.9: *State charts for health (left) and movement (right) transitions of the agents. Circled B's are branches at which –in this case– probabilities come into play. They are used to model the initial infection and the stochastic recovery of the agents*

The health states are again modelled as before, only allowing a susceptible agent to become infectious and only infectious agents being able to recover. This is solved by assigning a state chart *health* to the agent (see Fig. 3.9 left). Next we define movement, contacts and thus infections of the agents. This is solved by adding another state chart

(*movement*) to our prototype agent (see Fig. 3.9 right). This information and properties are sufficient for our purposes. If desired, it is possible in AnyLogic to produce colorful visualizations by creating animations of the system.

Next step is to define the infection procedure. We assume that infection is only possible during contact, thus we need to define *contact* – an event triggering the infection routine. For this a neighborhood is defined – these definitions can of course vary. In the present example a *"field of view"* is defined by vision range and distance (see Fig. 3.10 for a sketch). If an agent crosses another agents field of vision both are stopped and set as partners. This is triggering the *infection* event. In case that one of them is infected the second is put at risk of infection with a given probability. After a short contact break both agents continue to travel with new velocities and slightly offset travelling directions.

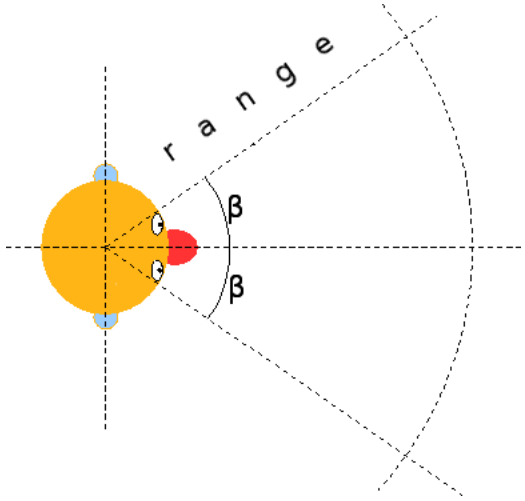

Figure 3.10: *Sketch of an agent's vision field – defined by vision range and vision distance.*

In case an agent reaches the boundary of the environment different strategies are applicable (e.g. periodic boundaries, reflection, etc.). In our simulation the reflective border condition was chosen because of better visualization. This means that an agent reaching the predefined border is triggering the event *transition* which results in the change of the leading sign of the velocity component interfering with the boundary (x- or y-component).

All that is needed now is to replicate the agent sufficiently often and run the simulation.

3.2.1 Stability Analysis

In this model setup we can already spot quite a few parameters that influence the outcome of our model. Obviously the infection and recovery rates, which we will not cover again[16]. The velocity and the size of the field of view also do play an essential role since they influence the number of contacts between individuals in our system. The number of contacts is of course also determined by the density of our population.

In the next sections we will tests the influence that modification of these parameters has upon our system. Due to programming issues the simulations have not been averaged, but all figures are drawn from representative results.

Maximum Velocity

The velocity of the agents is a random number between zero and *maxVelocity* –variable set at beginning of simulation. In the test runs all parameters have been kept the same, only *maxVelocity* was changed from 10 to 30 and 60[17]. We can clearly see the effects of these changes in Fig. 3.11. Distance of the vision field is set to 15 during this experiment[18].

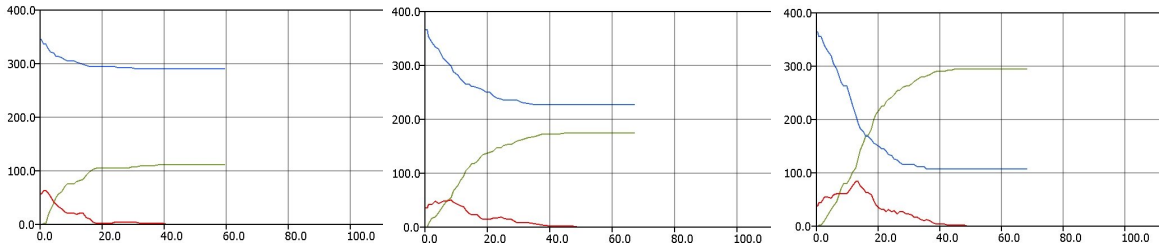

Figure 3.11: *The same simulation run with different* ***maxVelocity*** *of agents, from left to right: 10, 30 and 60 (color coding: susceptible=blue, infected=red, recovered=green; x-axis: units of time; y-axis: number of agents)*

Although the duration of the infection is quite short in all three runs, ranging from about 20 to 40 time steps, the outcome varies drastically. The number of individuals infected in the course of the epidemic is approximately doubling from run to run. The behavior of the overall system does not change abruptly by modifying this parameter. Only if we would let *maxVelocity* tend towards zero the infection would logically die off.

[16] See section 3.1 for experiments with infection and recovery rates.

[17] These figures have been arbitrarily chosen in order to obtain meaningful results.

[18] In order to compare this experiment with the experiment in **Vision Field**.

Vision Field

In this experiment we are going to test the impact of the size of the vision field. This is, in our AB implementation, the equivalent of the CA neighborhood. Although the vision field is calculated in a more complex way, namely by defining an angle of range, relative to the traveling direction of the agent and a vision distance (see Fig. 3.10).[19] We could of course test changes of both range and distance, but it is sufficient to vary only one of them because the effect we want to achieve, is to raise the number of contacts. This number is increasing with the area covered by the vision field. It is – more or less – regardless how the area is shaped.

The following results were produced under the same circumstances as before, varying only distance from 5 to 15 and 25 units of length (see Fig. 3.12). All experiments have been run with a *maxVelocity* of 10 for comparison purposes[20].

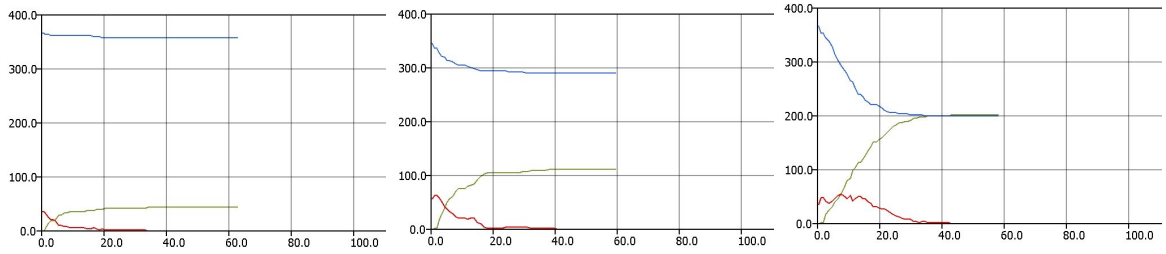

Figure 3.12: *The same simulation run with different* ***distance of vision field****, from left to right: 5, 15 and 25 (color coding: susceptible=blue, infected=red, recovered=green; x-axis: units of time; y-axis: number of agents)*

We again notice quite a strong influence of the parameter on the system. The population of recovered individuals -implying they have been infected during the epidemic- is again doubling at each run. The duration of the epidemic is also quite similar to the last experiment, varying from 20 to 35 time units. And same as before no abrupt changes occur save for the trivial case of setting the vision field to zero.

3.2.2 Extension of the Model - SIRS

As with the CA-model we will now expand the model in such a manner that it is capable of modelling SIRS-type epidemics. Since we already did define the state chart *health* for our agents we just need to refine it in such a way that it meets our needs.

[19] Of course this complexity also holds the advantage of more possibilities for the implementation.

[20] Thus run 2 of this experiment is the same as run 1 of the experiment in **Maximum Velocity**.

This is done by adding a transition from *recovered* to *susceptible* state (see Fig. 3.13). All that is left to do is set the trigger for this transition. In this case the trigger is a time interval specifying the duration of immunity.

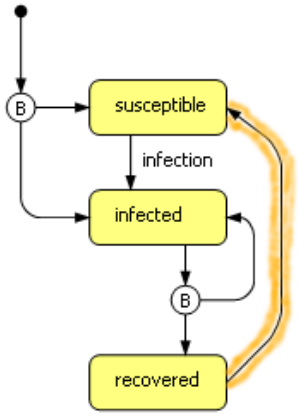

Figure 3.13: *Modified state chart "health" – this implementation extends the model onto a SIRS-type epidemic (see highlighted arrow on far right).*

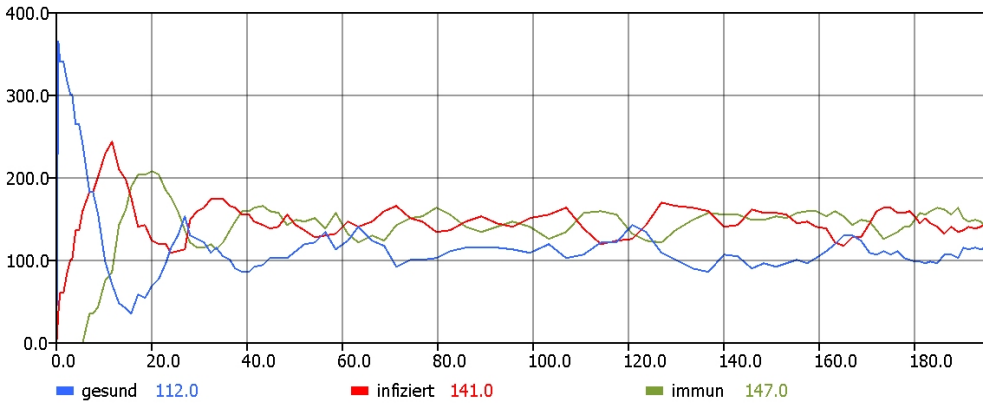

Figure 3.14: *Epidemic spread with immunity lasting for 10 units of time (x-axis: units of time; y-axis: number of agents)*

The duration of immunity does have a strong influence on the course of the epidemic. We will take a look at this influence by comparing immunity intervals of 10, 20, 30, 40 and 50 time units (see Fig. 3.14 through Fig. 3.16). These experiments have been run with *maxVelocity* of 30 and a vision distance of 5 units.

Firstly one can notice that for immunity length of 10 units (Fig. 3.14) the periodic pattern of the epidemic is not yet visible. With growing length of immunity the visibility of this pattern improves as well as the amplitudes of the sub-populations. The

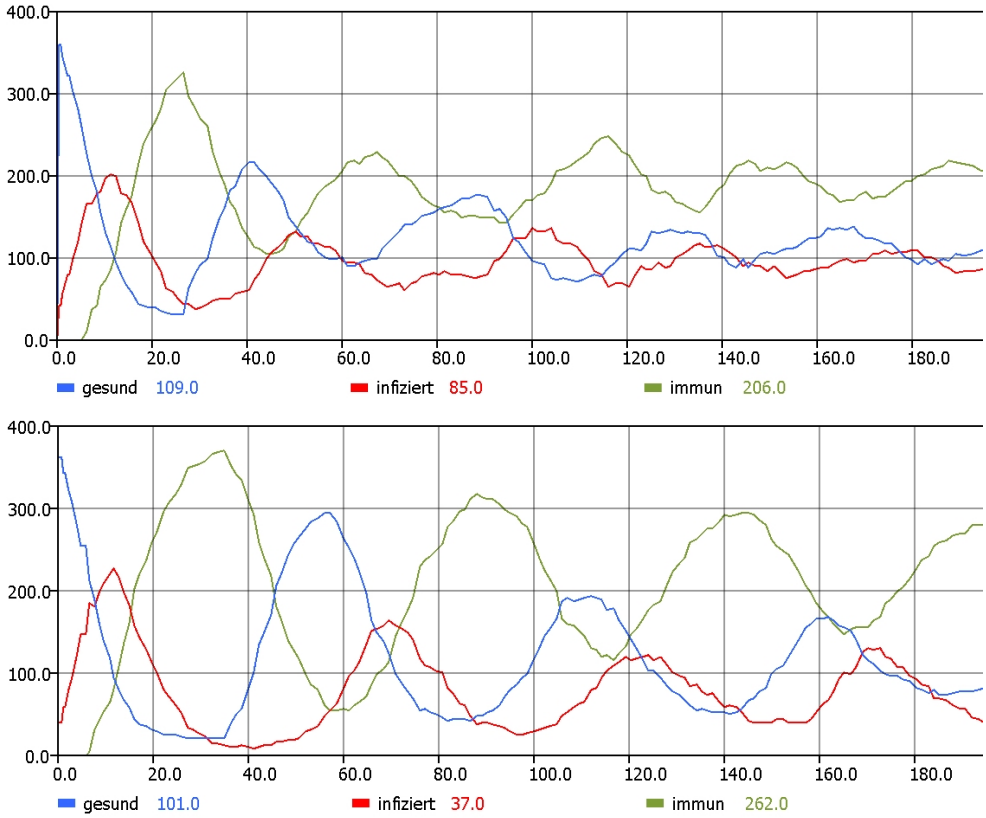

Figure 3.15: *Epidemic spread with immunity lasting for 20 (top) and 30 (bottom) units of time (x-axis: units of time; y-axis: number of agents)*

frequency on the other hand gets lower with increasing length. It is also visible, that in Fig. 3.15 and 3.16 the local minima of the infected population is dangerously close to extinction, thus to an extinction of the disease itself. In the next step (50 time units, Fig. 3.16) the minimum of the infected population reaches zero and the epidemic dies off due to a lack of hosts spreading the disease. The logical consequence that follows is the extinction of the sub-populations of immune individuals – with a delay equal to the duration of immunization.

Variation of the starting population of immunized (recovered) agents has been undertaken in the CA section.

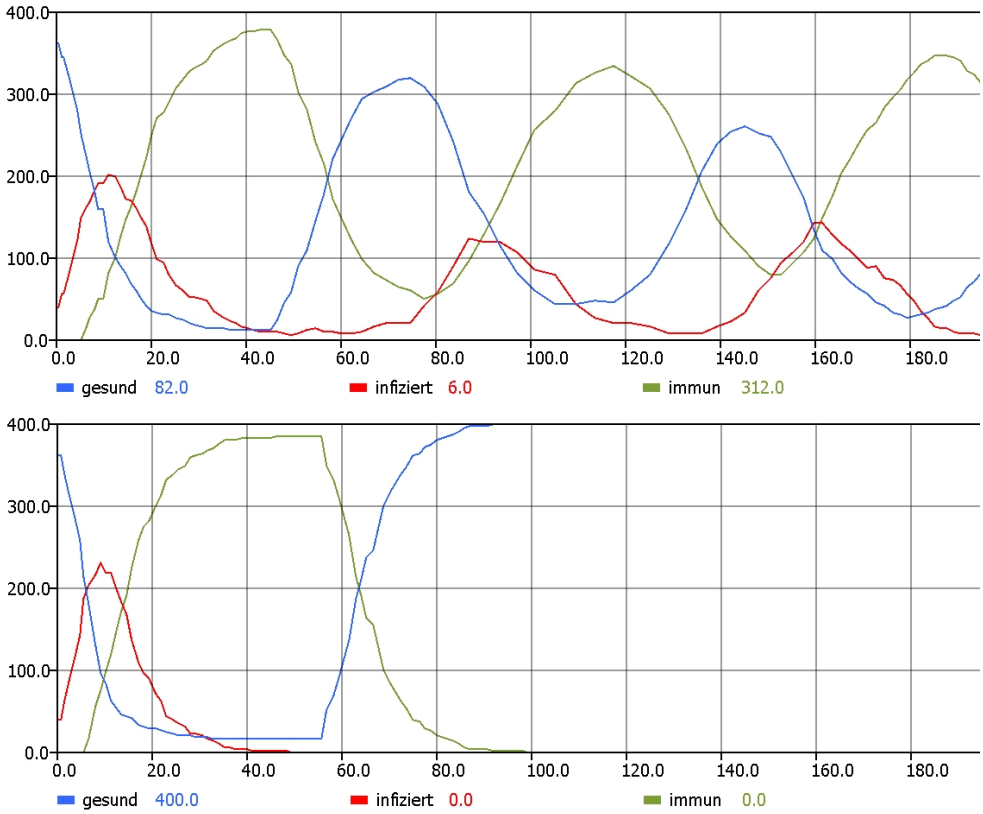

Figure 3.16: *Epidemic spread with immunity lasting for 40 (top) and 50 (bottom) units of time (x-axis: units of time; y-axis: number of agents)*

3.3 Comparison with ODE Approach

Finally we will take a short look at the classic way to solve the question of how such a SIR epidemic would look like. Based on the ODE-system (1.1) from chapter 1 we can compute solutions of the model. The ODE-system's behavior when fed with our standard parameters from the CA approach as well as with an increased 100% infection rate is displayed in Fig. 3.17. One can notice that the qualitative behaviour is very similar to the behaviour seen in the previous experiments. Of course there are differences when looking at the quantitative behaviour which can be explained by the influence of density and spatial distribution within the cellular automata.

And of course the influence of the infection rate is stronger within the ODE system. This is explained by the fact, that the infection rate within a CA is only applicable to susceptibles with infected neighbors. In the ODE model the infection rate is scaled by the number of infected individuals, but still remaining global because of the perfect

homogeneity of the population.

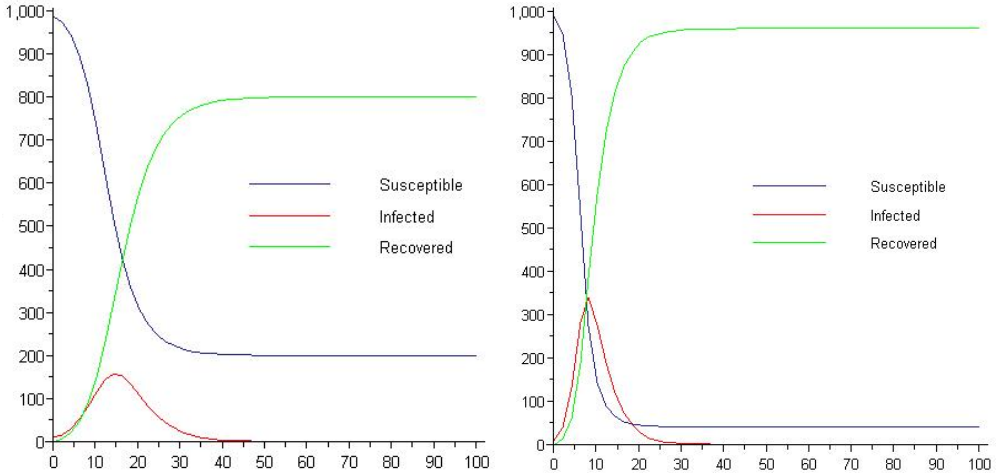

Figure 3.17: *The ODE-system in comparison: with standard settings from the CA (left) and with 100% infection rate (right) (x-axis: units of time; y-axis: number of individuals)*

3.4 Conclusion

Although we have been looking at fairly simple models of simplified epidemics one can already see quite a lot of interresting things. Firstly we showed that it is possible to model such an epidemic with cellular automata as well as with multi agent systems – both leading to similar results. The difference in results can be explained by different parametrization.

Which leads us to the next interresting topic: parametrization. As visible in sections 3.1.2 and 3.2.1 these small systems already offer quite a spectrum of parameters that need to be adjusted in order to fit the model (e.g. to match real data from epidemics).

Further the differences of the two approaches can be measured in terms of runtime. Both models were run with a population of 1000 individuals (a fairly small population) and an initial infection of 10%. For both models only the actual computation without initialisation was timed. The AnyLogic implementation of the AB approach took 100 seconds to compute 100 units of time[21]. During the the same time the CA implementation computed approximately 16,300 units of time[22]. This difference grows

[21] One has to be careful when comparing the units of time and the outcome of different models/approaches. If the quality and quantity of results shall be compared, a propper fitting is necessary.

[22] These figures have been measured on a fairly slow machine (notebook with Intel Pentium M 1.6 GHz CPU and 512 MB shared memory).

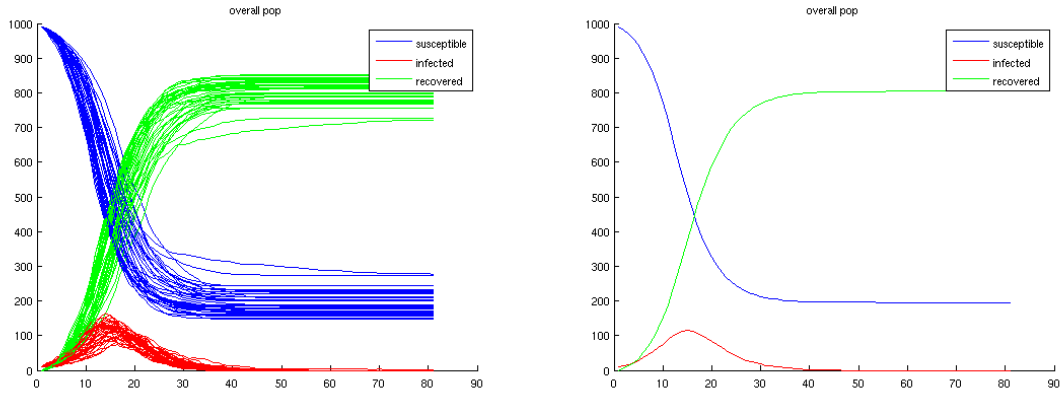

Figure 3.18: *The left graphic does show the variation of 40 consecutive runs with standard settings, the right graphic does show the computed average of those 40 runs.*

even larger for bigger populations[23].

Since CPU and memory usage are the factors that limit the performance of the simulation these findings are of high importance. Of course these effects can be overcome by the use of faster and more powerful machines, but one must keep in mind that these experiments were of a very small scale. The final simulation should be able to simulate the epidemic pattern for metropolitan cities or even countries over a whole disease season.

Since infection procedures in these models are of probabilistic nature a single simulation run is not sufficient to provide good results. In order to flatten the curves (representing single, stochastically influenced simulation runs) and receive a reasonable average behavior it is necessary to run the simulation several times[24], this magnifies the effect of poor runtime (Fig. 3.18 visualizes the effects of and the necessity for averaging several runs). To sum things up: it is necessary that the model is fairly fast and efficient.

The AB approach turns out to be on one hand computationally less effective but on the other more flexible than the CA approach. Looking at the enhancement of our CA in order to simulate a SIRS epidemic, we did already leave the strict definition of cellular automata into direction of agent based systems. This backs our basic idea – brought up in the introduction – to combine the two methods within a single model.

[23]It needs to be mentioned that the AB implementation was quite intuitive and the runtime could probably be improved by code optimization. Nevertheless a difference of factor 150 – 200 is an astonishing difference.

[24]Monte Carlo method, see [L6]

Chapter 4

The Extended Influenza-Model

In this chapter we will set up the final model to simulate an influenza epidemic in a realistic population. With this model we will run several experiments and interpret the findings. The models set up in chapter 3 did lack detail and thus realism. As every model is (and can be) only a simplification of reality one has to decide what kind of information is necessary for the modelling.

Since the goal is to model an influenza epidemic one needs to consider the characteristics of the disease. In the next step the social circumstances need to be modelled. To simplify the social behavior of humans is probably one of the hardest parts. How to condense this behavior for it to fit into a machine, without neglecting (the most) important influences. Yet another question that arises, is which methods to use for which part of the model and which programming language to choose.

Because it is efficient and convenient to use matrices for the modelling of such a system, we will base our model on them. As we already discussed in section 2.5, MATLAB for our purposes appears to be a good and useful simulation environment and therefore will be the programming language of choice.

The set up model has to be capable of producing reasonable results with the data provided and available. This of course sounds logical and simple but it often turns out to be harder than expected, as we will see in section 4.2.

4.1 Model Structure

In this section we are going to examine the characteristics of the disease and the demographic structure in which our model will be located. From the information drawn out of this investigation, we'll be able to set up the frame for the model structure.

4.1.1 Influenza

Influenza is an infectious disease caused by a virus of the family of influenza viruses of type A or B. Firstly this means that antibiotics are not effective against it. In addition the virus has the bad habit of constant mutation meaning that every year/season some strains die out and some new ones appear. This fact further decreases the effectiveness of drugs designed to conquer influenza.

The virus affects humans as well as other mammals and birds. It is mainly transmitted via body fluids such as saliva, blood and most important by coughing or sneezing. Alternatively infections also occur through contact with surfaces that have been contaminated with the virus. The survival time of the viruses varies, studies imply that at least for some strains the survival time is increasing. For the H5N1 virus the survival time at human body temperate of 37° C was two days in 1997 and six days in 2004. The survival times increase as temperatures decrease, allowing it the virus to survive longer in winter and in cold surroundings. It is suspected that it may even survive indefinitely long in frozen substances.[1]

Influenza usually appears in two seasons every year – during winter in the northern and southern hemisphere. Every year it affects between 5 and 20% of the worlds human population[2]. Most people suffering from influenza will recover after a period of one to two weeks. The greatest influenza related danger comes from secondary infections (mainly pneumonia) which can quickly become life-threatening[3].

Worldwide the influenza related (category ICD-10[4]) death toll reaches some hundred thousand people every non pandemic year. In western countries nearly all casualties of the usually circulating strains are over the age of 60. This group is at extreme risk of developing pneumonia if infected with influenza. The probability for this lies around 90%. Nevertheless one has to keep in mind that these figures are subject to change and mainly depend on the circulating strains. For example, more than 90% of the causalties of the Spanish flu pandemic[5] were under the age of 65.

The typical incubation period of influenza ranges from 1 to 4 days with an average of 2 days. And to make things more complicated infected persons are infectious already up to two days prior to development of symptoms. The symptomatic phase of the

[1] see [L5]

[2] These figures vary largely depending on the active strains.

[3] Pnuemonia is held accountable for 80-100% of all influenza-related deaths.

[4] ICD is the International Classification of Diseases published by the WHO.

[5] The Spanish flu was raging from 1918 to 1920, killing an estimated 15 to 40 million people.

disease lasts for approximately one week.[6]

Treatment and Prevention of Influenza

Effective treatment of influenza is limited to neuraminidase inhibitors, such as Tamiflu® (by ROCHE) or Relenza® (by GLAXOSMITHKLINE) and M2 inhibitors (adamantane derivatives). Since the latter are having quite serious side effects they are not used too often for influenza treatment. The aim of both drugs is to halt the spread of the virus within the body. If taken early enough they may shorten the recovery process by a few days. These two drugs are supposed to have a prophylactic efficiency of 70%, attested by the "Pandemieplan 2006 – Strategie für Österreich"[7].

Prophylaxis by vaccination is possible, although the effectiveness does strongly depend on the vaccine mixture. The WHO[8] every year issues a recommendation for a number of strains that are probable to circulate in the season following. The vaccine mixture is then produced by pharmaceutic companies according to this recommendation. If the recommendation does predict the strains correctly the vaccine is quite effective, although not fully excluding the possibility of infection. The effectiveness of the vaccine varies widely, but can reach up to 80% in ideal cases.[9]. The above mentioned "Pandemieplan 2006" does specify a vaccine effectiveness of 70-90% for people under the age of 65 and 30-70% for those older than 65 years[10]. Nevertheless most national influenza centers recommend that at least high risk groups should use prophylactic vaccination. Such guidelines are set up by every nation itself, but usually derived from international cooperation and guidelines issued by the WHO[11]. These groups usually consist of elderly people, people with weak immune system and medical personal (because of above avaerage exposure to the virus).

The problem of vaccination is to correctly determine the circulating strains. Since in pandemic cases one is facing a new mutation of the virus the vaccines will not be effective against it. The problem that arises in this situation, to produce as much serum as fast as possible, is that it takes three to six months before serum for a new strain can be produced in necessary quantities. During this time there is hardly any medical assistance except for the above mentioned drugs.

[6] Most information of this section was taken from medical online platforms, such as [W2, W8, W10].
[7] Pandemic Emergency Plan of the Austrian Ministry for Health and Women, see [L8]
[8] World Health Organization
[9] see [L16, L16]
[10] These quotas are again reached "under ideal circumstances" (see [L8]).
[11] see [W10] for web address

Unfortunately there is no available data for infection probabilities overall or for specific age groups. Although it is known, that normal influenza stems are mainly affecting young children and senior citizens. We will cover the topic of acquiring data to describe the process and likelyhood of infections later on in more detail (in section 4.2).

4.1.2 Demographic Structure

The model will be settled in Vienna, the capital of Austria. The demographic structure and data is derived from the 2001 census and 2004 micro census. The data is collected and processed by "Statistik Austria"[12]. The demographic structure of the Viennese population[13] for 2001 was as follows:

Population by age groups

Age	Inhabitants	in %
under 4	73,857	4.87%
5-9	76,849	5.06%
10-14	74,871	4.93%
15-19	74,485	4.91%
20-59	896,062	59.04%
60-79	263,846	17.39%
80 plus	57,679	3.80%
total	1,517,649	

The average inhabitants per household for Vienna were exactly 2. The detailed numbers for people living in the same household are as follows[14]:

[12] "Statistik Austria is an independent and non-profit-making federal institution under public law, responsible for performing scientific services in the area of federal statistics.", cit. [W9]

[13] see [L32]

[14] see [L33]

Inhabitants per Household		
Household size	Inhabitants	in %
singles	359,500	49.82%
2 persons	221,300	27.88%
3 persons	108,500	13.67%
4 persons	72,800	9.17%
5 persons	22,200	2.80%
6 and more	9,500	1.20%
total	793,800	

For the year 2005/2006 81.4% of the children in Vienna aged three to five (under the age of six) were in child care facilities[15], beeing a total of 53,864 children . During the same time 1,863 active child care facilities have been counted in Vienna, leading to an average of 28.91 children per facility[16].

In the year 2005/2006 "Statistik Austria" did count 62,113 pupils in elementary schools (aged 6-10) and 150,579 pupils in middle schools (aged 10-18) for Vienna. These numbers do not include special schools for mentally and/or physically challenged pupils (3,367 students). In that period of time the number of elementary schools in Vienna was 249, the number of middle schools 402, thus the average number of students per school would have been 249.45 and 374.57 respectively[17].

The 2001 census identified 821,458 employed persons and 87,691 working places in Vienna. This means that on average 9.37 workers were employed per working place. Following the 2005 micro census the average unemployment rate in Vienna did lie at 9.1% (of people aged 15 to 64) for the year 2005.

4.1.3 Shaping the Model

Since it is impossible to model the exact social behavior of all inhabitants it is necessary to compress it. This reduction of course needs to be sufficient in terms of representing the real behavior as well as in being simple enough to ensure a reasonable implementation and computation runtimes.

We can assume that an average persons day is divided in three major parts. The

[15]see "Statistik Austria" press release No.: 8.599-108/06

[16]Data derrived from cit. [L31].

[17]Data derived from "Statistik Austria" Excel-sheets:
(http://www.statistik.at/fachbereich_03/Uebersicht_Schulen_2005_06.xls
http://www.statistik.at/fachbereich_03/Schueler01_2005_06.xls)

first part being working time or time at school, second leisure time and social life and the third part time at home respectively sleeping. This basic division at the same time already seems to be a sensible reduction of the social behavior. We will take the following for a rough assumption: At the work place, child care facility or school a person is going to meet the same people every day. During leisure time a person usually visits friends, doctors, goes shopping, et cetera and usually stays within a defined surrounding. The people one meets during this time are often the same. Finally being at home it is assumed that only the family is together.

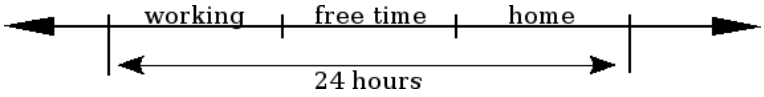

Figure 4.1: *Visualization of daily routine on a time schedule. The three parts of the day do not necessarily have to be equally long!*

This is naturally not representing real life social behavior. And especially certain work groups (e.g. sales people, medical personal, etc.) do have much more contacts than others. We must not forget about this, and it might prove useful in case the model is being expanded and more detail implemented.

This structure already implies several things that we need to care about when implementing the system in our programming language. For example the modelling methods used for the different parts of the model. It seems logical to use the agent based approach to control the whole system and cellular automata to model the sub-systems (schools, working places, neighborhoods, etc.). Such a structure would allow our agents to switch easily from one sub-system to another, without any time gaps within the model (see Fig. 4.1). In addition the CA modelled working places can be computed quite fast with MATLAB. On the other hand the reduction already shows some shortcomings of the model. For example there are no long-distance contacts included. We also do not consider all contacts during the leisure time but simply replace them by a "neighborhood" of random people.

At the beginning of the model the population is randomly initialized with the parameters derived from demographic data. This means that every agent does have a unique ID, a certain health state, an age, a work place (or child care facility respectively school) depending on its age, an assigned household and neighborhood. To further simplify the model it is assumed that children under the age of three are not going to any child-care facility or similar but are staying at home during the working part of the day. Further full employment with no job less people is assumed.

Every day all agents move to their work places (respectively schools or child care facilities) and spend the working time there. Excluded from this procedure are senior citizens which are assumed to stay at home during this time. The simulation of the working time is done by cellular automata: every workplace is simulated in a separate automaton. This is convenient to implement and offers great potential for parallelization. This is becoming specially interresting in the near future with increasing numbers of cores on CPUs.

After work the agents proceed to the neighborhoods which are again simulated separately by cellular automata, thus parallelization is applicable here as well! It would be possible to process several households/neighborhoods parallel on machines with multiple processors or cores and by this improve the performance of the model.

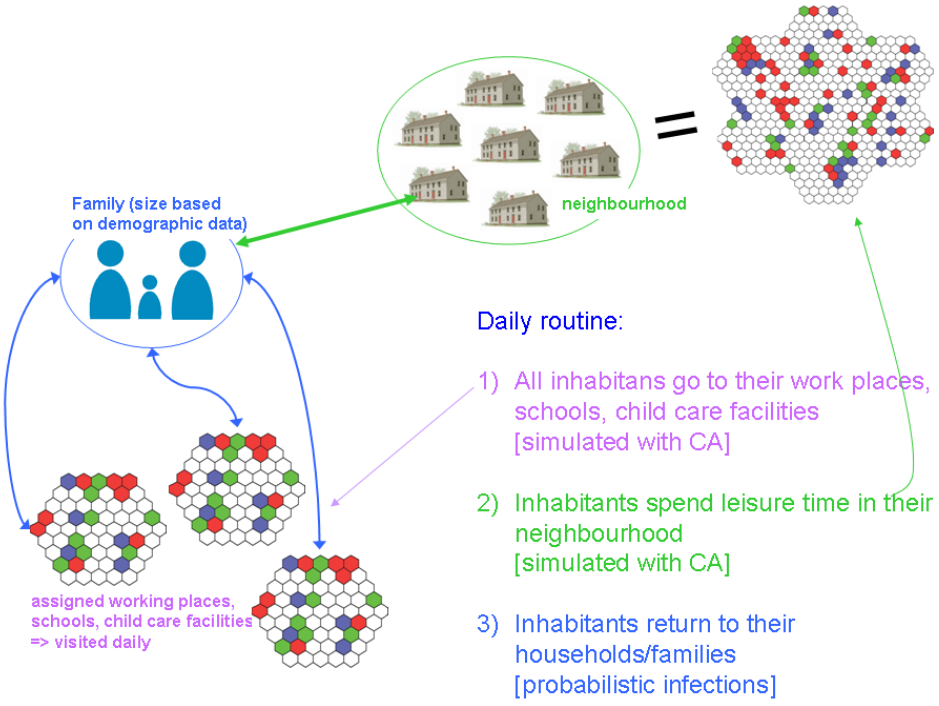

Figure 4.2: *Social structure of the used model.*

At the end of the day the agents return into their households. Here infection is simulated by simple probabilities since contact between all members of the household can be taken as given. In single households infection is of course not possible. An overview of this model structure is given in Fig. 4.2.

Attributes of Agents

The implementation of the agents is realized by a look-up table, as already sketched in chapter 3. This look-up table, in form of a matrix, contains all agent related information which is used throughout the simulation. This is a very convenient way to solve this task, since firstly MATLAB is handling matrices in a very efficient way. Secondly by defining this look-up table as a global variable all subroutines are able to directly draw out relevant information and update the states of the agents. One can imagine the global look-up table lying alongside the model, always being in range to draw information from it or write changes into it. In addition it is very simple to add attributes if the model is enlarged or extended (as shown in section 3.1.4).

The look-up table does store following agent related information:

Information stored in look-up table

Row	Attribute
1	Key - unique ID of agent
2	Health status
3	Age of agent
4	Household ID
5	Neighborhood ID
6	Workplace number (age specific)
7	Timestamp
8	Infection probability (age specific)
9	Controlvariable (if symptomatic)
10	Help variable

Variables one to six are self explanatory. Entry seven is used to save the time of infection, outbreak of symptoms, time of recovery or time of death. This is possible since the health states are mutually exclusive, as explained earlier.[18] Variable eight is again self explanatory, and variable nine is used to control whether a symptomatic person is going to stay home or continue to be part of the daily routine. Finally variable ten is an auxiliary variable used to keep track whether the probability for death already was applied on the infected agent for the current case of disease[19].

The health status of the agent is currently structured in 5 parts, namely being susceptible, infected (without symptoms), symptomatic, recovered (and immune) and

[18] Thus the current health state of an agent characterizes the event which the variable is tracking.

[19] The agents are put at risk of death during every infection only once.

dead. For future extensions the respective numbering is not consecutive (see table below). This is allowing to add more states to the model without the need to rewrite large parts of the program or loosing logical numbering.

Health states of agents

	State
1	Susceptible
2	Susceptible - type II (not yet used)
3	Susceptible - type III (not yet used)
4	[reserve]
5	Infected without symptoms
6	Symptomatic
7	[reserve]
8	Recovered (and immune)
9	[reserve]
10	Dead

For example could state two and three be used for vaccinated, or otherwise protected, agents and state four and seven to model the course of the infection more detailed in future extensions of the model.

Such a look-up table for 1,000,000 agents does consume approximately 7 MB of memory. This is for a matrix of type `int32` as used in the model. The amount of memory used of course depends on the data type of the matrix. A matrix of type `single` or `double` would need 35 MB or 70 MB respectively. Thus one should think about code optimization in terms of memory usage and of course runtime as well.

A Tribute to Runtime

As we are aiming to simulate large populations with the model that we are about to build, the use of resources used by the program is crucial. For example, a common (and widely spread) mistake when implementing CA is to use two lattices[20] and copy their contents back and forth. This seemingly intuitive approach turns out to be quite inefficient.[21]

With some optimization effort it is possible to distinctly improve the performance of the implementation. The necessary steps of course also depend on the platform used

[20] One representing the current state, the second one to store the results of the update.

[21] see [W3]

for programming. MATLAB being a computer language optimized for the treatment of matrices and vectors, is suited to take advantage of these data structures. One example for the use of this feature is to use vectorized loops. A simple example would be to replace search functions implemented with loops, as for example

```
% IDs of agents with status=2 are placed in vector 'infected'
index = 1;   % Index for addressing correct entry of vector
table_size = length(agent_table(1,:));

for i=1:table_size
  if (agent_table(1,i)==2))
    infected(index) = i;
    index = index + 1;
  end
end
```

by vector based implementations, such as

```
% IDs of agents with status 2 are placed in vector 'infected'
infected = find(agent_table(1,:)==2);
```

This is not only shorter in code, but essentially faster than the loop-equivalent. For a matrix of size 10 by 100,000 created by

```
% matrix similar to the look-up table used in the model
agenttable=round(rand(10,100000)*4);
```

the runtimes to execute the first loop using the *for* and *if* commands had a runtime of 1.344 seconds, whereas the vectorized loop took only 0.019 seconds. Thus the vectorized loop is 70 times faster![22]

4.2 Disease Data

As mentioned in section 4.1.1 accurate data of disease transmission and infection probabilities is not yet available. This might change since such data is might be (or maybe already is) recorded easily with the introduction of the E-Card[23].

In addition available rates would be very specific to certain settings. However this data is inevitable for a realistic model and significant simulation results. The optimum would be to have data of infections with high spatial and temporal resolution. Since influenza does have an incubation and duration period of a few days the temporal resolution is much more important than for models of diseases with longer periods

[22]Both loops tested on a note book with CPU 1.6 GHz and 512 MB shared memory.

[23]The E-Card is a chipcard that did replace the paper health insurance certificate in Austria. Data regarding the patient is stored on it and centrally processed.

(e.g. HIV). The infection risk is also a completely different one (e.g. it is very unlikely that one individual infects 30 other with HIV on a single day). Data collected by medicals with the day of diagnosis, residential location and age of the infected individual would be ideal.

The Sentinella-Network is collecting this kind of data in Austria as well as in several neighboring countries. The findings of the Austrian network can be viewed at the homepage of the "Diagnostisches Influenza Netzwerk Österreich"[24] (abbr. DINÖ). It needs to be mentioned that the sites is copyright protected by "Roche Austria GmbH" – the company producing the influenza drug Tamiflu®, a neuraminidase inhibitor.

Unfortunately this network is very loose, since it only consits of about 50 physicians, spread over all of Austria and does not cover metropolitan areas. In addition the correct diagnosis of influenza often requires a confirmation of the virus by a laboratory to exclude common cold as cause of the symptoms. This confirmation costs time and money and is therefore only done in random cases. This is supported by the fact that during influenza season the rate of correct influenza diagnosis by medicals is very high.

Of course some data or information on the course of influenza epidemics does exist. Together with the Viennese MA 15[25] the DINÖ is publishing an estimate of new influenza infections for the city during influenza season.[26]. The "European Influenza Surveillance Scheme"[27] (abbr. EISS) does monitor the influenza levels in European countries and weekly publishes them on their website. Although this coverage is dependend of the national surveillance networks and coverage is often inconsistent. In Germany the "Arbeitsgemeinschaft Influenza"[28] (abbr. AGI) does monitor and evaluate the influenza activity – the AGI is financially supported by four pharmaceutic companies that are producing influenza vaccine.

When trying to compare figures from various sources there are quite a few problems one needs to face. Often the periods of surveillance vary, and almost all surveillance systems only cover the influenza season and not the activity during the whole year. Another problem is the incidences they are counting. Some schemes are counting "influenza like illnesses" (ILI) others "influenza and common flu" or "acute respiratory infections" (ARI). Some sources are providing only the number of laboratory confirmed samples others only extrapolations.

[24] Diagnostic Influenza Network Austria, see [W4] for web address

[25] The MA 15 (Viennese municipality department 15) is in charge of health and social issues.

[26] In the northern hemisphere the influenza season lasts for about 15 weeks during winter. See [W5, W4] for web addresses of DINÖ.

[27] see [W6] for web address

[28] Working group Influenza, see [W1] for web address

Since this thesis is written in cooperation with the "Hauptverband der österreichischen Sozialversicherungsträger"[29] (abbr. HVB) the author did have limited access to their data. Unfortunately this data does currently offer only very poor spatial and temporal resolution. This situation will hopefully change, with a new system in place. On the other hand the figures taken from the HVB are supposed to be very accurate. Thus it is possible to extrapolate and estimate data from other sources on basis of the HVB data.

The numbers of main diagnosis influenza (ICD group 10) and ICD-10 related deaths, that are transmitted to the HVB for the year 2005 are displayed in Table 4.3 (on page 60), including all cases in Austria and all cases in Vienna.

From these figures the higher death rate, as well as an increased number of diagnosis, amongst senior citizens is obvious. We can assume, that almost 100% of all influenza related deaths are recorded, since persons with severe health problems are very likely to consult a doctor. This is not necessarily the case for the disease itself, especially younger and middle aged men are known to rarely consult doctors. Thus it is necessary to verify the infection figures.

Rough Estimate of Death Rate

With the demographic data and the $\mathbb{P}$(diagnosis) and $\mathbb{P}$(death) from the HVB we can calculate the total number of incidences per age group (for Austria and Vienna in the year 2005), the results are presented in Table 4.4 (on page 60). The figures of the DINÖ and MA15 for the year 2005 can be taken from the graphics presented on the website of the institute of virology[30] (see also Fig. 4.3).

If we add up the number of estimated cases for the year 2005 (read out of these graphics), we do receive an approximate 350,000 infections (in spring and winter 2005).

One may assume that some 70-100% of influenza cases do occur during the monitored period of time (influenza season), which leads to a range of about 350,000 to 500,000 infections for the year 2005 for Vienna (equaling 23 - 33% of the population). This meets the estimates, which state that 5 - 20% of the population is infected every (non-pandemic) year.[31]

Dividing the number of infections onto the remaining weeks of the year we do get a range of 0 up to 6,000 infections/week (off-season). With these figures we can take a

[29] Main Association of Austrian Social Security Institutions

[30] see [W5] for web address

[31] see [W2]

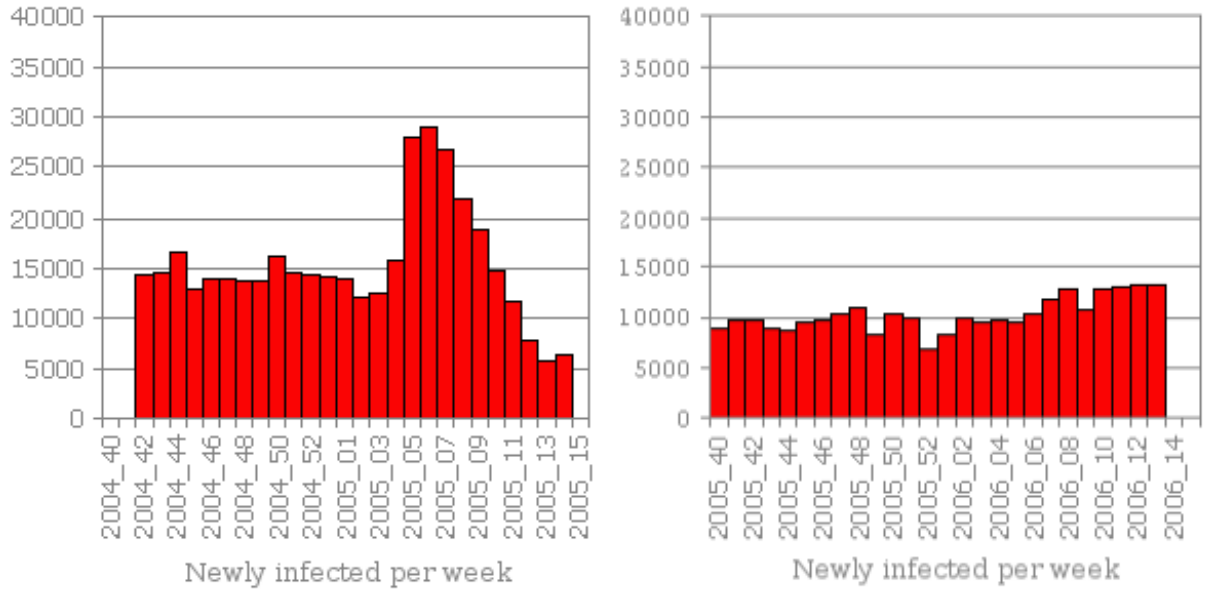

Figure 4.3: *Weekly number of new infections with influenza like illness (ILI), estimated by the Viennese MA15 for the seasons 2004/05 (left) and 2005/06 (right)*

shot at estimating the approximate probability – for our model – for the death of an agent, given illness and an age of over 60. This probability lies around 0.20%, based on the calculated figures (see Table 4.1). What we still lack, is the probability for an agent to get infected. This probability is very hard to obtain for several reasons. To start with, it is different for every strain of virus, thus dependend on the circulating strain(s). Secondly such a probability does not exist in literature (to the knowledge of the author). And even if it would exist in literature it could not be applied one-to-one onto the model.

% of Infections occurring during season	Diagnosed infections per year		Number of Diagnoses for age ≥ 60	Death Ratio (ill & ≥ 60) [based on: 456]
	estimate for Vienna	% HVB-reported [based on: 5,515]		
100 %	350,000	1.58 %	175,000	0.26 %
90 %	388,889	1.42 %	194,444	0.23 %
80 %	437,500	1.26 %	218,750	0.21 %
70 %	500,000	1.10 %	250,000	0.18 %

Table 4.1: *Probability for death provided that person is over the age of 60 and ill* $\mathbb{P}(\, death \mid age \geq 60,\, health\ state = ill\,)$

Age group	in percent	confidence interval ($p = 95\%$)	ratio
0-4	36%	[28,46]	1.09
5-18	62%	[57,67]	1.88
19-64	25%	[21,28]	0.76
≥ 65	21%	[15,27]	0.64
overall	33%	[30,37]	1.00

Table 4.2: *Average illness attack rates for different age groups (estimated for H2N2 virus; source: see [L25]).*

What can be done is to estimate the relation between the infection probabilities of different age groups. Although even this estimate remains relative, since the probabilities are changing with the type of circulating virus – and of course: it is just an estimate.

The estimated illness attack rates[32] for the different age groups, displayed in table 4.2 is taken from [L25], representing the H2N2 virus[33]. Surprisingly the attack rates are lower for the older population, and even the babies do show less susceptibility to the virus than children. This is interresting since the reports about the Spanish flu describe a similar attack behavior.

To receive a better (more realistic) infection behavior the model needs to be fitted to data from observations. As already mentioned, one would need infection data with high temporal and spatial resolution together with the age and – in the best case – even the residential location of the patient, to obtain the optimum results. Such data was unfortunately not available to the author. Thus the infection rates used within the model are (more or less) based on subjective judgment.

[32] illness attack rate = exposed people infected/exposed people

[33] The virus causing the "Asian flu" pandemic (1957-1958)

4.3 Experiments and Findings

The disease information from sections 4.1 and 4.2 enables us to conduct first experiments with the final model. The full MATLAB® source code of the model can be found in Appendix B.

Description of experiment result graphics in Fig. 4.5 through 4.8:

- Top left: Sub-populations within the system (x-axis: days, y-axis: number of individuals)
- Top middle: Infection-related sub-populations – infected with & without symptoms and dead (axes as before)
- Top right: Number of simultaneously infected individuals with the corresponding number of days (x-axis: number of individuals, y-axis: number of days)
- Second row: Ratio of infected babies (left), child care children (middle) and elementary school pupil (right) over the time of the simulation (x-axis: days, y-axis: number of individuals)
- Third row: Ratio of infected high school students (left), adults (middle) and senior citizens (right) over the time (in days) of the simulation (axes as before)

Color coding used in top left and top middle plot: blue – susceptible, pink – infected without symptoms, red – infected with symptoms, green – recovered, black – immunized. No color coding used for the other graphics.

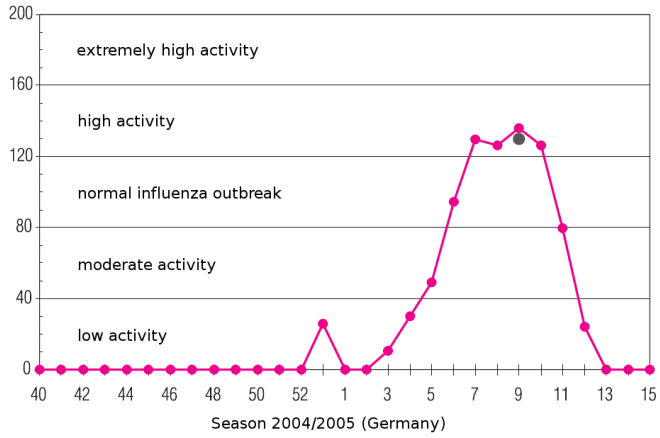

Figure 4.4: *The influenza activity in Germany during season 2004/05 according to the EISS-Index (x-axis: weeks; y-axis: EISS-Index) [Source of graphic: AGI-Report [L1]].*

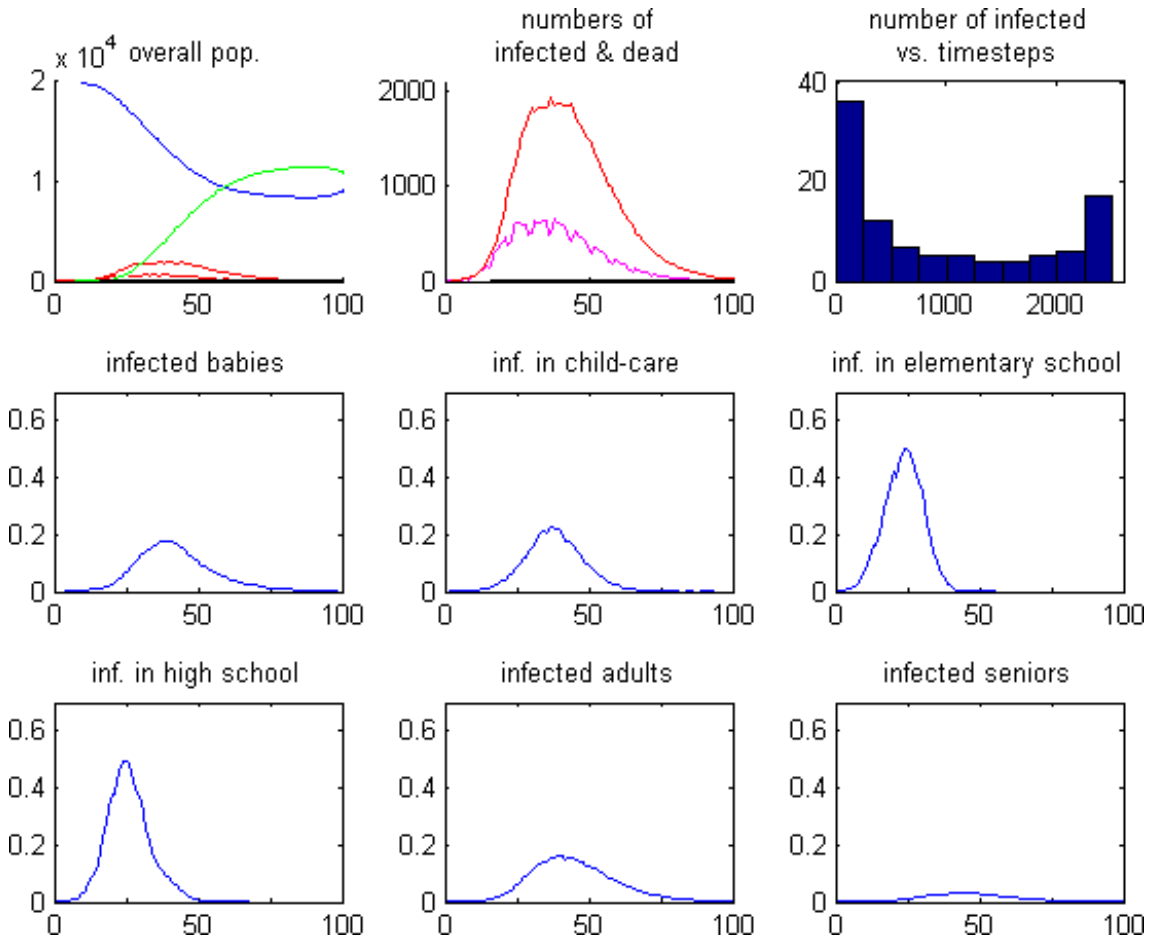

Figure 4.5: *Experiment with a neighborhood size of 250 individuals (For a description of the plotted figures and color coding see page 55).*

Since the available data does not allow for a reasonable fitting of the model – especially of $\mathbb{P}(\text{infection})$ and $\mathbb{P}(\text{death})$ – the result can only be understood as being academic. But there are still some remarkable things that can be found in the results of the experiments. One of the first things to notice is the shape of the curve described by the number of infected individuals. It does not show a single peak as in chapter 3 but does grow up to a point where a plateau of infected individuals is reached and held. This corresponds with the influenza activity of the season 2004/05 in Germany, described by the EISS-Index in Fig. 4.4.[34] A possible interpretation for this would be, that the saturation of infected agents reaches a certain level within the cellular automata which allows for a very good spread of the disease. It keeps on this level until potential hosts are getting fewer and thus the number of infected starts to decline.

[34]Graphic taken from [L1].

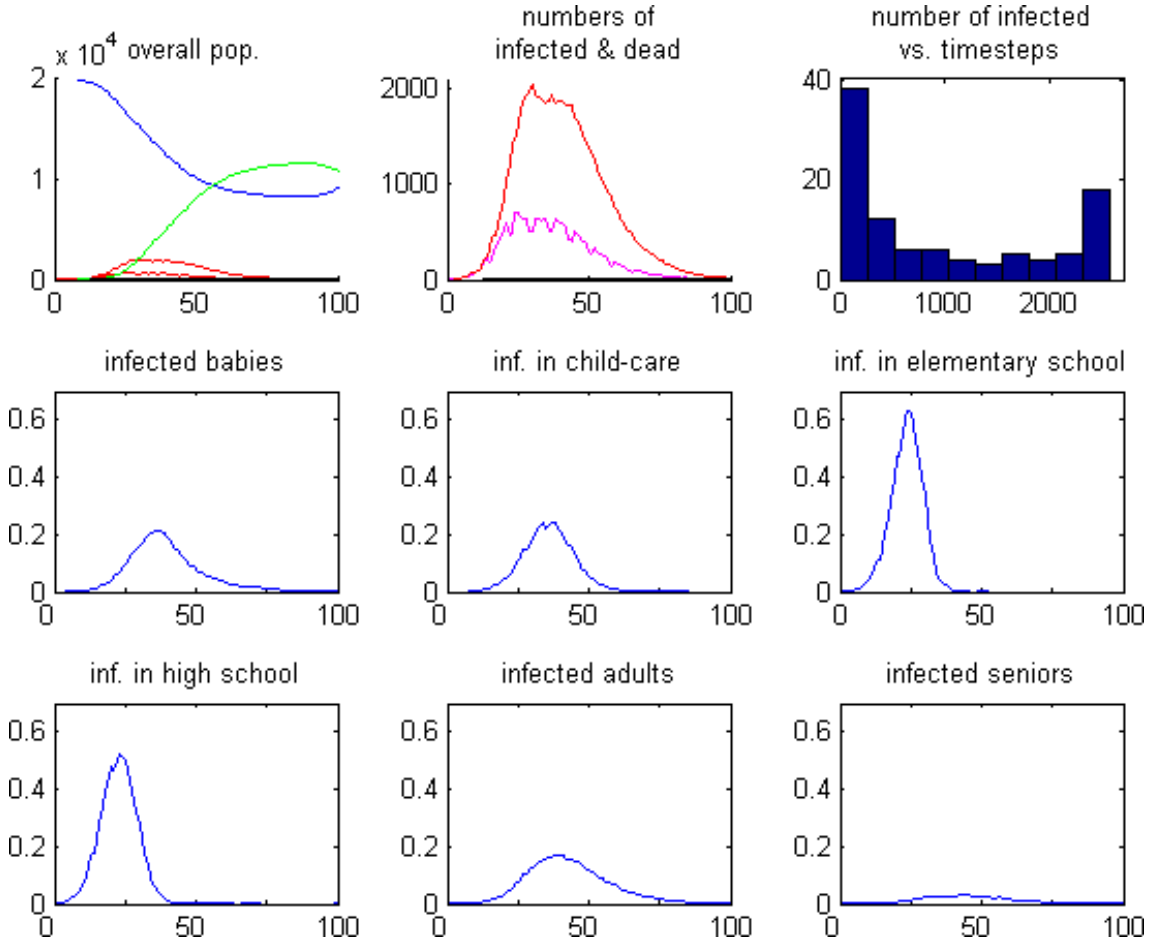

Figure 4.6: *Experiment with a neighborhood size of 100 individuals (For a description of the plotted figures and color coding see page 55).*

In Fig. 4.5 and 4.6 one can see the impact of changes to the neighborhood size. The neighborhood size was set to 250 and 100 individuals respectively. This did not change the (approximate) density within the CA, since the automata are programmed to fit their size according to a pre-defined density. The results of both experiments have been averaged over 10 and 5 runs respectively.[35] The influence of this change is most evident for the age groups of middle school and high school students (they also had the highest infection rates).

What is interresting to notice is the delay of the epidemic spread for different age groups. For example the peak of infected adults is reached at a point of time when there are hardly any infected students any more. This effect is reduced in Fig. 4.7

[35]The difference of repetitions is due to lack of time, since 10 consecutive repetitions took about 16 hours on the computer used.

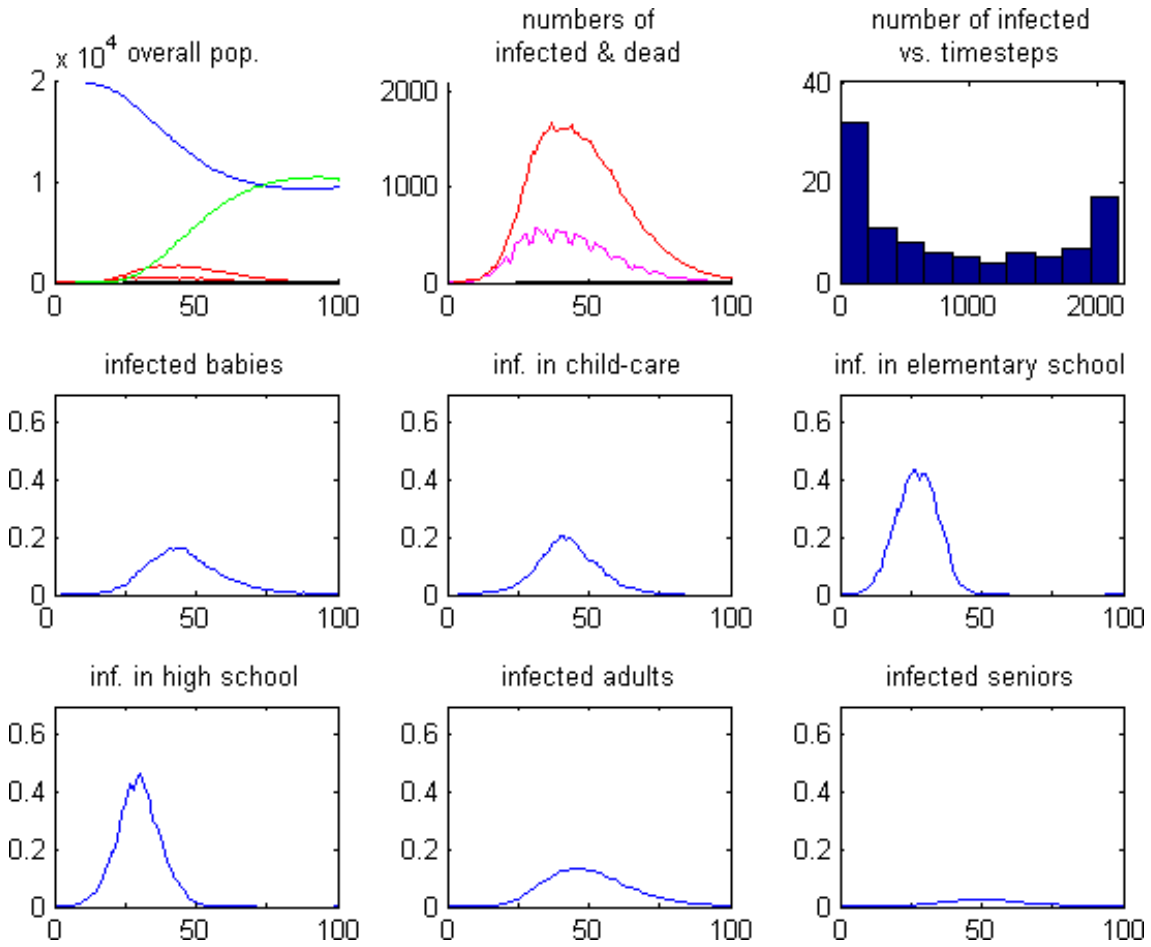

Figure 4.7: *Results for a run with an increased probability that agents stay at home if they become symptomatic (For a description of the plotted figures and color coding see page 55).*

in which the probability for an infected individual to stay at home is significantly raised. Also clearly visible in latter result is the lower maximum of the epidemic. A (not averaged) run with settings similar to Fig. 4.6 was conducted for a population of 100,000 individuals. The result of both runs (see Fig.4.8[36]) are fairly similar. But the curve of infected starts to raise approx. 10 days later in the larger population. This can be explained with the initially infected population which in both cases was 10 agents. For an larger total population (e.g. 1 million and more) an even stronger delay of the outbreak can be expected.

It would be very interresting to validate these effects (as far as possible) through data from real epidemics. If such a (temporal) coherence is proved in reality, this would

[36]One has to keep in mind that this experiment was not averaged. But because of the high number of agents the graphic is representative, although not as accurate as other (averaged) graphics.

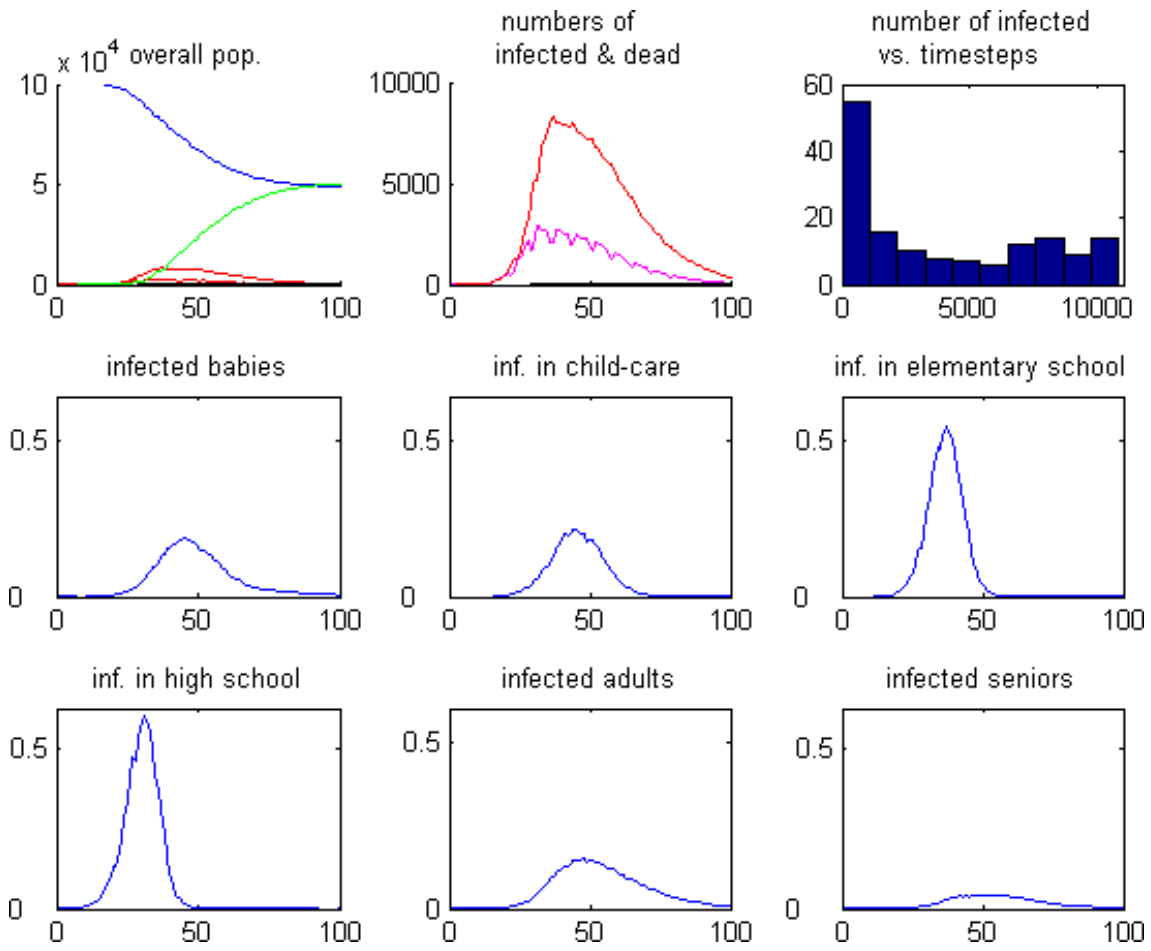

Figure 4.8: *Experiment with a population of 100,000 agents and settings similar to those in Fig. 4.6 (For a description of the plotted figures and color coding see page 55).*

offer new possibilities for the control of epidemics, e.g. vaccination appeals for special age groups (in the face of an acute threat or likely epidemic outbreak). Especially in the case that certain age groups would show increased risk of infection and/or death, such measures could be life saving.

Altogether this (still simple) model of an urban population leads to very interresting and partially unexpected results. We can reason that the quality of results should improve with better data becoming available and the model being expanded – if necessary.

With such results different prevention and response strategies can be verified and tested, leading to a better understanding of epidemic patterns. A deeper knowledge of these enables better planning and response, which again leads to health care improvements and cost savings.

	Main diagnosis		Discharge: Death	
	ICD-10 per 10,000 inhibitants (2005)			
Age group	Austria	Vienna	Austria	Vienna
Total population	43.63	36.34	4.16	3.71
40-59	19.60	18.76	0.75	0.71
60-79	81.14	57.80	7.70	5.72
over 80	362.68	217.31	60.47	52.88

Table 4.3: *Incidences of ICD-10 diagnoses and deaths for Austria and Vienna per age group*

Totals of ICD-10 diagnoses and deaths						
	Austria			Vienna		
Age group	Pop.in 10,000	Total diagnoses	Total deaths	Pop.in 10,000	Total diagnoses	Total deaths
under 20	180.12			30.01		
20-39	231.10			46.99		
40-59	222.12	4354	167	42.62	799	30
60-79	142.92	11634	1100	26.38	1525	151
over 80	31.00	10160	1881	5.77	1253	305
Sum	807.36	26148	3148	151.76	3578	486
Sum ≥ 60	174.03	21794	2981	32.15	2778	456
AllAges		35226	3361		5515	563
Difference of		13432	380		2736	107
AllAges vs ≥ 60		38%	11%		50%	19%

Table 4.4: *Total incidences of ICD-10 diagnoses and deaths for Austria and Vienna; calculation based on figures of Table 4.3 and estimates from Fig. 4.3.*

Chapter 5

Conclusion and Perspectives

Looking at the methods that have been and still are used for simulation and forecast of epidemics we found the situation to be unsatisfactory. Especially if considering the huge amount of data that is collected and available nowadays.

This data seems to make new approaches for modelling of epidemics possible. Thus we investigated two modelling methods – cellular automata and agent based systems. Both showed that they can be compared with the classical ODE approach when simulating a simple SIR epidemic. Hence we proceeded to find a possibility of combining these methods in order to build a hybrid model capable of simulation of influenza epidemics within inhomogeneous populations.

Setting up the model in the sense of simplifying reality and transforming it into a computable system is always the most time consuming part of modelling. In addition to this necessary reduction one must also judge which kind of data will be necessary to shape and parametrize the model and how much of this data is available. This was not any different for this model.

The structure of the model was gradually expanded. In this aspect the agent based "umbrella" that is steering the model proved extremely valuable, since it has a flexible structure allowing for easy changes and extensions without rewriting the whole program. It is possible to add more features and/or information to the agent by simply adding another variable to the look-up table.

The biggest problem that was encountered and could not be conquered satisfactory was the parametrization of the model. Fitting the model to reality would require more detailed data. Namely infection data with high spatial and temporal resolution, together with the infected person's age and – in the best case – also the patients residential location. Since influenza is not a notifiable disease this kind of data is not

publicly available – although it does exist.

Despite the fact that the model currently only resembles a very simplified social structure and is not "fitted" properly, the results are promising as we could see in the simulation outcome (see Fig. 4.5 through 4.8). The behavior of the modelled system is similar to real influenza patterns observed (see Fig. 4.4), and still offers a lot of possibilities for further enhancement.

The model is capable of simulating temporal infection coherences (e.g. disease spread by child care facilities, and infection development within age pools) which can partially be derived from common sense. But in addition to this, the individual structure is not lost as for example in ODE-systems (e.g. infected persons always return to their household constantly putting their family at risk). Thus the model enables us to test strategies for prevention of and response to epidemic outbreaks.

It seems likely that, based on this model, acceptable if not good results can be achieved – with the necessary information available. This because, as mentioned before, the model still offers a lot of potential for improvement. For example the neighborhood sub-system could be improved, as well as the situation of job less and elderly implemented in a more realistic way. In addition hospitals and/or medical facilities could be implemented, and so on.

Based on such a fitted model a lot of scenarios can be tested. Starting from effectiveness of countermeasures such as prophylactic vaccination up to quarantine in case of severe pandemic outbreaks. Investigations on the spread of the disease by different age pools (e.g. transmission via pupils) can be undertaken. And with better understanding of these kind of coherence the existing emergency plans and procedures can be evaluated and – if necessary – improved.

To embrace the results of this thesis, the combination of the two approaches can be seen as successful. It seems to be worthy of further investigation and study, where the presented model can be used as a base and be extended to suit the developer's needs.

Using such a model to test strategies and forecast the spread of epidemics, would entail the possibility for better planning and thus improved allocation of resources. Which again would make it possible to save human lives and money – health care- as well as disease related follow up costs.

Appendix A

Acknowledgments

This thesis was written in cooperation with the "Hauptverband der österreichischen Sozialversicherungsträger", the Main Association of Austrian Social Security Institutions.

I would like to express my thankfulness to Günther Zauner and Niki Popper, both from the Drahtwarenhandlung, who helped me very much while and with writing this thesis. Thanks also belongs to Günter Schneckenreither who was a great help optimizing the model's code, especially in terms of runtime.

This thesis was written using LaTeX and Kile on the Linux-based Ubuntu and Kubuntu operating systems. All of these programs/operating systems are free software.

Appendix B

Source Code of Model

Listing B.1: *infmod.m - Main program*

```
% autor: stefan emrich
% co-autor: guenter schneckenreither
% dateiname: infmod.m
%
% hauptprogramm zur simulation einer (influenza) epidemie. kombination des
% agent-based und cellular automata ansatzes.

% Nummerierung der Gesundheitsstatus:
% 1 ... Susceptible
% 2 ... Sus v.2 (noch nicht implementiert) -ev. Sus mit Prophlaxe
% 3 ... Sus v.3 (noch nicht implementiert) -ev. Sus mit Impfung
% 4 ... Reserve
% 5 ... Infiziert OHNE Symptome
% 6 ... Infiziert, symptomatisch
% 7 ... Reserve
% 8 ... Genesen (Immun)
% 9 ... Reserve
% 10... Tod


%% Funktion

function [eval_mat,krank_age]=infmod(fig_handler)
% fig_handler wird verwendet um plots alle repetitionen zu behalten ;)

warning off
% profile clear on

%-------------------------------------------------------------------------
% zur zeit kein output...
% [eval_mat, ag_mat, count_inf] =
%-------------------------------------------------------------------------

% zeit stoppen:
dauer_sim = cputime; % double
```

```
%% Eingangs Variablen
%*************************************************************************

% I) Festlegung der Population(en):
pop_size = int32(10000);
inf_size = int32(10);
rec_size = int32(0);

% II) Festlegung der CA-Belegungsdichten:
belegung_ap = double(0.3);     % Dichte am Arbeitsplatz
belegung_fz = double(0.1);     % Dichte in der Nachbarschaft/Freizeit

% III) Statistische Daten werden in 'infm_init.m' festgelegt!!
% 1) Groesse d. Altersgruppen
% 2) Haushaltsspez. Informationen
% 3) Infektionsraten (altersabhaengig)
% 4) Groesse der soz.Strukturen

% V) Festlegung der Laufzeit(en):
laufzeit = int32(100);
arbeitszeit = int16(100);
freizeit_week = int16(70); % Freizeit unter der Woche
freizeit_end = int16(140); % Freizeit am Wochenende

% VI) Infizierte Individuen bleiben ab Ende des Tages zuhause
% (fuer die Dauer der Kranheit) [aus: 0, ein:1]
%stayhome = int8(0);



%% LUT (Look Up Table) und globale Datenspeicher
%*************************************************************************

% Agenten Matrix:
global ag_mat
ag_mat = zeros(10,pop_size,'int32');

% Richtungsaenderung bei Kollision:
global ca_turn
ca_turn = [2 3 4 5 6 1; 6 1 2 3 4 5];

% Zaehler fuer Infektionen:
% Arbeitsplatz, Nachbarschaft und Familie getrennt!
global count_inf
count_inf = zeros(3,1,'int32');

% Vektor fuer altersbedingte Wahrsch. bei Inf. zuhause zu bleiben
global prob_stayhome
prob_stayhome = 0:130;

global pop_vektor
pop_vektor = zeros(1,5,'single');
```

```
%% Restliche Varaiblen und Speicherfleder
%**************************************************************************

% Evaluations Matrix zum Speichern der Daten:
eval_mat = zeros(laufzeit+1,5,'int32');
% Erste Zeile der Matrix:
eval_mat(1,:) = [pop_size-inf_size-rec_size, inf_size, int32(0), rec_size, int32(0)];
% Evaluierungsmatrix fuer Altersgruppen:
krank_age = zeros(laufzeit,6,'int32');
% Relativierte Evaluierungsmatrix :
krank_std = zeros(laufzeit,6,'single');

% Speicherplatz fuer die Anzahl der Nachbarschaften:
anz_nachb = int32(0);
% Speicherplatz fuer die Anzahl der Haushalte:
anz_haush = int32(0);
% Speicherplatz fuer die Anzahl der Arbeitsstaetten:
anz_work = int32(0);



%% Initialisierung der Agenten
%**************************************************************************

[anz_nachb,anz_haush,anz_work] = infm_init(pop_size,inf_size,rec_size);

%--------------------------------------------------------------------------
% ag_mat ist global definiert !
% die Infektionwahrscheinlichkeit muss durch 10000 dividiert werden!
%--------------------------------------------------------------------------



%% Simulation
%**************************************************************************

disp('--------------------------------------------------------------------------------');
disp('(infmod.m) SIMULATION BEGINNT...');
disp(' ');


for ind_zeit=int32(1):laufzeit
    tic

    if double(ind_zeit)/7 ~= ind_zeit/7

        % Arbeit
        %********************
        for ind_arbeitspl=int32(1):anz_work

            arbeiter = int32(find(ag_mat(6,:)==ind_arbeitspl & ag_mat(9,:)==int32(0)));
```

```
        if length(arbeiter)>0 % kann noch verbessert werden
            infm_ca(arbeitszeit,arbeiter,ind_zeit,belegung_ap,int8(1));
        end

    end % Arbeit
    freizeit = freizeit_week;
else
    freizeit = freizeit_end;
end

% Freizeit
%********************
for ind_nachb=int32(1):anz_nachb

    nachbarn = int32(find(ag_mat(5,:)==ind_nachb & ag_mat(9,:)==int32(0)));

    if length(nachbarn)>0  % kann noch verbessert werden
        infm_ca(freizeit,nachbarn,ind_zeit,belegung_fz,int8(2));
    end

end % Freizeit


% Familie
%********************

for ind_fam = int32(1):anz_haush

    familie = int32(find(ag_mat(4,:)==ind_fam));
    anz_fam_inf = single(length(familie(ag_mat(2,familie)==int32(5) |...
        ag_mat(2,familie)==int32(6))));

    if anz_fam_inf > 0

        fam_susc = familie(ag_mat(2,familie)==int32(1));

        infektionen = fam_susc(...
            rand(1,length(fam_susc)) < 1-(1-single(ag_mat(8,fam_susc))/...
            10000).^anz_fam_inf...
            );

        % Status wird auf infiziert gesetzt:
        ag_mat(2,infektionen) = int32(5);

        % Infektionszeitpunkt wird gespeichert:
        ag_mat(7,infektionen) = ind_zeit;

        % Infektion wird gezaehlt:
        count_inf(3) = count_inf(3)+int32(length(infektionen));

    end % if

end % Familie
```

```
    % Genesung
    %********************
    infm_genesung(ind_zeit);


    % Tod
    %********************
    infm_death(ind_zeit);



    % Daten speichern und Ausgabe
    %**************************************

    eval_mat(ind_zeit+1,:) = [...
        int32(length(find(ag_mat(2,:)==int32(1)))),...
        int32(length(find(ag_mat(2,:)==int32(5)))),...
        int32(length(find(ag_mat(2,:)==int32(6)))),...
        int32(length(find(ag_mat(2,:)==int32(8)))),...
        int32(length(find(ag_mat(2,:)==int32(10))))       ];

    krank_age(ind_zeit,1)=length(find( ag_mat(3,:)<=3 & (ag_mat(2,:)==5 |...
        ag_mat(2,:)==6)));
    krank_age(ind_zeit,2)=length(find( ag_mat(3,:)>3 & ag_mat(3,:)<6 &...
        (ag_mat(2,:)==5 | ag_mat(2,:)==6)));
    krank_age(ind_zeit,3)=length(find( ag_mat(3,:)>=6 & ag_mat(3,:)<10 &...
        (ag_mat(2,:)==5 | ag_mat(2,:)==6)));
    krank_age(ind_zeit,4)=length(find( ag_mat(3,:)>=10 & ag_mat(3,:)<18 &...
        (ag_mat(2,:)==5 | ag_mat(2,:)==6)));
    krank_age(ind_zeit,5)=length(find( ag_mat(3,:)>=18 & ag_mat(3,:)<60 &...
        (ag_mat(2,:)==5 | ag_mat(2,:)==6)));
    krank_age(ind_zeit,6)=length(find( ag_mat(3,:)>=60 & (ag_mat(2,:)==5 |...
        ag_mat(2,:)==6)));



    save('save_infmod', 'eval_mat', 'ag_mat', 'count_inf', 'krank_age');


    dauer=toc;

    disp([num2str(ind_zeit), '. Tag simuliert in ', num2str(dauer), ' Sekunden.']);

end % laufzeit



%% Ausgabe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dauer_sim = cputime-dauer_sim;
disp(' ');
```

```
disp(['Gesamte Laufzeit der Simulation: ', num2str(dauer_sim), ' Sekunden']);


disp(' ');
disp('STATISTIKEN:');
disp(' ');
disp(['Infektionen am Arbeitspl:          ',num2str(count_inf(1))]);
disp(['Infektionen in der Nachbarschaft: ',num2str(count_inf(2))]);
disp(['Infektionen zu Hause:             ',num2str(count_inf(3))]);
disp(['Infektionen gesamt:               ',num2str(sum(count_inf))]);
disp(' ');

% Ausgabe der Matrize mit den Pop-Groessen:
% eval_mat

disp('--------------------------------------------------------------------------------');

% Relativierung der Evaluierungsmatrix
krank_std(:,1)=single(krank_age(:,1))/pop_vektor(1);
krank_std(:,2)=single(krank_age(:,2))/pop_vektor(2);
krank_std(:,3)=single(krank_age(:,3))/pop_vektor(3);
krank_std(:,4)=single(krank_age(:,4))/pop_vektor(4);
krank_std(:,5)=single(krank_age(:,5))/pop_vektor(5);
krank_std(:,6)=single(krank_age(:,6))/pop_vektor(6);

clear krank_age;

figure(fig_handler);
clf;
subplot(331), hold on
subplot(331), plot(eval_mat(:,1),'b');
subplot(331), plot(eval_mat(:,2),'r');
subplot(331), plot(eval_mat(:,3),'r');
subplot(331), plot(eval_mat(:,4),'g');
subplot(331), plot(eval_mat(:,5),'k');
title('overall pop')
subplot(331), hold off
subplot(3,3,2), hold on
subplot(3,3,2), plot(eval_mat(:,2),'m');
subplot(3,3,2), plot(eval_mat(:,3),'r');
subplot(3,3,2), plot(eval_mat(:,5),'k');
title('infected and dead')
subplot(3,3,2), hold off
subplot(3,3,3), hist(double(eval_mat(:,2)+eval_mat(:,3)))
title('# of infected/timestep')
subplot(3,3,4), plot(krank_std(:,1))
axis([0 100 0 1])
title('infected babies')
subplot(3,3,5), plot(krank_std(:,2))
axis([0 100 0 1])
title('infected in k.garden')
subplot(3,3,6), plot(krank_std(:,3))
axis([0 100 0 1])
title('infected in elementary school')
```

```
subplot(3,3,7), plot(krank_std(:,4))
axis([0 100 0 1])
title('infected in high school')
subplot(3,3,8), plot(krank_std(:,5))
axis([0 100 0 1])
title('infected adults')
subplot(3,3,9), plot(krank_std(:,6))
axis([0 100 0 1])
title('infected seniors')

% Loeschung der globalen Variablen

clear global ca_shift
clear global ca_turn

% profile off viewer

end % of function
```

Listing B.2: *infm_init.m - Initialisation of Agents*

```
% autor: stefan emrich
% co-autor: guenter schneckenreither
% dateiname: infm_init.m
%
% initialisierung der agenten fuer die simulation.
% inklusive eingabe der demografie- und krankheits-parameter

function [anz_nachb,anz_hh,anz_work] = infm_init(pop_size,inf_size,rec_size)

%-------------------------------------------------------------------------
% ag_mat ist global definiert!
%-------------------------------------------------------------------------

global ag_mat
global prob_stayhome
global pop_vektor


disp(' ');
disp('------------------------------------------------------------------------------------------------------------------------ ');
disp('(infm_init.m) AGENTEN WERDEN INITIALISIERT ... ');
disp(' ');


tic


% Statsitische Eingaben (relative Werte)
%**************************************************************************

% Bevoelkerungsaufteilung nach Alter:
proz_sen = 0.21;     % Senioren (ueber 60 Jahre)
proz_erw = 0.61;     % Erwachsene (18-60)
proz_ms  = 0.08;     % Mittelschueler (10-18)
proz_vs  = 0.04;     % Volksschueler (6-10)
proz_kg  = 0.03;     % Kindergartenkinder (3-6)
% Babies (0-3) werden ueber Differenz gerechnet

% Altersspezifische Infektionsraten:
inf_rate_sen   = 0.0013; %0.002;
inf_rate_erw   = 0.0017; %0.0016;
inf_rate_ms    = 0.0025; %0.002;
inf_rate_vs    = 0.0025; %0.0024;
inf_rate_kg    = 0.003; %0.004;
inf_rate_baby  = 0.002; %0.006;

% Altersspezifische Wahrsch. bei Inf. zuhause zu bleiben:
p_sth_sen = 0.8;
p_sth_erw = 0.7;
p_sth_ms  = 0.75;
p_sth_vs  = 0.85;
p_sth_kg  = 0.93;
p_sth_baby= 0.99;
```

```
% Haushaltsinformation:
avg_hh_size = 2;     % Durchschn. HH-groesse in Personen
hh1 = 0.45;          % Haushalte mit 1 Person in %
hh2 = 0.3;           % usw...
hh3 = 0.13;          % usw
hh4 = 0.08;
hh5 = 0.04;          % HH mit mehr als 5 Personen

% Soziale Strukturen:
nachb_size  = 100;   % Nachbarsch.groesse (in Haushalten)
work_size   = 10;    % Arbeitspl.groesse
mschul_size = 375;   % Mittelschulsgr.
vschul_size = 250;   % Volksschulgr.
kgartn_size = 29;    % Kindergartengr.



% Variablen berechnen (absolute Werte)
%**************************************************************************

%--------------------------------------------------------------------------
% Damit die Agenten mit ihren Daten als 32-bit Integer gespeichert werden
% k nnen, mu  die Infektionswahrscheinlichkeit umgerechnet werden!
% F r die Berechnungen ist es aber n tzlich, dass pop_size als double
% vorleigt.
%--------------------------------------------------------------------------

pop_size=double(pop_size);

% Infektionsraten:
inf_rate_sen   = int32(inf_rate_sen  * 10000);
inf_rate_erw   = int32(inf_rate_erw  * 10000);
inf_rate_ms    = int32(inf_rate_ms   * 10000);
inf_rate_vs    = int32(inf_rate_vs   * 10000);
inf_rate_kg    = int32(inf_rate_kg   * 10000);
inf_rate_baby  = int32(inf_rate_baby * 10000);

% Anz.d Nachbarschaften:
anz_nachb = int32(ceil(pop_size/nachb_size));

% Individuen pro Alterspool:
anz_senioren = floor(pop_size * proz_sen);
anz_arbeiter = floor(pop_size * proz_erw);
anz_mschuler = floor(pop_size * proz_ms);
anz_vschuler = floor(pop_size * proz_vs);
anz_kgkinder = floor(pop_size * proz_kg);
anz_baby     = pop_size-(anz_senioren+anz_arbeiter+anz_mschuler+anz_vschuler...
    +anz_kgkinder);
pop_vektor = [anz_baby, anz_kgkinder, anz_vschuler, anz_mschuler, anz_arbeiter,...
    anz_senioren];

clear proz_sen proz_erw proz_ms proz_vs proz_kg;
```

```
% Anz d. Haushalte:
anz_hh  = ceil(pop_size/avg_hh_size); % wird erst sp ter in int32 umgewandelt
anz_hh1 = floor(anz_hh * hh1);
anz_hh2 = floor(anz_hh * hh2);
anz_hh3 = floor(anz_hh * hh3);
anz_hh4 = floor(anz_hh * hh4);
anz_hh5 = anz_hh-(anz_hh1+anz_hh2+anz_hh3+anz_hh4+1);

if (hh1+hh2+hh3+hh4+hh5)>1
    error('ACHTUNG! Summe der Haushalte groesser als 100 Prozent!');
end

clear hh1 hh2 hh3 hh4 hh5;

% Anz. d. Arbeitsplaetze (inkl. Schulen & Kindergaerten)
anz_arbeitsp = ceil(anz_arbeiter / work_size);
anz_mschulen = ceil(anz_mschuler / mschul_size);
anz_vschulen = ceil(anz_vschuler / vschul_size);
anz_kigarten = ceil(anz_kgkinder / kgartn_size);
anz_work     = int32(anz_arbeitsp+anz_mschulen+anz_vschulen+anz_kigarten);

clear work_size mschul_size vschul_size kgartn_size

% Vektor fuer Wahrsch. bei Inf. zuhause zu bleiben
for i=0:130
    if i>=60
        prob_stayhome(i+1)=p_sth_sen;
    elseif i>=18
        prob_stayhome(i+1)=p_sth_erw;
    elseif i>=10
        prob_stayhome(i+1)=p_sth_ms;
        elseif i>=6
        prob_stayhome(i+1)=p_sth_vs;
    elseif i>=3
        prob_stayhome(i+1)=p_sth_kg;
    else
        prob_stayhome(i+1)=p_sth_baby;
    end
end

clear p_sth_sen p_sth_erw p_sth_ms p_sth_vs p_sth_kg p_sth_baby

if anz_arbeiter < anz_hh
    error('ACHTUNG! Es existieren weniger Erwachsene als Haushalte!');
end
```

```
% Agenten initialisieren
%*************************************************************************


%-------------------------------------------------------------------------
% Generierung der Agenten mit:
% Ageneten-ID
% Alter (entsprechend demographischen Daten => eingabe oben)
% Inf. Wahrscheinlichkeit (altersabh ngig)
%-------------------------------------------------------------------------


ag_id = int32(1); % ag_id wird ab jetzt immer als in 32 gespeichert!


% Generierung der Erwachsenen
for i=1:anz_arbeiter
    ag_liste(ag_id).id = ag_id; % ist schon int32!
    ag_liste(ag_id).status = int32(1);
    ag_liste(ag_id).alter = int32(ceil(rand*41.99 + 17));
    % Jedem HH einen Erwachsenen!
    if i<=anz_hh
        ag_liste(ag_id).HHid = int32(i);
        ag_liste(ag_id).nachb = int32(ceil(i/nachb_size));
    else
        ag_liste(ag_id).HHid = int32(0);
        ag_liste(ag_id).nachb = int32(0);
    end
    ag_liste(ag_id).work = int32(floor(rand*anz_arbeitsp));
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_erw; % ist schon int32!
    ag_liste(ag_id).stay_home = int32(0);
        ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
% Generierung d. Senioren
for i=1:anz_senioren
    ag_liste(ag_id).id = ag_id;
    ag_liste(ag_id).status = int32(1);
    ag_liste(ag_id).alter = int32(ceil(abs(normrnd(9,11)))+59);
    ag_liste(ag_id).HHid = int32(0);
    ag_liste(ag_id).nachb = int32(0);
    ag_liste(ag_id).work = int32(0);  % vorl. annahme: senioren "arbeiten" zuhause
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_sen;
    ag_liste(ag_id).stay_home = int32(0);
    ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
% Generierung d. Mittelschueler
for i=1:anz_mschuler
    ag_liste(ag_id).id = ag_id;
    ag_liste(ag_id).status = int32(1);
```

```
    ag_liste(ag_id).alter = int32(floor(rand*7.999+10));
    ag_liste(ag_id).HHid = int32(0);
    ag_liste(ag_id).nachb = int32(0);
    ag_liste(ag_id).work = int32(ceil(rand*anz_mschulen + anz_arbeitsp));
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_ms;
    ag_liste(ag_id).stay_home = int32(0);
        ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
% Generierung d. Volksschueler
for i=1:anz_vschuler
    ag_liste(ag_id).id = ag_id;
    ag_liste(ag_id).status = int32(1);
    ag_liste(ag_id).alter = int32(floor(rand*3.999+6));
    ag_liste(ag_id).HHid = int32(0);
    ag_liste(ag_id).nachb = int32(0);
    ag_liste(ag_id).work = int32(ceil(rand*anz_vschulen + anz_mschulen +...
        anz_arbeitsp));
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_vs;
    ag_liste(ag_id).stay_home = int32(0);
    ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
% Generierung d. Kindergartenkinder
for i=1:anz_kgkinder
    ag_liste(ag_id).id = ag_id;
    ag_liste(ag_id).status = int32(1);
    ag_liste(ag_id).alter = int32(floor(rand*2.999+3));
    ag_liste(ag_id).HHid = int32(0);
    ag_liste(ag_id).nachb = int32(0);
    ag_liste(ag_id).work = int32(ceil(rand*anz_kigarten + anz_vschulen +...
        anz_mschulen + anz_arbeitsp));
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_kg;
    ag_liste(ag_id).stay_home = int32(0);
    ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
% Generierung d. Babies
for i=1:anz_baby
    ag_liste(ag_id).id = ag_id;
    ag_liste(ag_id).status = int32(1);
    ag_liste(ag_id).alter = int32(floor(rand*2.999));
    ag_liste(ag_id).HHid = int32(0);
    ag_liste(ag_id).nachb = int32(0);
    ag_liste(ag_id).work = int32(0);
    ag_liste(ag_id).infzeitp = int32(0);
    ag_liste(ag_id).inf_wahsch = inf_rate_baby;
    ag_liste(ag_id).stay_home = int32(0);
    ag_liste(ag_id).surviver = int32(0);
    ag_id = ag_id + 1;          % naechster Agent-ID
end
```

```
clear ag_id inf_rate_baby inf_rate_kg inf_rate_vs inf_rate_ms inf_rate_erw inf_rate_sen;
clear anz_baby anz_kgkinder anz_vschuler anz_mschuler anz_arbeiter anz_senioren;
clear anz_arbeitsp anz_mschulen anz_vschulen anz_kigarten;



% Zuweisung der Agenten in Haushalte und Nachbarschaften
%**************************************************************************

%--------------------------------------------------------------------------
% Waehrend der Generierung der Agenten wurde sichergestellt, dass jeder
% Haushalt einen Erwachsenen bewohner hat.
%--------------------------------------------------------------------------


hh_vec = zeros(1,anz_hh,'int32');


for i=1:anz_hh2
    zufl = ceil(rand*anz_hh);
    while hh_vec(zufl) ~= 0
        zufl = zufl+1;
        if zufl > anz_hh
            zufl=1;
        end
    end
    hh_vec(zufl) = 1;
end
for i=1:anz_hh3
    zufl = ceil(rand*anz_hh);
    while hh_vec(zufl) ~= 0
        zufl = zufl+1;
        if zufl > anz_hh
            zufl=1;
        end
    end
    hh_vec(zufl) = 2;
end
for i=1:anz_hh4
    zufl = ceil(rand*anz_hh);
    while hh_vec(zufl) ~= 0
        zufl = zufl+1;
        if zufl > anz_hh
            zufl=1;
        end
    end
    hh_vec(zufl) = 3;
end
for i=1:anz_hh5
    zufl = ceil(rand*anz_hh);
    while hh_vec(zufl) ~= 0
        zufl = zufl+1;
```

```
        if zufl > anz_hh
            zufl=1;
        end
    end
    hh_vec(zufl) = 4;
end

clear anz_hh5 anz_hh4 anz_hh3 anz_hh2 anz_hh1;

%????????????????????????????????
%disp(['fuer HH benoetigt: ', num2str(sum(hh_vec) + length(hh_vec))]);
%disp(['pop_size:          ', num2str(pop_size)]);
%????????????????????????????????

%-------------------------------------------------------------------------
% 'hh_vec' enthaelt nun die Anzahl der noch fehlenden Bewohner pro
% haushalt
%-------------------------------------------------------------------------

for i=1:anz_hh
    for j=1:hh_vec(i)
        hilfe = 0;
        zufl = ceil(rand*pop_size);
        while hilfe == 0
            if ag_liste(zufl).HHid == 0
                ag_liste(zufl).HHid = int32(i);
                ag_liste(zufl).nachb = int32(ceil(i/nachb_size));
                hilfe = 1;
            else
                zufl = zufl+1;
                if zufl > pop_size
                    zufl=1;
                end
            end
        end
    end
end

%-------------------------------------------------------------------------
%jetzt sind noch einige agenten ohne HH. diese geh ren alle den 5+ HHs
%zugewiesen. wie zu bewerkstelligen?
%-------------------------------------------------------------------------

hh_vec = find(hh_vec == 4);

for i=1:pop_size
    if ag_liste(i).HHid==0
        hilfe = ceil(rand * length(hh_vec));
        ag_liste(i).HHid = int32(hh_vec( hilfe ));
        ag_liste(i).nachb = int32(ceil( hilfe / nachb_size ));
    end
end

clear hh_vec;
```

```
% Infektions- bzw. Impfstatus setzten
%*************************************************************************

% Initialinfektion (Inf)
for i=1:inf_size
    j=ceil(rand*pop_size);
    while ag_liste(j).status~=1
        j=j+1;
        if (j>pop_size), j=1; end
    end
    ag_liste(j).status=int32(5);
end

% geimpfte Individuen (Rec)
for i=1:rec_size
    j=ceil(rand*pop_size);
    while ag_liste(j).status~=1
        j=j+1;
        if (j>pop_size), j=1; end
    end
    ag_liste(j).status=int32(8);
end



% Umformungen und Generieren der Agenten Matrix
%*************************************************************************

%anz_nachb=int32(anz_nachb); % wird oben schon als int32 gespeichert
%anz_work=int32(anz_work); % wird oben schon als int32 gespeichert
anz_hh=int32(anz_hh);

% Umformen der Agentenliste (fuer CAs):
ag_cell = struct2cell(ag_liste);
clear ag_liste
ag_temp = cell2mat(ag_cell);
clear ag_cell
ag_mat(:,:) = ag_temp(:,1,:);
clear ag_temp



% Status - Ausgabe

dauer=toc;

disp(['Initialisierung der Agenten wurde in ',num2str(dauer),' Sekunden ausgefuehrt.']);
disp(' ');
disp('ERRECHNETE AUSGANGSDATEN:');
disp(' ');
disp(['anz.d. bewohner:        ', num2str(max(ag_mat(1,:)))]);
```

```
disp(['hoechster status:       ', num2str(max(ag_mat(2,:)))]);
disp(['kleinster status:       ', num2str(min(ag_mat(2,:)))]);
disp(['max. alter:             ', num2str(max(ag_mat(3,:)))]);
disp(['min. alter:             ', num2str(min(ag_mat(3,:)))]);
disp(['anz.d. haushalte:       ', num2str(max(ag_mat(4,:)))]);
disp(['kleinst HausH ID:       ', num2str(min(ag_mat(4,:)))]);
disp(['anz.d. nachbarschaften: ', num2str(max(ag_mat(5,:)))]);
disp(['kleinste nachb.sch. ID: ', num2str(min(ag_mat(5,:)))]);
disp(['anz.d. arb.plaetze:     ', num2str(max(ag_mat(6,:)))]);
disp('------------------------------------------------------------------------------------------------------------------------ ');
disp(' ');
disp(' ');
disp(' ');


end % function
```

Listing B.3: *infm_ca.m - CA implementation*

```
% autor: stefan emrich
% co-autor: guenter schneckenreither
% dateiname: infm_ca.m
%
% der programmteil der die zellulaeren automaten beschreibt


function infm_ca(laufzeit,id_teilnehmer,zeit_glob,ca_dichte,umfeld_typ)

% Datentypen: int16, int32, int32, double, int8

%% GLOBALE VARIABLEN
global ag_mat % int32
global count_inf % int32
global ca_turn % double


%% 'ca_matrix' GENERIEREN
partikel_anz = double(length(id_teilnehmer));
ca_size = int32(ceil(sqrt(partikel_anz/(6*ca_dichte))));

ca_matrix = zeros(ca_size,ca_size,6,'int32');
ca_matrix(1:partikel_anz) = id_teilnehmer(:);
ca_matrix(1:ca_size*ca_size*6) = ca_matrix(randperm(ca_size*ca_size*6));

clear partikel_anz ca_dichte;


%% RESTLICHE VARIABLEN UND SPEICHERPLATZ
ind_coll_zeile = zeros(50,1,'double');
ind_coll_spalte = zeros(50,1,'double');

id_susc = zeros(length(id_teilnehmer),1,'int32');
id_infektionen = zeros(length(id_teilnehmer),1,'int32');
anz_inf = sparse(0);

%% CA STARTEN

for zeit_ca = int16(1):laufzeit

%% INFEKTIONEN

    % ID der susceptibles:
    id_susc = id_teilnehmer( ag_mat(2,id_teilnehmer)==1 )';

    % Anzahl der infected in jeder Zelle auf dem Gitter:
    anz_inf = sparse(histc(int8(ismembc(ca_matrix,id_teilnehmer(...
        ag_mat(2,id_teilnehmer)==5 | ag_mat(2,id_teilnehmer)==6) ...
        )),int8(1),3));

    % ZWISCHEN SCHRITTE:
    % 1) 'int32(find(ismembc(ca_matrix,id_susc)))'
    % liefert die 3-dim linearen Indizes der Positionen der susceptibles
```

```
% 2) 'rem( index - 1, ca_size*ca_size) + 1'
% berechnet die 2-dim linearen Indizes aus den 3-dim Indizes
% 3) 'rem( int32(find(ismembc(ca_matrix,id_susc))) - 1, ca_size*ca_size) + 1'
% liefert also die (2-dim linearen Indizes der) Positionen der susc auf
% dem Gitter
% 4) 'full(anz_inf( positionen_der_susceptibles ))'
% liefert die Anzahl der infected in den Zellen der susceptilbes

% ID der susceptilbes, die infiziert werden:
id_infektionen = id_susc( ...
    rand(length(id_susc),1) < ...
        1-(1-double(ag_mat(8,id_susc)')/10000).^ ...
        full(anz_inf(rem(int32(find(ismembc(ca_matrix,id_susc))) - 1,...
        ca_size*ca_size) + 1)) ...
    );

ag_mat(2,id_infektionen) = int32(5);
ag_mat(7,id_infektionen) = zeit_glob;
count_inf(umfeld_typ) = count_inf(umfeld_typ) + int32(length(id_infektionen));
ag_mat(9,id_infektionen) = infm_stayhome(id_infektionen);



%% KOLLISIONEN

[ind_coll_zeile,ind_coll_spalte] = ...
    find( ...
        (...
        logical(histc(ca_matrix,int32(0),3)==3) & ...
        logical(ca_matrix(:,:,1))==logical(ca_matrix(:,:,3)) & ...
        logical(ca_matrix(:,:,1))==logical(ca_matrix(:,:,5))...
        ) | (...
        logical(histc(ca_matrix,int32(0),3)==4) & ...
        logical(ca_matrix(:,:,1))==logical(ca_matrix(:,:,4)) & ...
        logical(ca_matrix(:,:,2))==logical(ca_matrix(:,:,5))...
        ) ...
    );

for ind = int8(1):int8(length(ind_coll_zeile))
    ca_matrix(ind_coll_zeile(ind),ind_coll_spalte(ind),:) = ...
        ca_matrix(ind_coll_zeile(ind),ind_coll_spalte(ind),ca_turn(1+...
        round(rand),:));
end

%% LATTICE UPDATE


ca_matrix(:,:,1) = ca_matrix(:,[ca_size,1:ca_size-1],1);
ca_matrix(:,:,2) = ca_matrix([2:ca_size,1],[ca_size,1:ca_size-1],2);
ca_matrix(:,:,3) = ca_matrix([2:ca_size,1],:,3);
ca_matrix(:,:,4) = ca_matrix(:,[2:ca_size,1],4);
ca_matrix(:,:,5) = ca_matrix([ca_size,1:ca_size-1],[2:ca_size,1],5);
ca_matrix(:,:,6) = ca_matrix([ca_size,1:ca_size-1],:,6);
```

```
%% CA ENDE

% Visualisierung:
%{
    if umfeld_typ==2
        infm_ca_lattice(ca_matrix,ca_size);
        %pause;
    end
 %}

end % zeit_ca


end % function
```

Listing B.4: *infm_ca_lattice.m - CA subroutine*

```
% Fuer die visualisierung notwendiges teilprogramm

function infm_ca_lattice(ca,ca_size);
G=-100*ones(ca_size*3,ca_size*4);
for zeile = 1:ca_size
    for spalte = 1:ca_size
        G((zeile-1)*3+1,(spalte+zeile-1)*2+1)=ca(zeile,spalte,3);
        G((zeile-1)*3+2,(spalte+zeile-1)*2+1)=ca(zeile,spalte,4);
        G((zeile-1)*3+3,(spalte+zeile-1)*2+1)=ca(zeile,spalte,5);
        G((zeile-1)*3+1,(spalte+zeile-1)*2+2)=ca(zeile,spalte,2);
        G((zeile-1)*3+2,(spalte+zeile-1)*2+2)=ca(zeile,spalte,1);
        G((zeile-1)*3+3,(spalte+zeile-1)*2+2)=ca(zeile,spalte,6);
    end
end
imagesc(G,[900 1000]);
drawnow;
end
```

Listing B.5: *infm_stayhome.m - 'Stayhome' routine*

```
% Routine die bestimmt ob ein infizierter Agent w hrend der Krankheitsdauer
% zu Hause bleibt, oder weiterhin einen normalen tagesablauf hat

function [output] = infm_stayhome(id_infektionen)

    global ag_mat
    global prob_stayhome

    age_inf = ag_mat(3,id_infektionen);
    output  = (rand < prob_stayhome(age_inf + 1));


end %funtion
```

Listing B.6: *infm_genesung.m - Recovery routine*

```
% Routine die zur Genesung & Krankheitsentwicklung innerhalb des
% Infektionsmodells dient

function infm_genesung(zeit)

global ag_mat

    % Vector mit allen erkrankten Individuen:
    vec_id = find(ag_mat(2,:)==5);
    inf_id = find(ag_mat(2,:)==6);



    % Ausbruch der Symptome bei erkrankten Individuen (probablisitisch)
    for i=vec_id
        if ((zeit - ag_mat(7,i)) >= 2)
            ag_mat(2,i) = 6;
            ag_mat(7,i) = zeit;
            ag_mat(9,i) = 0;
        end
    end



    % Genesung der erkrankten Individuen (probablisitisch)
    for i=inf_id
        if ( (zeit - ag_mat(7,i)) >= (abs(normrnd(6,1)-4)+4) )
            ag_mat(2,i) = 8;
            ag_mat(7,i) = zeit;
            ag_mat(9,i) = 0;
        end
    end



    % Verlust der Immunitaet
    rec_id = find(ag_mat(2,:)==8);

    for i=rec_id
        if (zeit-ag_mat(7,i) >= 75)  % alternativ: normrnd(25,3))
            ag_mat(2,i) = 1;
            ag_mat(10,i) = 0; % Bei neuerlicher Erkrankung ist Sterberisiko
        end                       % wieder normal
    end

end
```

Listing B.7: *infm_death.m - Death routine*

```
% Routine die die Todesfaelle innerhalb des Infektionsmodells berechnet

function infm_death(zeit)

global ag_mat

  if (zeit/4) == round(zeit/4) % ausfuehrungn nur nach jedem 4 zeitschritt
    % zecks laufzeit. ermoeglicht ausserdem bessere verbreitung der krankheit

        % Vektor mit allen kranken Individuen:
    inf_id = find(ag_mat(2,:)==6);

    % Tod der erkrankten Individuen (probabilistisch & altersspez.)
    for i=inf_id
        if (ag_mat(3,i)>=80 && ag_mat(10,i)==0 && rand<0.025)
            ag_mat(2,i) = 10;    % Status fuer Tod (sicherheitshalber hoch :)
            ag_mat(7,i) = zeit; % Todeszeitpunkt
            ag_mat(9,i) = 1;     % Individuum wird nicht mehr in CA genommen
        elseif (ag_mat(3,i)>=60 && ag_mat(10,i)==0 && rand<0.02)
            ag_mat(2,i) = 10;
            ag_mat(7,i) = zeit;
            ag_mat(9,i) = 1;
        else
            ag_mat(10,i) = 1;    % Individuum kann an DIESER Erkrankung nicht
        end                      % mehr sterben
    end

  end %of if(zeit/4)==...

end % of function
```

Bibliography

[L1] Arbeitsgemeinschaft Influenza, *Abschlussbericht der Influenzasaison 2004/05*, Berlin, 2005

[L2] Anderson, May, *Population Biology of Infectious Diseases*, Springer, 1982

[L3] Bandini, Pavesi, *Simulation of Pesticide Percolation in the Soil Based on Cellular Automata*, International Environmental Modelling and Software Society, http://www.iemss.org/iemss2002/proceedings/pdf/volume%20tre/258_pavesi.pdf

[L4] Baylei Norman T. J. , *The Mathematical Theory of Infectious Diseases and its Applications*, Charles Griffing & Company LTD, 1975

[L5] Behrens Doris, *Influenza Report 2006*, (www.InfluenzaReport.com)

[L6] Brenner Reinhard, *Methoden zur Monte-Carlo-Simulation und deren Implementierung*, Diploma-Thesis, Vienna University of Technology, 1995

[L7] Bridges, Thompson et al., *Effectiveness and Cost-Benefit of Influenza Vaccination of Healthy Working Adults*, The Journal of American Medical Association, 2000

[L8] Bundesministerium für Gesundheit und Frauen, *Influenza Pandemieplan - Strategie für Österreich, 3.Auflage November 2006*, Hausdruckerei des BMGF, 2006

[L9] Cliff, Andrew D., *Atlas of disease distributions*, Blackwell, 1988

[L10] Diekmann, Heesterbeek, Metz, *The Legacy of Kermack and McKendrick*, Cambridge University Press, 1995

[L11] Diekmann, Heesterbeek, *Mathematical Epidemiology of Infectious Diseases*, John Wiley & Son Ltd., 2000

[L12] Deutsch, Dormann *Cellular automaton modeling of biological pattern formation*, Birkhäuser, 2005

[L13] Dowdle Walter R., *Influenza Pandemic Periodicity, Virus Recycling, and the Art of Risk Assessment*, Emerging Infectious Diseases, 2006 (http://www.cdc.gov/ncidod/EID/vol12no01/05-1013.htm)

[L14] Emrich Gerhard, *Modellansätze zur Kontrolle von Epidemien*, Diploma-Thesis, Technische Universität Wien, 1978

[L15] Hecht Charles E., *Statistical Thermodynamics and Kinetic Theory*, W.H. Freeman and Company, New York, 1990

[L16] Foster, Talsma et al.,*Influenza Vaccine Effectiveness in Preventing Hospitalization for Pneumonia in the Elderly*, American Journal of Epidemiology, 1992

[L17] Gardner M. *The fantastic combinations of John Conway's new solitaire game "Life"*, Scientific American, 1970

[L18] Gilbert, Terna, *How to Build and Use Agent-Based Models in Social Science*, Mind & Society, 1, 2000, Vol.1, pp.57-72

[L19] Hegselmann, Flache, *Understanding Complex Social Dynamix: A Plea for Cellular Automata Based Modelling*, Journal of Artificial Societies and Social Simulation, vol.1, no.3, 1998 (http://jasss.soc.surrey.ac.uk/1/3/1.html)

[L20] Hegselmann, Flache, *Do Irregular Grids make a Difference? Relaxing the Spatial Regularity Assumption in Cellular Models of Social Dynamics*, JASSSJournal of Artificial Societies and Social Simulation, 2001

[L21] Hötzendorfer, Breitenecker, Popper, *Cellular Automata Modelling for Epidemic Dynamics of SIR Type*

[L22] Jennings, Sycara, Wooldridge, *A Roadmap of Agent Research and Development (Autonomous Agents and Multi-Agent Systems)*, Kluwer Academic Publishers, 1998

[L23] Kansal et al., *Simulated Brain Tumor Growth Dynamics Using a Three-Dimensional Cellular Automaton*, Academic Press, 2000

[L24] Kilbourne Edwin D., *Influenza Pandemics of the 20th Century*, Emerging Infectious Diseases, 2006 (http://www.cdc.gov/ncidod/EID/vol12no01/05-1254.htm)

[L25] Longini, Halloran, Nizam, Yang, *Longini, Jr., M. Elizabeth Halloran, Azhar Nizam, and Yang Yang*, American Journal of Epidemiology, 2004

[L26] Lustick J., *Agentbased modelling of collective identity: testing constructivist theory*, JASSS, 2000

[L27] Reid, Taubenberger, Fanning, *The 1918 Spanish influenza:integrating history and biology*, Microbes and Infection, 2001

[L28] Roli, Zambonelli, *On the Emergence of Macro Spatial Structures in Dissipative Cellular Automata, and its Implications for Agentbased Distributed Computing*, http://www.sci.unich.it/ aroli/pubs/dca.pdf

[L29] Schelling Thomas, *Micromotives and Macrobehaviour*, Norton, 1978

[L30] Situngkir Hokky , *Epidemiology through Cellular Automata - Case of Study: Avian Influenza in Indonesia*, 2004

[L31] Statistik Austria, *Kindertagesheimstatistik 2005/2006*, Statistik Austria, 2006

[L32] Statistik Austria, *Wohnsituation der Bevölkerung - Ergebnisse der Volks-, Gebäude- und Wohnungszählung 2001*, Statistik Austria, 2006

[L33] Statistik Austria, *Wohnungen - Ergebnisse der Wohnungserhebung im Mikrozensus Jahresdurchschnitt 2004*, Statistik Austria, 2005

[L34] Wolf-Gladrow Dieter A., *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*, Springer, 2000

[L35] Wolfram Stephen, *A new kind of science*, B&T, 2002

Web - References

[W1] AGI - Arbeitsgemeinschaft Influenza,
http://influenza.rki.de/

[W2] CDC - Center for disease control and prevention ,
http://www.cdc.gov/

[W3] Cellular automata optimization,
http://cell-auto.com/optimisation/index.html

[W4] DINÖ - Diagnostic Influneza Network Austria,
http://www.influenza.at/

[W5] DINÖ at the Institute of Virology, Medical University of Vienna,
http://www.virologie.meduniwien.ac.at/home/virus-epidemiologie/dinoe/lang_1-content.html

[W6] EISS - European Influenza Surveillance Scheme,
http://www.eiss.org

[W7] HVB - Hauptverband der österreichischen Sozialversicherungen (Main Association of Austrian Social Security Institutions),
http://www.sozialversicherung.at

[W8] Netdoktor,
http://www.netdoktor.at/

[W9] Statistik Austria,
http://www.statistik.at

[W10] WHO - World Health Orgnization,
http://www.who.int/

[W11] Wikipedia on Hippocrates,
http://en.wikipedia.org/wiki/Hippocrates