Theoretical Biophysics
A Computational Approach
Concepts, Models, Methods and Algorithms
Reaction-diffusion

Dieter W. Heermann

April 7, 2020

Heidelberg University

Table of Contents

Introduction

Introduction I

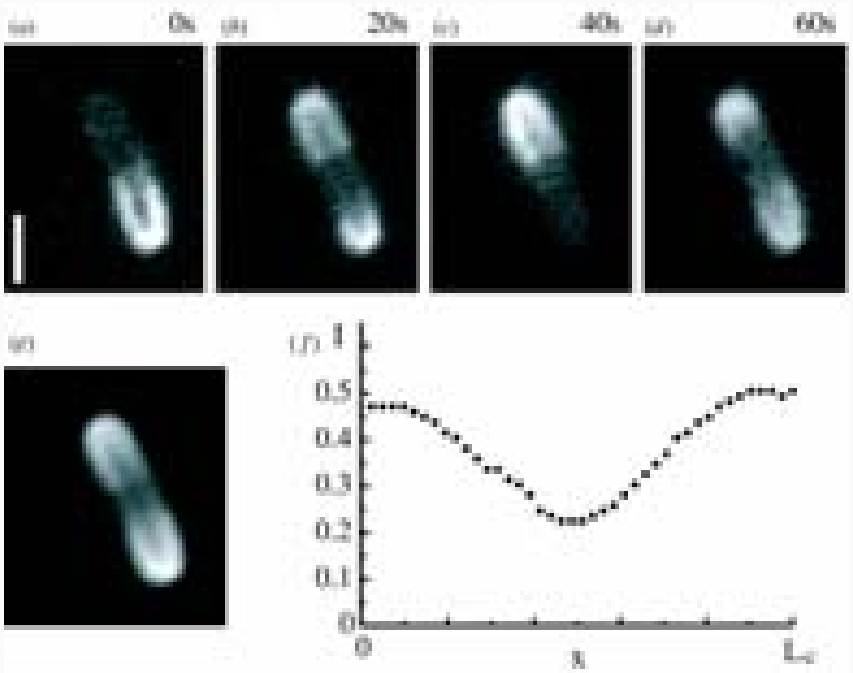

Figure 1: Image taken from [1]: Oscillations of MinD GFP in E. coli. (a)-(d) Fluorescence images of MinD GFP in a cell at subsequent time points separated by 20 s. (e) Time average of all frames during one oscillation period. Two subsequent frames are separated by 1 s. (f) Fluorescence intensity I obtained from a line scan of the fluorescence signal in (e). The background signal has been subtracted from the total signal which has then been rescaled with the maximum intensity during the oscillation. The slight asymmetry is due to bleaching during the observation period. Scale bar: $1\mu m$. The cell length is $L_c = 2.3\mu m$.

Introduction II

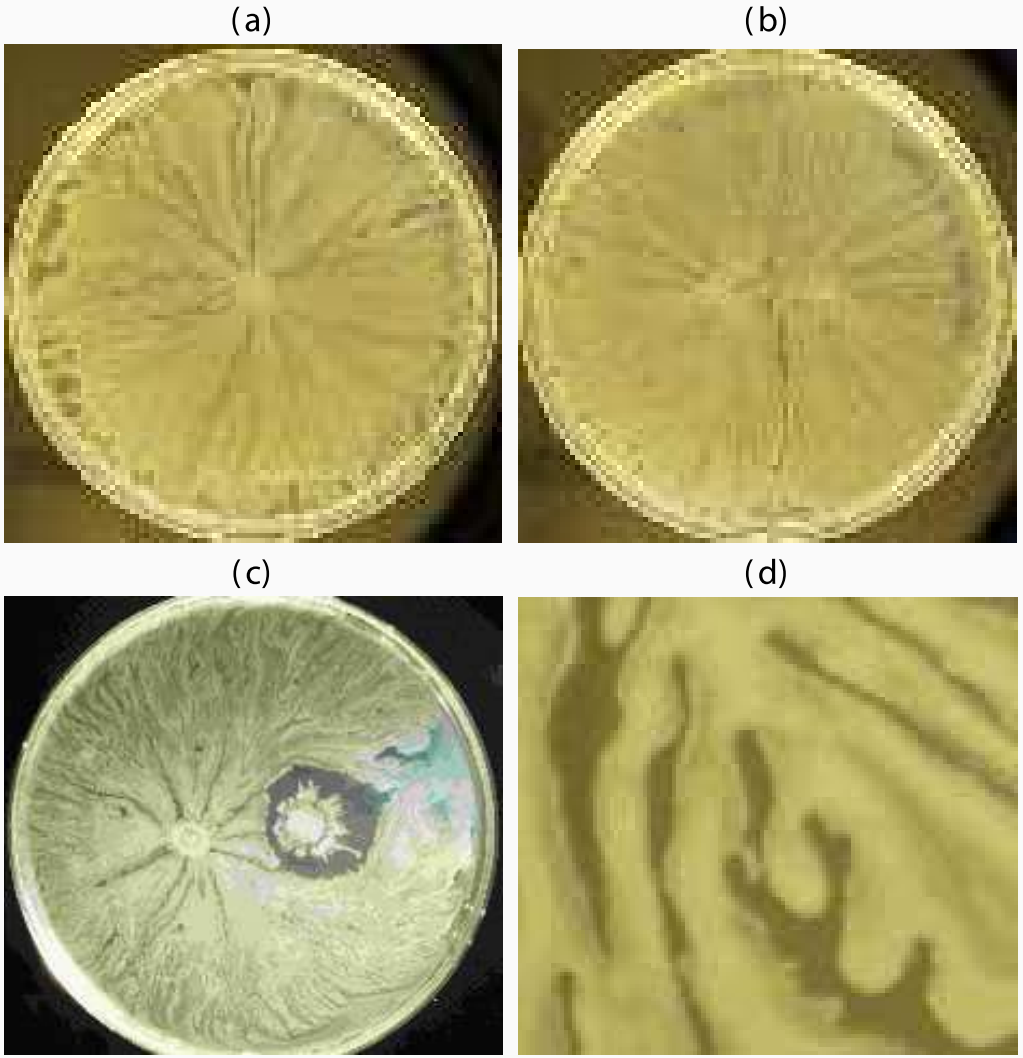

Figure 2: Image taken from [1]: .

Reaction-Diffusion Equation

Reaction-Diffusion Equation I

Let $P(\mathbf{x}, t)$ denote for example a population at time t and position $\mathbf{x}$. The population can change as follows:

- the individual particles can move around
- they may produce new indivuduals or kill existing individuals
- other causes

We shall assume Ficks law

$$J(\mathbf{x}, t) = -d(\mathbf{x})\nabla_{\mathbf{x}} P(\mathbf{x}, t) \tag{1}$$

where J is the flux and $d(\mathbf{x})$ is the diffusion coefficient at $\mathbf{x}$.

Assume the rate of change of the density function due to *other causes* is $f(\mathbf{x}, t, P)$, the *reaction rate* . We use the balance law to derive a differential equation. For this we choose a region O. Then the total population in O is

$$\int_O P(\mathbf{x}, t) d\mathbf{x} \tag{2}$$

and the rate of change of the total population is

$$\frac{d}{dt}\int_O P(\mathbf{x},t)d\mathbf{x}\ . \tag{3}$$

The net growth of the population inside the region O is

$$\int_O f(\mathbf{x},t,P(\mathbf{x},t))d\mathbf{x} \tag{4}$$

and the total out flux is

$$\int_{\partial O} \mathbf{J}(\mathbf{x},t)\cdot\mathbf{n}(\mathbf{x})dS\ . \tag{5}$$

The balance law implies

$$\frac{d}{dt}\int_O P(\mathbf{x},t)d\mathbf{x} = -\int_{\partial O} \mathbf{J}(\mathbf{x},t)\cdot\mathbf{n}(\mathbf{x})d\mathbf{x} + \int_O f(\mathbf{x},t,P(\mathbf{x},t))d\mathbf{x}\ . \tag{6}$$

Since

$$\int_{\partial O} \mathbf{J}(\mathbf{x},t)\cdot\mathbf{n}(\mathbf{x})dS = \int_O div\,\mathbf{J}(\mathbf{x},t)d\mathbf{x} \tag{7}$$

it follows

$$\int_O \frac{\partial}{\partial t}P(\mathbf{x},t)d\mathbf{x} = \int_O \left[div\left(d(\mathbf{x})\nabla_{\mathbf{x}}P(\mathbf{x},t)\right) + f(\mathbf{x},t,P(\mathbf{x},t))\right]d\mathbf{x}\,. \tag{8}$$

Since O is arbitrary we have

$$\frac{\partial}{\partial t}P(\mathbf{x},t) = \left[div\left(d(\mathbf{x})\nabla_{\mathbf{x}}P(\mathbf{x},t)\right) + f(\mathbf{x},t,P(\mathbf{x},t))\right] \tag{9}$$

with the diffusion term and the reaction term. If we assume that the diffusion is not space dependent $d(\mathbf{x}) = D$ we find

$$\frac{\partial}{\partial t}P(\mathbf{x},t) = D\Delta P(\mathbf{x},t) + f(\mathbf{x},t,P(\mathbf{x},t)) \tag{10}$$

the *reaction diffusion equation*.

If we disregard the diffusion term we obtain

Reaction-Diffusion Equation IV

$$\frac{\partial P}{\partial t} = f(t, P) \tag{11}$$

where $P = P(t)$. Hence we recover the typical population models if we assume

- $f(P) = kP$ Malthus linear growth
- $f(P) = kP(1 - P/N)$ logistic growth.

Taking spatial inhomogenities into account we get for the Malthus case

$$\frac{\partial P}{\partial t} = D\Delta P(\mathbf{x}, t) + kP \tag{12}$$

and

$$\frac{\partial P}{\partial t} = D\Delta P(\mathbf{x}, t) + kP(1 - P/N) \tag{13}$$

for the logistic growth that describe a spatially distributed population which satisfies general growth pattern.

We rewrite the above two equations as

$$\frac{\partial P}{\partial t} = D\Delta P(\mathbf{x}, t) + Pg(P)\ . \tag{14}$$

What is left is to specify the boundary conditions. Suppose we have a closed system. For each individual involved, if $\mathbf{J}(\mathbf{x}, t)$ is the flux of the individual, then the flux across a boundary point $\mathbf{x}$ is $\mathbf{J}(\mathbf{x}, t) \cdot \mathbf{n}(\mathbf{x})$. If we assume Ficks law, then for a closed system, at each boundary point $\mathbf{x}$

$$\nabla u(\mathbf{x}, t) \cdot n(\mathbf{x}) = 0 \tag{15}$$

where $u(\mathbf{x}, t)$ is the concentration. Therefore a well-posed closed reaction diffusion equation is a *initial value boundary problem*

$$\begin{aligned} \tfrac{\partial}{\partial t} u &= D\Delta u + f(\mathbf{x}, t, u) && t > 0, \mathbf{x} \in \Omega \\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}) && \mathbf{x} \in \Omega \\ \nabla u(\mathbf{x}, t) \cdot n(\mathbf{x}) &= 0 && t > 0, \mathbf{x} \in \partial\Omega \end{aligned}$$

i.e. with *von Neumann boundary condition*.

In general there are three commonly used boundary conditions

- $\nabla u(\mathbf{x}, t) \cdot n(\mathbf{x}) = \phi(\mathbf{x}) \qquad t > 0, \mathbf{x} \in \partial\Omega \qquad$ von Neumann
- $\nabla u(\mathbf{x}, t) \cdot n(\mathbf{x}) + a(\mathbf{x}) u(\mathbf{x}, t) = \phi(\mathbf{x}) \qquad t > 0, \mathbf{x} \in \partial\Omega \qquad$ Robin
- $u(\mathbf{x}, t) = \phi(\mathbf{x}) \qquad t > 0, \mathbf{x} \in \partial\Omega \qquad$ Dirichlet

with $a(\mathbf{x}) \geq 0$. A further boundary condition is the *periodic*.

There are two phenomena that one often observes solving reaction-diffusion equations

- **Wave propagation**: On an unbounded habitat the population will move from an occupied area to an unoccupied area with a constant velocity.
- **Critical patch size**: On a bounded area with $u = 0$, the persistence of a population depends on the size of the habitat.

We now want to investigate the case with a no-flux boundary condition

$$
\begin{aligned}
&\frac{\partial}{\partial t} u = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \lambda u \qquad t > 0, (x, y) \in R = (0, a) \times (0, b) \\
&\nabla u \cdot n = 0 \qquad\qquad (x, y) \in \partial R \\
&u(x, y, 0) = u_0(x, y) \qquad\qquad (x, y) \in R \\
&D, \lambda, a, b > 0
\end{aligned}
$$

Reaction-Diffusion Equation VII

We assume that we can separate variables

$$u(x, y, t) = U(t)V(x, y) \; . \tag{16}$$

Then

$$U'(t) = (Dk + \lambda)U(t) \tag{17}$$

and

$$\begin{aligned} \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} &= kV \quad t > 0, (x, y) \in R & (18) \\ \nabla V \cdot \mathbf{n} &= 0 \qquad (x, y) \in \partial R \; . & (19) \end{aligned}$$

Assume further

$$V(x, y) = W(x)Z(y) \tag{20}$$

Reaction-Diffusion Equation VIII

then

$$\frac{W''(x)}{W(x)} + \frac{Z''(x)}{Z(x)} = k \tag{21}$$

and hence

$$\frac{W''(x)}{W(x)} = \text{const} \quad \text{and} \frac{Z''(x)}{Z(x)} = \text{const} \, . \tag{22}$$

The von Neumann boundary condition implies

$$W'(0) = W'(a) = 0 \tag{23}$$
$$Z'(0) = Z'(b) = 0 \tag{24}$$

and thus

Reaction-Diffusion Equation IX

$$W''(x) = k_1 W(x), \quad x \in (0,a), W'(0) = W'(a) = 0 \tag{25}$$
$$Z''(y) = k_2 Z(y), \quad y \in (0,b), Z'(0) = Z'(b) = 0 \tag{26}$$
$$k = k_1 + k_2 \ . \tag{27}$$

The problems stated in eq are eigenvalue problems in one dimension. The eigenvalues and eigenfunctions are

$$k_{1n} = -\frac{n^2\pi^2}{a^2}, \quad W_n(x) = \cos\left(\frac{n\pi x}{a}\right), \quad n \in \mathbb{N} \tag{28}$$
$$k_{2n} = -\frac{m^2\pi^2}{b^2}, \quad Z_m(y) = \cos\left(\frac{m\pi y}{b}\right), \quad m \in \mathbb{N} \tag{29}$$

from which we get

Reaction-Diffusion Equation X

$$k_{n,m} = -\frac{n^2\pi^2}{a^2} - \frac{m^2\pi^2}{b^2} \tag{30}$$

$$V_{n,m} = \cos\left(\frac{n\pi x}{a}\right)\cos\left(\frac{m\pi y}{b}\right), \quad n, m \in \mathbb{N}\,. \tag{31}$$

Put it all together we have

$$u(x, y, t) = \sum_{n=0}^{m=0} c_{n,m} e^{(Dk_{n,m}+\lambda)t} \cos\left(\frac{n\pi x}{a}\right)\cos\left(\frac{m\pi y}{b}\right) \tag{32}$$

and the $c_{n,m}$ are determined by the initial condition.

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Reaction

Reaction I

Some general literature on population dynamics

- Modeling Differential Equations in Biology [2]
- Mathematical Biology, Vol. 1: An Introduction [3]
- Mathematical Biology, Vol. 2: Spatial Models and Biomedical Applications [4]
- Essential Mathematical Biology [5]
- Mathematics in Population Biology [6]
- Mathematical Models in Biology [7]

Let P denote a population and we shall assume spatial homogeniety. The evolution of the popululation can be described by a general differential equation assume some function g

$$\frac{dP}{dt} = Pg(p) \tag{33}$$

Eq 33 can also be interpreted in terms of a discrete equation (first considered by Robert May)

$$P_{n+1} = P_n g(P_n)\ . \tag{34}$$

If $g(P_n) = k(1 - P_n/N)$ then the equation yields the *logistic map* where k is the control parameter

$$P_{n+1} = kP_n(1 - P_n/N)\ . \tag{35}$$

Dividing both sides by N and matching the substitution $x_n = P_n/N$ we arrive at

$$x_{n+1} = kx_n(1 - x_n) \tag{36}$$

with $x_n \in [0, 1]$ and $k > 0$ which is a *non-linear difference equation*. We consider this as a function of the control parameter k

$$x_{n+1} = kx_n(1 - x_n) = f(k, x_n)\ . \tag{37}$$

This equation is also known as the **logistic growth equation**.

Iterative Models III

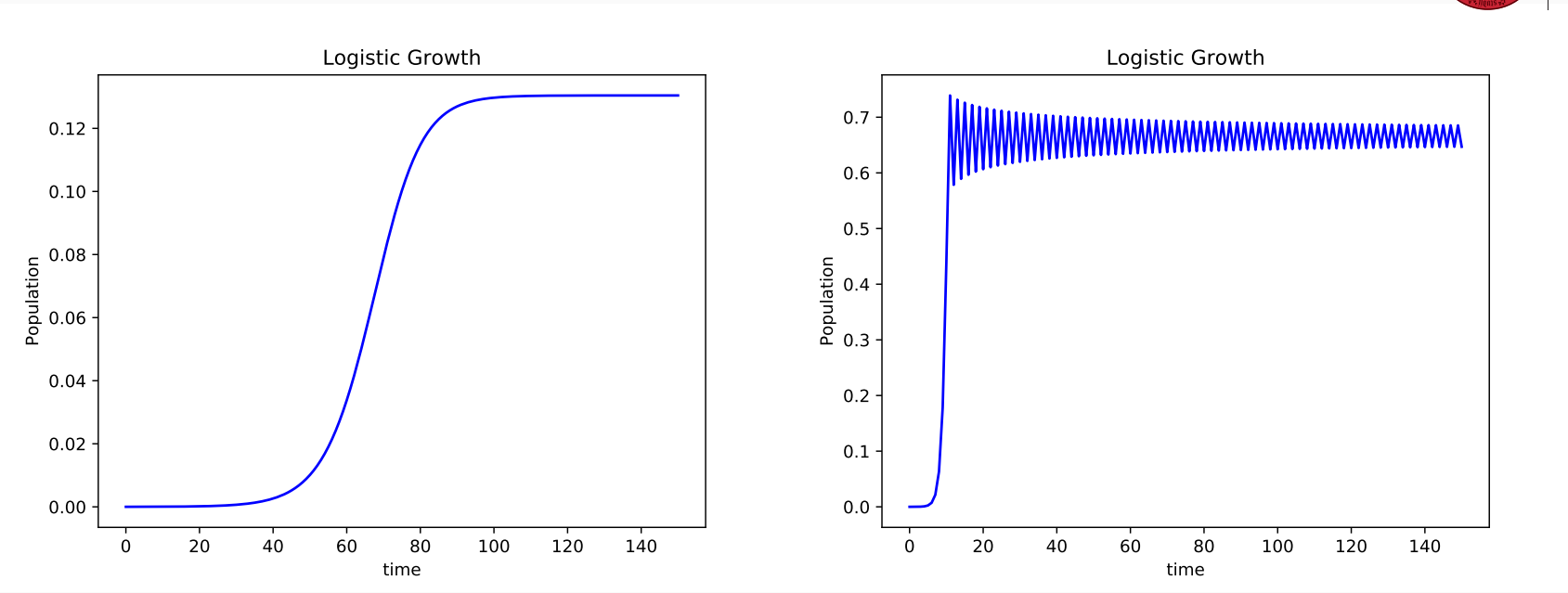

Figure 3: Result of the discrete form of the logistic map (37). The parameters for the left panel were: $x_0 = 0.00001$, k=1.15 and $n = 150$ and for the right panel k was chosen to be equal to 3.

Iterative Models IV

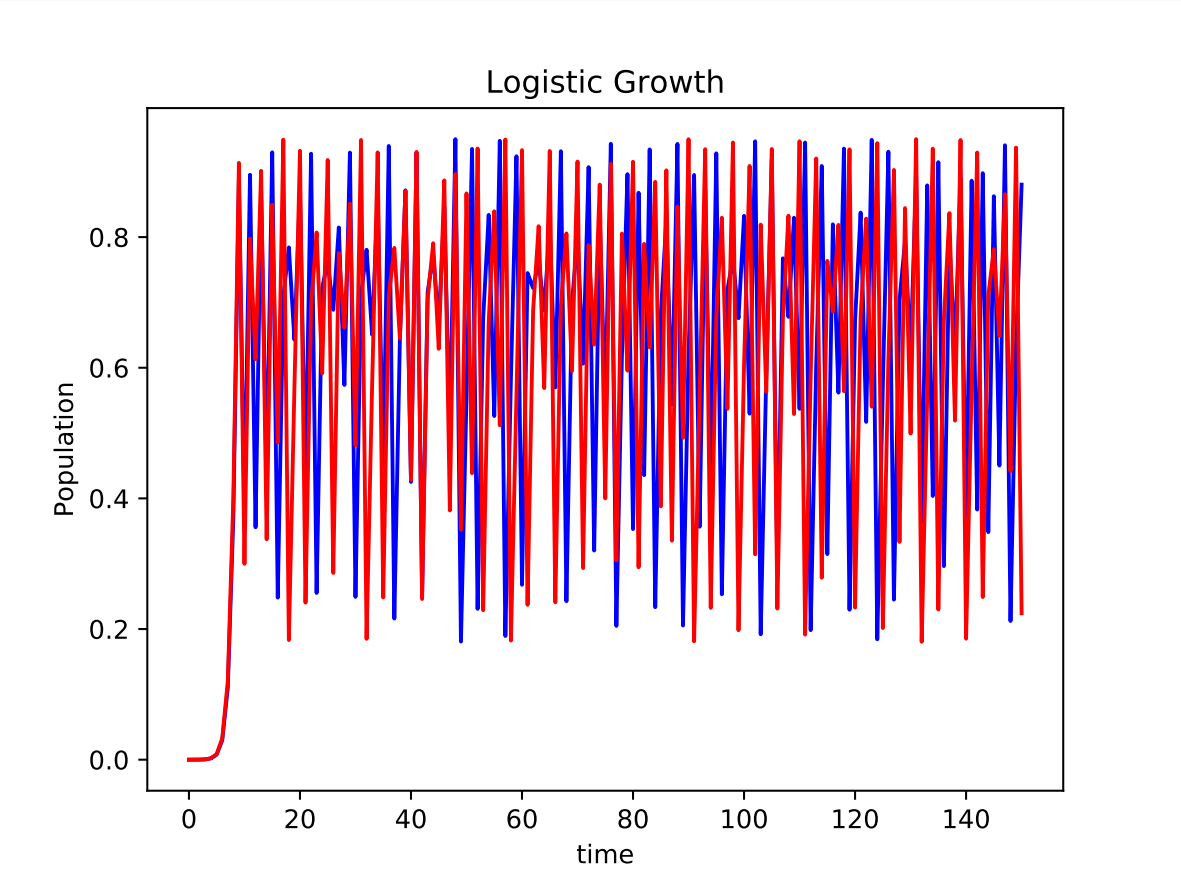

Figure 4: Result of the discrete form of the logistic map (37). The parameters for the left panel were: $x_0 = 0.00001$ (blue) and $x_1 = 0.000011$ (red), k=3.8 and $n = 150$.

Logistic Map

```
1
  import numpy as np
3
  def logisticMap(n,k,x0):
5   x = np.zeros((n+1,2))
    x[0,0] = 0
7   x[0,1] = x0
    for i in range(n):
9     x[i+1,0] = x[i,0] + 1
      x[i+1,1] = k*x[i,1]*(1-x[i,1])
11
    return x
13
  n  = 150
15 k  = 1.15
  x0 = 0.00001
17
  x = logisticMap(n,k,x0)
```

Code 1: Logistic growth

The question immediately arising is whether there exists a *fixed point*

$$x^* = f(k, x^*) \; . \tag{38}$$

Assume that we have a small pertubation of x^*

$$\eta = x_n - x^* \tag{39}$$

then

$$\begin{aligned} x^* + \eta_{n+1} &= f(k, x^* + \eta_n) && (40) \\ &= f(k, x^*) + f'(k, x^*)\eta_n + O(\eta_n^2) \; . && (41) \end{aligned}$$

Linearization near x^* leads to

$$\eta_{n+1} = f'(k, x^*)\eta_n \; . \tag{42}$$

Define $\lambda = f'(k, x^*)$. If

$$|\lambda| = |f'(k, x^*)| < 1 \tag{43}$$

then $(\eta_n) \to 0$ as $n \to \infty$. The sequence (x_n) converges to x^* and x^* is *linearly stable*.

Consider the Diagram 6. The derivative of $f(k, x)$ at the intersection point (being equal to $x^* = (k-1)/k$ with $y = x$ is

$$f'(k, x^*) = 2 - k \; . \tag{44}$$

To ensure that $|f'| < 1$ we need $k > 1$. On the other hand we should not leave the interval $[0, 1]$ so that

$$1 < k \leq 4 \tag{45}$$

Clearly if $1 < k < 3$ then $|f'(k, x^*)| < 1$ and every initial value leads the stable equilibrium value

$$x^* = \frac{k-1}{k} \; . \tag{46}$$

If we choose $k = k_0 = 3$ then we have a marginally stable equilibrium point. Choose

$$x_1 = x^* + \delta \ . \tag{47}$$

Linearization of $f(k, x_1)$ yields

$$x_2 = x^* - \delta \ . \tag{48}$$

and vice versa. Hence we get an oscillation and the period has doubled. The point

$$k_0 = 3 \qquad \text{and} \qquad x_0^* = x^*(k_0) \tag{49}$$

is a *branching point*. If k is slightly above k_0 then $x^* = (k-1)/k$ becomes unstable and a period doubling sets in, i.e., the fixed point does not satisfy $x^* = f(k, x^*)$ but

$$x^* = f(k, f(k, x^*)) \tag{50}$$

hence we have a *bifurcation*. So now we need to study the map $f \circ f$

$$x_{n+1} = k^2 x_n (1 - x_n)[1 - k x_n (1 - x_n)] \tag{51}$$

and look for the fix points. and have two of them in the range

$$3 \leq k \leq 1 + \sqrt{6} \approx 3.449 \; . \tag{52}$$

Then we obtain two mariginaly stable fixpoints and corresponding branching with a period doubling. This *period doubling cascade* continues until we reach a limit point

$$k_\infty = 3.56994... \tag{53}$$

Figure 5 shows the bifurcation diagram for the logistic equation. If we consider

$$\lim_{n \to \infty} = \frac{k_i - k_{i-1}}{k_{i+1} - k_i} = \delta \tag{54}$$

then this value is universal under certain conditions on the smoothness of the maps with a value of

$$\delta = 4.669201609 \tag{55}$$

the **Feigenbaum constant**. Beyond approximately 3.57 we find the onset of chaos.

Figure 5 shows the bifurcation diagram for the logistic equation.

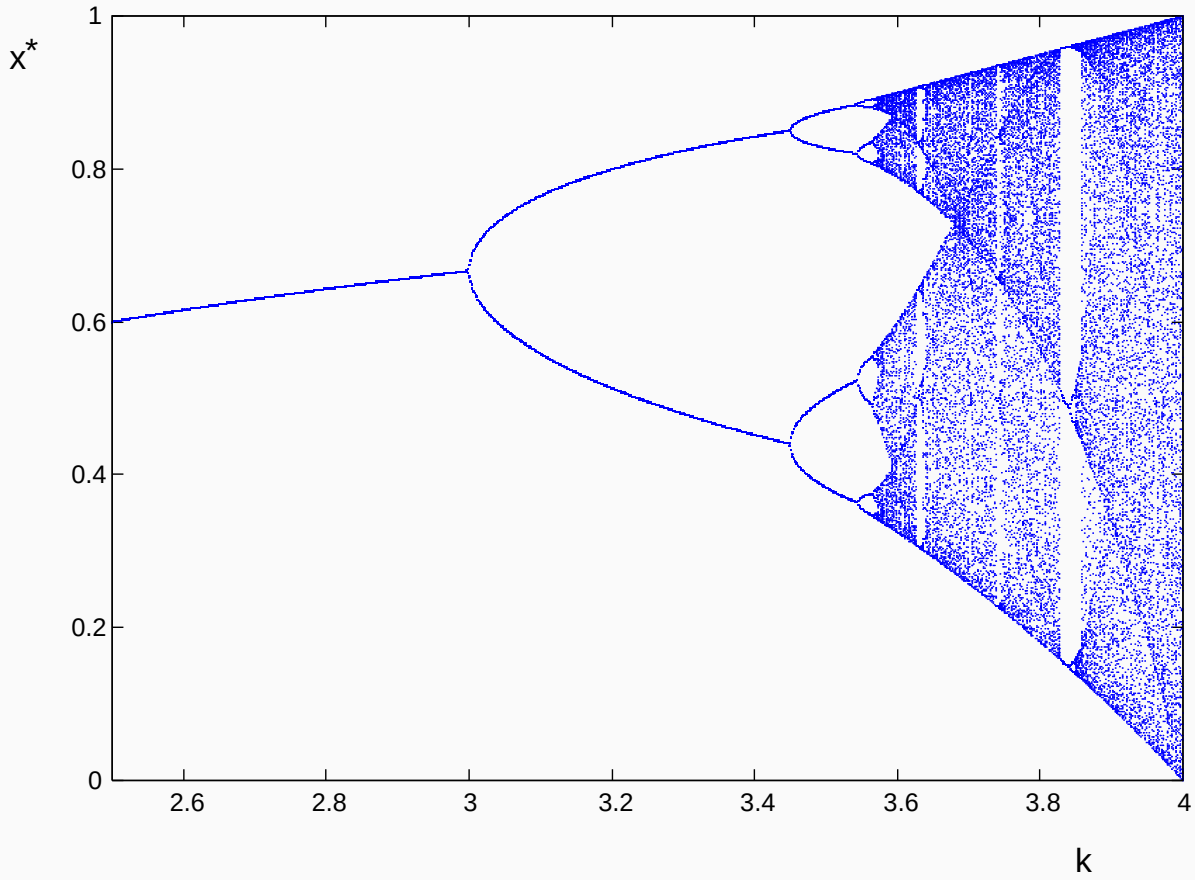

Figure 5: Bifurcation diagram for the logistic map. The figure was generated using the octave program from the wikipedia shown in the text.

Iterative Models VII

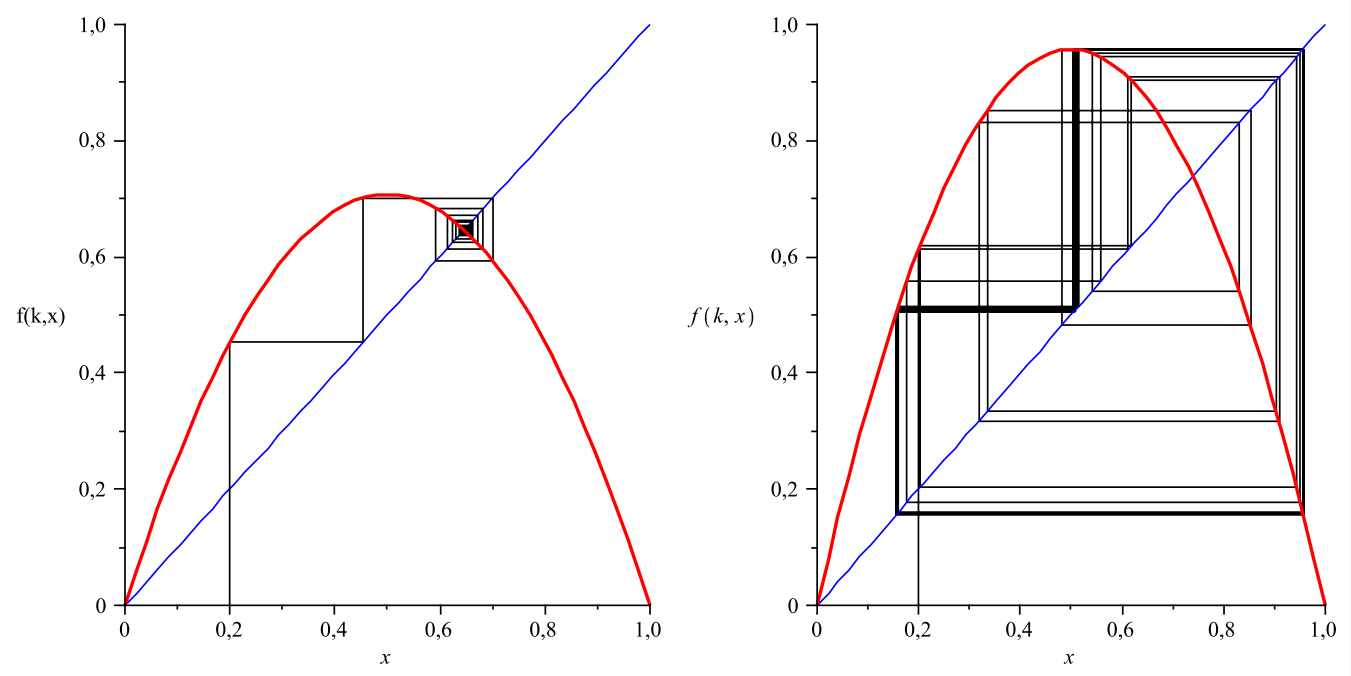

Figure 6: Cobweb for the logistic map for the case $k = 2.83, 3.83$ (left, right image) and an initial value of $x(0) = 0.2$.

Iterative Models VIII

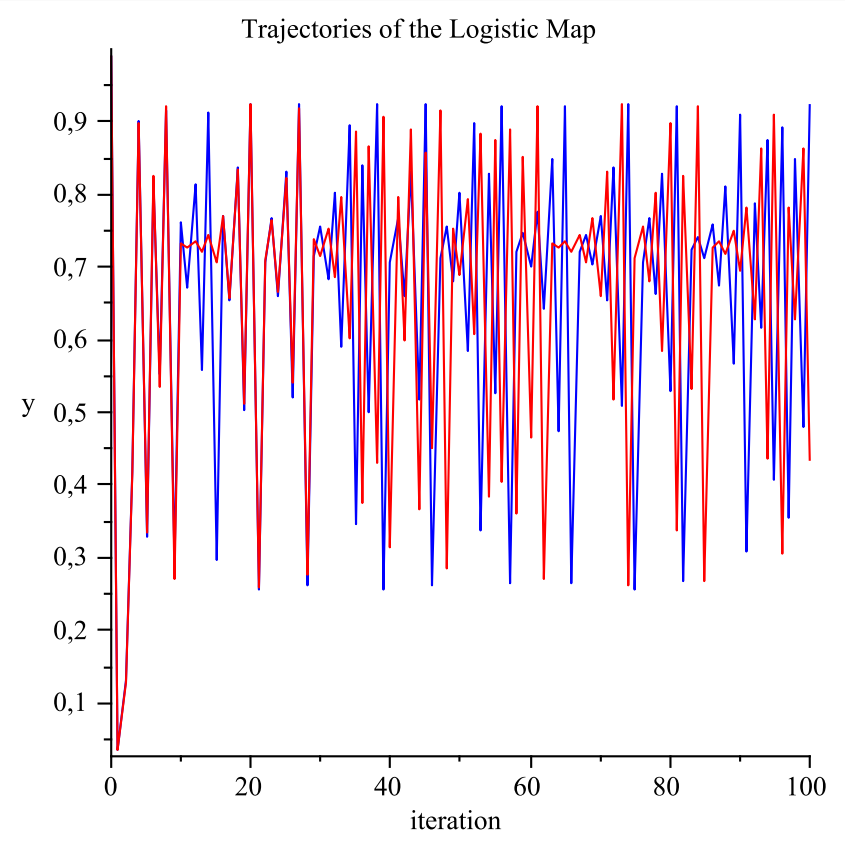

Figure 7: Two trajectories for the logistic map. The trajectories where obtained using $k = 3.7$ and two initial values that are very close together: 0.99 and 0.9901. The two trajectories, although having close initial value start to deviate quickly.

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Finite State Cellular Automata

Cellular Automata have many applications beside the reaction-diffusion systems which we will study here, e.g. fluid dynamics, growth, reproduction, competition and evolution etc.

Consider first a one dimensional lattice Λ. In general we will be looking at lattice like $\mathbb{Z}^d$, $(\mathbb{Z} \mod L)^d$, etc.

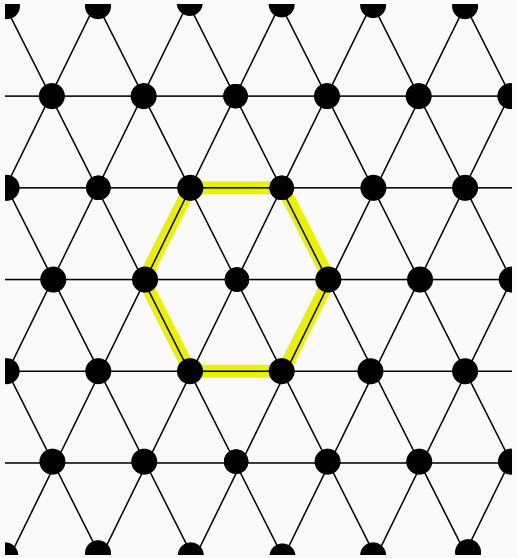

Figure 8: Hexagonal lattice

Each site i ($i = 1, ..., N$) can be in k states. At each time step t every cell changes state (synchronous updating) depending on its present state and on the states of its neighbours

$$s_i(t+1) = f(s_{i-r}, s_{i-r-1}, ..., s_i, s_{i+1}, s_{i+r}) \tag{56}$$

where r is called the radius of the neighbourhood. Thus we have

$$p = k^{2r+1} \tag{57}$$

possible permutations and k^p possible rules to generate the next step.

As for the models and methods we have discussed in the first lectures we have to specifiy the boundary conditions, e.g.

- free
- periodic

Thus cellular automata are specified by

- space

- states
- neighbourhood
- rules

Let us look at an example. We fix the number of states to two $\{0, 1\}$ and choose the rule to only look at the nearest neighbours and ignore the own state (see Figure 9)

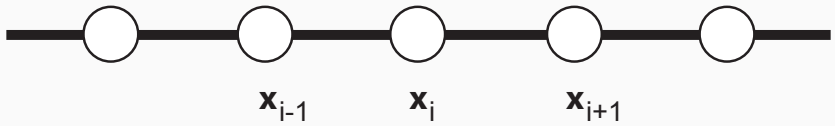

Figure 9: Two states: 0,1. Rule: only look at the nearest neighbours and ignore the own state

Table 1: Exclusive OR (XOR)

$i-1$	$i+1$	i
0	0	0
0	1	1
1	0	1
1	1	0

hence we only look at the nearest neighbours and ignore the state of the automaton that is going to be updated. Figure 10 shows the result for this cellular automaton with the initial condition setting all cells to 0 except the center cell which is set to 1.

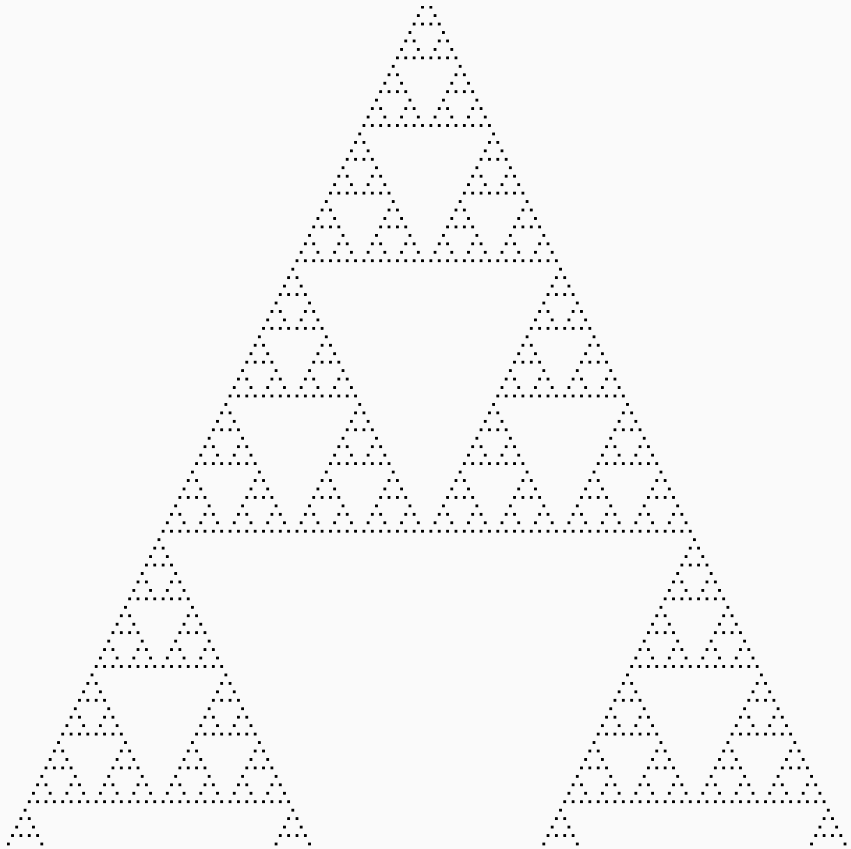

Figure 10: Visualization of the time evolution (y-axis) of the exclusive or (XOR) cellular automaton. In particular, the figure shows the result for the XOR CA for 100 time steps.

Another example is the

$$\text{Rule 30} \quad (00011110_2 = 30)\,. \tag{58}$$

Figure 11: Image taken from Wikipedia: Richard Ling <wikipedia@rling.com> - Own work; Location: Cod Hole, Great Barrier Reef, Australia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=293495

Finite State Cellular Automaton Models VII

Assume m out of $2r+1$ distinct neighbourhoods map to non-zero states. To classify the outcome of the time evolution (see Exercise 6), ie. the long term behaviour of the dynamical system we define

$$\lambda := \frac{m}{k^{2r+1}} \ . \tag{59}$$

We want to classify the automaton into trivial (uniform), chaotic, stable periodic, aperiodic, localized complex, ... etc.

Class 1 small λ

Class 2 $\lambda \approx 0.5$.

Recall that chaotic is meant to imply

- sensitivity to the initial conditions,
- topological mixing and
- dense periodic orbits.

We will denote the transition rule by G for general lattices or graphs

$$G = [g(x)] \quad x \in \Lambda \tag{60}$$

and the overall state

$$S = [s(x,t)] \; . \tag{61}$$

Note that one can also define time-dependent rules. One example is to alternate between two rules.

Cellular automata with fixed rules are called **deterministic automata**. Cellular automata are called **probabilistic** if from a given set of rules, each rule is applied with a probability.

Thus for example for $d = 2$ we have

$$\begin{array}{rcl} G & : & s(t) \to s(t+1) \\ g_{xy} = g_{ij} & : & s_{ij}(t) \to s_{ij}(t+1) \; . \end{array} \tag{62}$$

We further define two typical neigbourhoods: the **von Neumann and the Moore neighbourhood**, i.e. the nearest and the next-nearest neighbours of a given cell (see Figure 12)

$$|x - x_0| + |y - y_0| \leq r \quad \text{von Neumann} \tag{63}$$

$$|x - x_0| \leq r \text{ and } |y - y_0| \leq r \quad \text{Moore .} \tag{64}$$

We denote the neighbourhood by $\mathcal{N}(r)$.

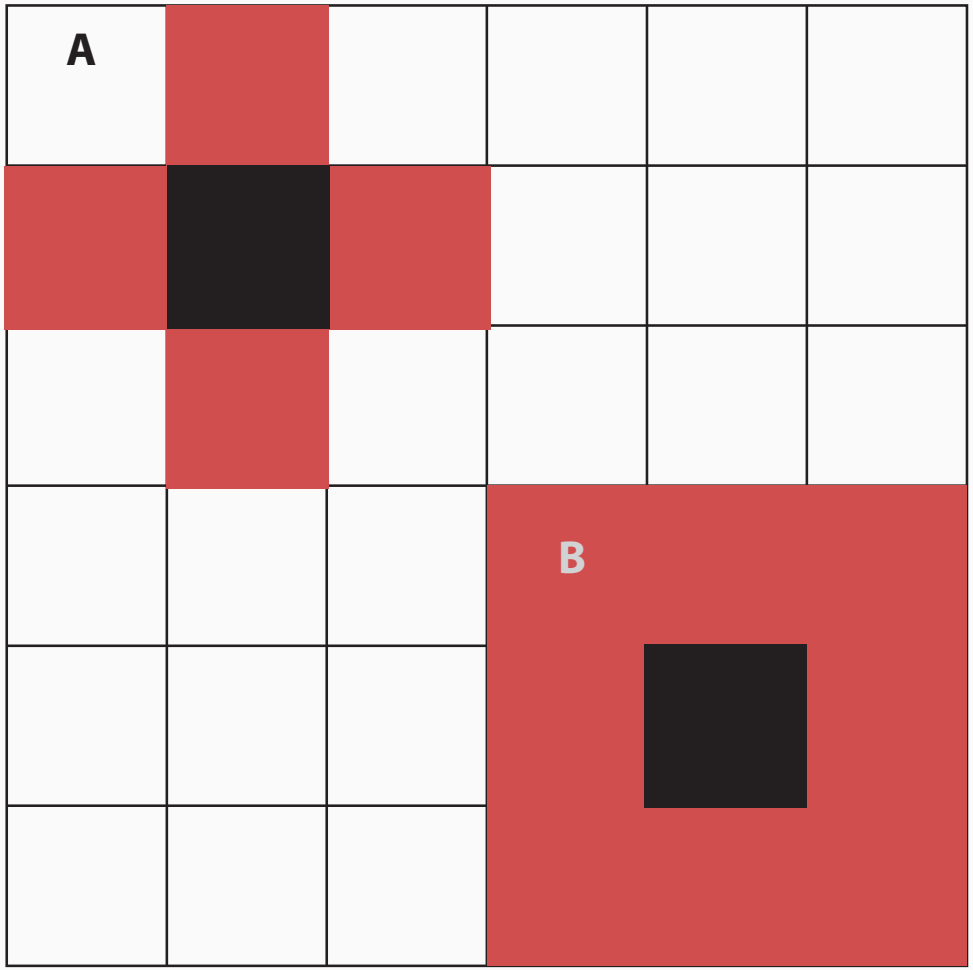

Figure 12: von Neumann (A) and the Moore (B) neighbourhood.

Recall that we defined the transition

$$s_i(t+1) = f(s_{i-r}, s_{i-r-1}, ..., s_i, s_{i+1}, s_{i+r})$$

where f is the rule for the transition. Suppose we change the rule according to to some probability from f to some other function g. Depending on the number of states a single automaton can have and depending on the neighbourhood we have a number of possible rules. Let R be the set of possible rules and $R_{\mathcal{N}(\nabla)}$ a finite the subset of R ($|R_{\mathcal{N}(\nabla)}| = n$).

We define a **probabilistic cellular automata** (PCA) (stochastic cellular automata) as a discrete-time dynamical system with synchronous update of the states where the updating rule is chosen according to a probability distribution.

$$G(s_{\mathcal{N}(r)}) := \begin{cases} z^1 & \text{with probability} W(s_{\mathcal{N}(r)} \to z^1) \\ z^2 & \text{with probability} W(s_{\mathcal{N}(r)} \to z^2) \\ \dots & \\ z^n & \text{with probability} W(s_{\mathcal{N}(r)} \to z^n) \end{cases} \tag{65}$$

Here $z^i \in$

The basic algorithm for the time development of the cellular automaton is

Algorithm 1 Basic Algorithm: Cellular Automaton

```
1: initialize every cell of the cellular automaton
2: for n_cycles do
3:     store the state of every cell
4:     for every cell s_i do
5:         apply rule to s_i
6:     end for
7: end for
```

This algorithm can be implemented in an object oriented approach by defining a class for the cell and for the automaton. The cell class implements all the book keeping of the state.

```
class Cell:
    def __init__(self):
        self.prev_state = 0
        self.state = 0

    def get_state(self):
        return self.state

    def set_state(self, state):
        self.state = state

    def get_prev_state(self):
        return self.prev_state

    def set_prev_state(self, state):
        self.prev_state = state

    def copy_state(self):
        self.prev_state = self.state
```

Code 2: Cellular automaton: Basic cell class

The automaton class implements the topology (here a simple $1 - D$ lattice), the initialization and the generation of the next step with the implementation of the rule that govern the update of the cell.

Finite State Cellular Automaton Models

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

```
class Automata:
    def __init__(self, numcols):
        self.cols = numcols
        self.cells = []
        for i in range(numcols):
            cell = Cell()
            self.cells.append(cell)
    def get_all_cells(self):
        return self.cells
    def next_generation(self):
        for i in range(self.cols):
            self.cells[i].copy_state()
        for i in range(self.cols):
            curr_cell = self.cells[i]
            Automata.apply_rule(self,i)
    def apply_rule(self,i):
       state = self.cells[i].get_prev_state()
       left = i-1
       if (left < 0):
          left = self.cols-1
       right = i+1
       if (right == self.cols):
          right = 0
       if (self.cells[left].get_prev_state() <> self.cells[right].
     get_prev_state()):
            self.cells[i].set_state(1)
       else:
          self.cells[i].set_state(0)
    def init_automata(self):
       center = int(self.cols / 2)
```

One subclass of cellular automata consists of the rules where the new state depends only on the sum of states of the neighbours, this class of *totalistic automata*:

$$s_{ij}(t) = G\left(\sum_{\alpha=-r}^{r}\sum_{\beta=-r}^{r} a_{\alpha\beta}s_{i+\alpha,j+\beta}(t)\right) \qquad i,j = 1,...,L \tag{66}$$

where $a_{\alpha,\beta}$ are some coefficients.

Time Dependent Diffusion Equation in d=1 (Heat Equation) I

Consider the equation

$$\frac{\partial P}{\partial t} = k \frac{\partial^2}{\partial x^2} P \tag{67}$$

with the discretization of the form

$$P_i^{n+1} - P_i^n = k \frac{\Delta t}{(\Delta x)^2} (P_{i+1}^n - 2P_i^n + P_{i-1}^n) \, . \tag{68}$$

The evolution of this cellular automaton under the constraint $k\Delta t/(\Delta)^2 = 0.01$ is shown in Figure 13 (thus not finite state).

Time Dependent Diffusion Equation in d=1 (Heat Equation) II

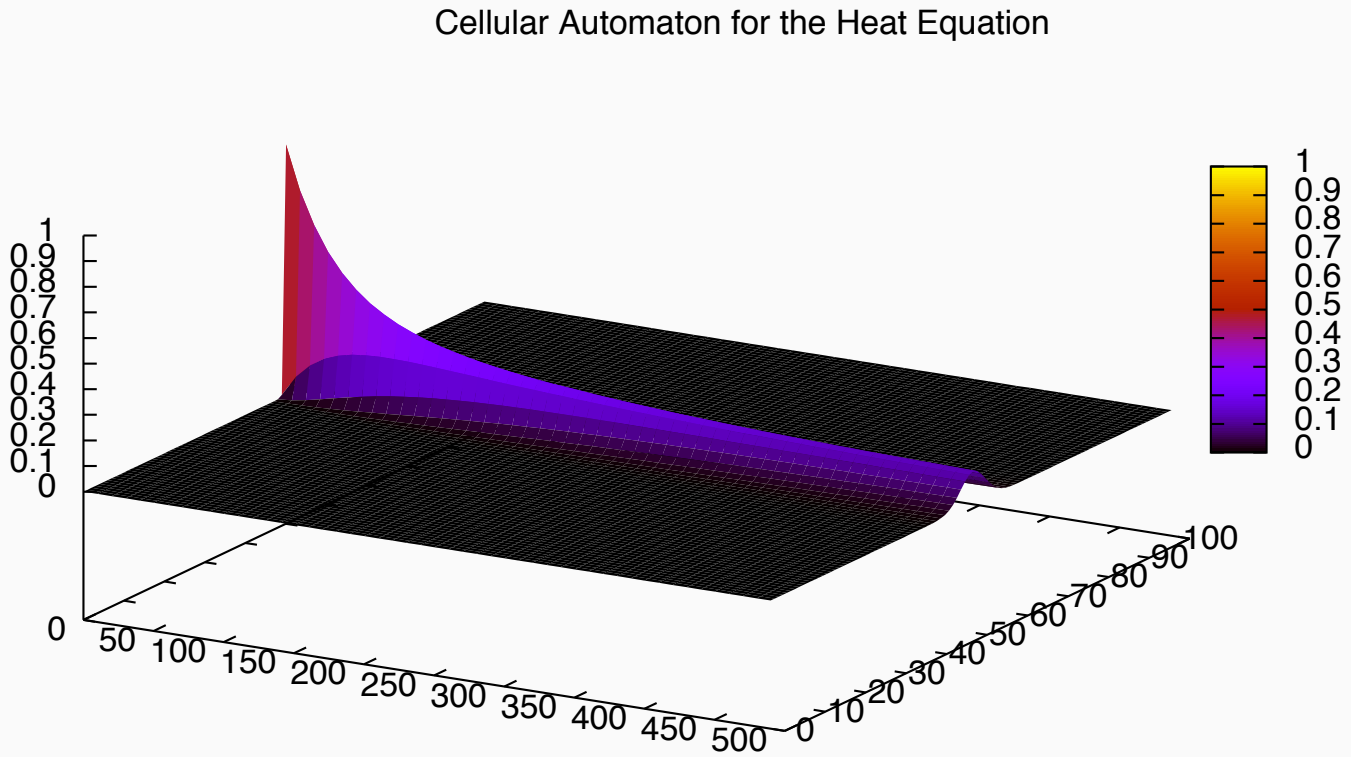

Figure 13: Evolution of the d=1 heat equation cellular automaton with the parameter $k\Delta t/(\Delta)^2 = 0.01$ and initial condition 1 at the center of the one-dimensional lattice.

Consider the equation

$$\frac{\partial P}{\partial t} = D(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})P + f(P) \tag{69}$$

where we think of P as a state variable. D is a constant. A possible discretization and conversion into a cellular automaton is

$$\frac{P_{ij}^{n+1} - P_{ij}^{n}}{\Delta t} = D\left(\frac{P_{i+1,j}^{n} - 2P_{ij}^{n} + P_{i-1,j}^{n}}{(\Delta x)^2} + \frac{P_{i,j+1}^{n} - 2P_{ij}^{n} + P_{i,j-1}^{n}}{(\Delta y)^2}\right) + f(P_{ij}^{n}) \tag{70}$$

Choose $\Delta t = \Delta x = \Delta y = 1$, then

$$P_{ij}^{n+1} = D\left(P_{i+1,j}^{n} + P_{i-1,j}^{n} - P_{i,j+1}^{n} + P_{i,j-1}^{n}\right) + (1 - 4D)P_{ij}^{n} + f(P_{ij}^{n}) \tag{71}$$

This can be rewritten as

$$P_{ij}^{n+1} = \sum_{k,l=-r}^{r} a_{kl} P_{i+k,j+l}^{n} + f(P_{ij}^{n}) \tag{72}$$

In the spirit of the one-dimensional heat equation, the **Fisher diffusion logistic equation** reads

$$\frac{\partial P}{\partial t} = k\frac{\partial^2}{\partial x^2}P + \alpha P(1-P) \mod M \tag{73}$$

An example of a solution for the $s = \{0, ..., M\}$ state model with the parameters (see notation of example) $k\Delta t/(\Delta)^2 = 1$ and $\alpha = 100$ is shown in Figure 14.

Cellular Automata Examples: Fisher Diffusion Logistic Equation II

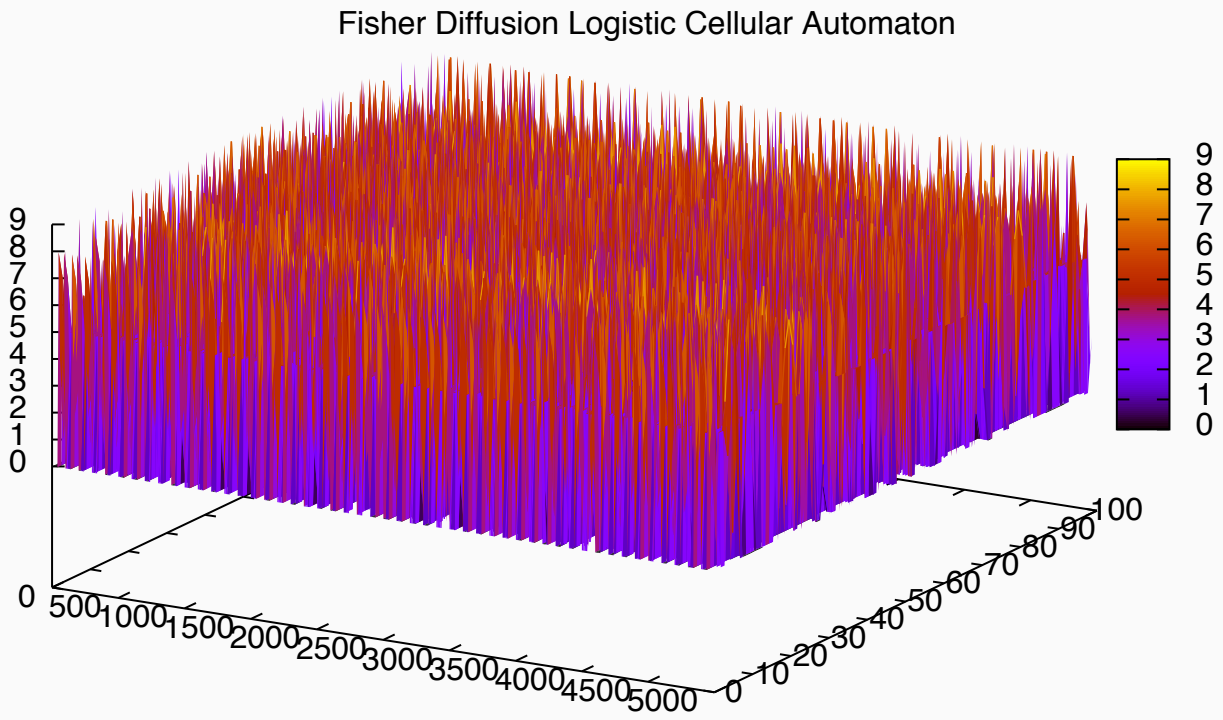

Figure 14: Evolution of the d=1 Fisher diffusion logistic equation cellular automaton with the parameter $k\Delta t/(\Delta)^2 = 0.01$ and initial condition 1 at the center of the one-dimensional lattice.

The **Noisy Burgers cellular automaton** is defined as

$$\frac{\partial P}{\partial t} = 2P\frac{\partial P}{\partial x} + \frac{\partial^2}{\partial x^2}P + \nabla\eta \tag{74}$$

where η is a noise with

$$\begin{aligned} <\eta(x,t)> &= 0 \\ <\eta(x,t)\eta(x',t')> &= 2D\delta(x-x')\delta(t-t')\ . \end{aligned} \tag{75}$$

Cellular Automata Examples: Game of Life I

Assume a two-dimensional lattice with periodic boundary conditions. The Game of Life is composed of cellular automatons each of which is either 'on/alive' or 'off/dead'. The state of each automaton at time t is determined by its own state and the states of its eight immediate neighbours at $t-1$ according to the following simple rules:

- Any 'on' cell (at time t-1) with fewer than two 'on' neighbours (at t -1) transitions to an 'off' state at time t.
- Any 'on' cell (t -1) with two or three 'on' neighbours (t -1) remains 'on' at time t.
- Any 'on' cell (t -1) with more than three 'on' neighbours (t -1) transitions to an 'off' state at time t
- And 'off' cell (t -1) with exactly three 'on' neighbours (t -1) transitions to an 'on' state at time t.

Cellular Automata Examples: Game of Life

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

```
class Game(object):

    def __init__(self, state, infinite_board = True):

        self.state = state
        self.width = state.width
        self.height = state.height
        self.infinite_board = infinite_board

    def step(self, count = 1):

        for generation in range(count):

            new_board = [[False] * self.width for row in range(self.height)]

            for y, row in enumerate(self.state.board):
                for x, cell in enumerate(row):
                    neighbours = self.neighbours(x, y)
                    previous_state = self.state.board[y][x]
                    should_live = neighbours == 3 or (neighbours == 2 and
     previous_state == True)
                    new_board[y][x] = should_live

            self.state.board = new_board
```

Cellular Automata

```
    def neighbours(self, x, y):
2
        count = 0
4
        for hor in [-1, 0, 1]:
6           for ver in [-1, 0, 1]:
                if not hor == ver == 0 and (self.infinite_board == True or (0
 <= x + hor < self.width and 0 <= y + ver < self.height)):
8                   count += self.state.board[(y + ver) % self.height][(x + hor
 ) % self.width]

10      return count

12  def display(self):
        return self.state.display()
```

Cellular Automata

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

```
class State(object):

    def __init__(self, positions, x, y, width, height):
        active_cells = []
        for y, row in enumerate(positions.splitlines()):
            for x, cell in enumerate(row.strip()):
                if cell == 'o':
                    active_cells.append((x,y))

        board = [[False] * width for row in range(height)]

        for cell in active_cells:
            board[cell[1] + y][cell[0] + x] = True

        self.board = board
        self.width = width
        self.height = height

    def display(self):

        output = ''

        for y, row in enumerate(self.board):
            for x, cell in enumerate(row):
                if self.board[y][x]:
                    output += ' o'
                else:
                    output += ' .'
            output += '\n'
```

Cellular Automata

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

```
glider = """ oo.
             o.o
             o.. """

my_game = Game(State(glider, x = 2, y = 3, width = 10, height = 10))
print my_game.display()
my_game.step(500)
print my_game.display()
```

Excercises

Excercises I

Exercise 1: **Generalized Logistic Map**

The logistic map (Equation 37) can be generalized as follows

$$x_{n+1} = kx_n^2(1 - x_n) \ . \tag{76}$$

Here we have assumed that the growth rate in the low-density limit is proportional to x^2. What are the r values that yield non-zero population?

Exercise 2: **Henon Map**[8]

Henon proposed the following map.

$$(x, y) \to (1 + y - ax^2, bx) \ . \tag{77}$$

Start with the following parameter values: $a = 1.4, b = 0.3$. Plot the sequence (x_n, x_{n+1}) for different parameters.

Exercise 3: **Chaotic maps**

Study the following chaotic maps. Plot the sequence (x_n, x_{n+1}) for different parameters.

$$x_{n+1} = x_n e^{1-x_n} \ , \tag{78}$$

$$x_{n+1} = r\frac{x_n}{(1 + x_n)^8} \ . \tag{79}$$

Excercises II

Exercise 4: **Quadratic form**

Consider the equation

$$z_{n+1} = z_n^2 + c \quad z, c_n \in \mathbb{C} \, . \tag{80}$$

Analyze this map in terms of the **Mandelbrot Set**. For this consider the set of complex values for which the trajectory of 0 remains bounded. Consider specifically the boundary.

Exercise 5: **Julia Set**

Consider the equation

$$z_{n+1} = z_n^2 + c \quad z, c_n \in \mathbb{C} \, . \tag{81}$$

at fixed c. The Julia set is composed of the starting values z_0 for which the trajectory remains bounded, i.e. $|z_n| < \kappa(c)$ for any given n.

Exercise 6: **Wolfram Classification**

Wolfram [9, 10] classified for $d = 1$ the cellular automata into the following classes (see Figure 15 for two examples):

- From almost all initial states the automaton transitions into a homogeneous state.

Excercises III

- From almost all initial states the automaton transitions into periodic structures.
- From almost all initial states the automaton transitions into aperiodic structures.
- Complex spatial structures occur

Search for at least one example for the above cases.

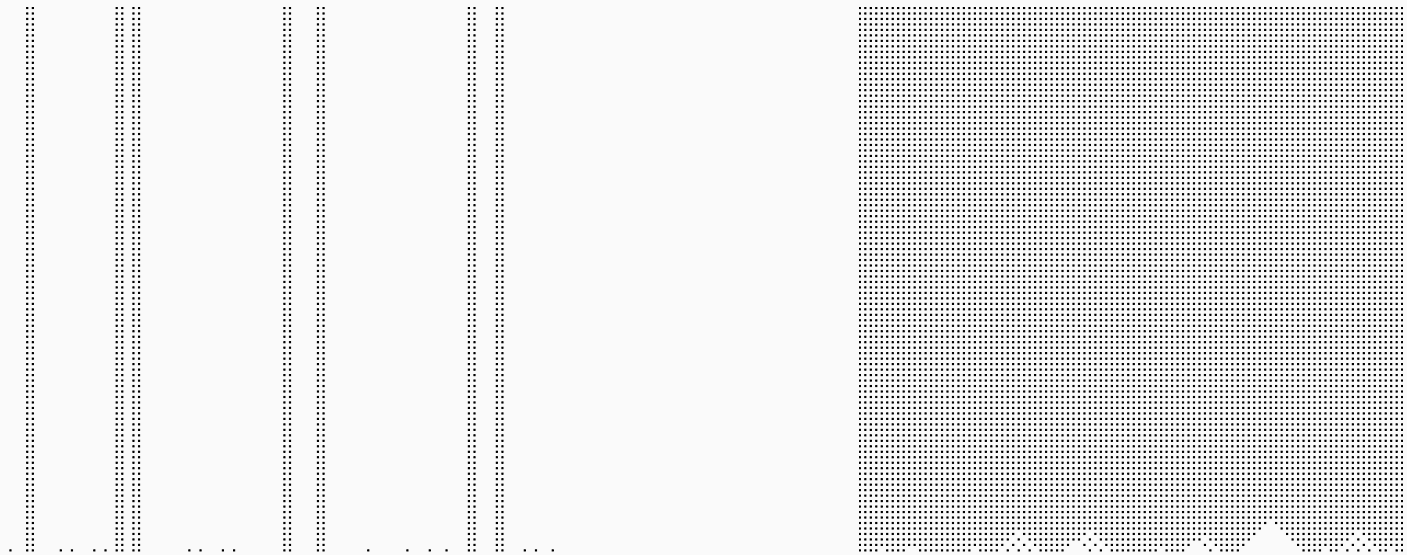

Figure 15: Two examples of the evolution of the d=1 cellular automaton: **A** with the rule $x(i)$ AND $(x(i-1)$ XOR $x(i+1))$. The initial random state turns into a fixed structure already after one step. **B** with the rule $(x(i-1)$ OR $x(i+1))$ from a random state.

Excercises IV

Bibliography

References

[1] G. Meacci and K. Kruse. *Phys. Biol.*, 2:89–97, 2005.

[2] Clifford Henry Taubes. *Modeling Differential Equations in Biology*. Prentice Hall, 2000.

[3] James Dickson Murray. *Mathematical Biology, Vol. 1: An Introduction*, volume 1. Springer-Verlag, New York, 2002.

[4] James Dickson Murray. *Mathematical Biology, Vol. 2: Spatial Models and Biomedical Applications*, volume 2. Springer-Verlag, New York, 2002.

[5] Nicholas F. Britton. *Essential Mathematical Biology*. Springer-Verlag, London, 2003.

[6] Horst R. Thieme. *Mathematics in Population Biology*. Princeton University Press, 2003.

[7] Leah Edelstein-Keshet. *Mathematical Models in Biology*. McGraw-Hill, Boston, 2005.

[8] M. Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50(1):69–77, 1976.

Bibliography II

[9] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics,*, 55(3):601–644.

[10] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Index

Index I

Index II

Index III