Japanese Journal of Statistics and Data Science
https://doi.org/10.1007/s42081-024-00283-w

ORIGINAL PAPER

Parameter estimation for cellular automata

Alexey Kazarnikov[1] · Nadja Ray[2] · Heikki Haario[3] · Joona Lappalainen[3] · Andreas Rupp[4]

Received: 1 August 2024 / Revised: 30 November 2024 / Accepted: 6 December 2024

Abstract
Self-organizing complex systems can be modeled using cellular automaton models. However, the parametrization of these models is crucial and significantly determines the resulting structural pattern. In this research, we introduce and successfully apply a sound statistical method to estimate these parameters. The decisive difference to earlier applications of such approaches is that, in our case, both the CA rules and the resulting patterns are discrete. The method is based on constructing Gaussian likelihoods using characteristics of the structures, such as the mean particle size. We show that our approach is robust for the method parameters, domain size of patterns, or CA iterations.

Keywords Cellular automaton · Discrete model · Parameter identification · Statistical approach

✉ Andreas Rupp
rupp@math.uni-sb.de

Alexey Kazarnikov
kazarnikov@gmail.com

Nadja Ray
nadja.ray@ku.de

Heikki Haario
heikki.haario@lut.fi

Joona Lappalainen
joona.lappalainen@lut.fi

1 Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Mathematikon, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

2 Mathematical Institute for Machine Learning and Data Science, Catholic University of Eichstätt-Ingolstadt, Hohe-Schul-Str. 5, 85049 Ingolstadt, Germany

3 School of Engineering Science, Lappeenranta–Lahti University of Technology, P.O. Box 20, 53851 Lappeenranta, Finland

4 Department of Mathematics, Faculty of Mathematics and Computer Science, Saarland University, 66123 Saarbrücken, Germany

Published online: 30 January 2025

1 Introduction

Cellular automaton (CA) models are widely used to describe self-organizing, complex systems such as tumor growth (Moreira & Deutsch, 2002), protein bioinformatics (Xiao et al., 2011), chemical reactions (Menshutina et al., 2020), formation and turnover of soil microaggregates (Ray et al., 2017; Zech et al., 2022), geospatial environmental modeling (Ghosh et al., 2017), urban planning (Santé et al., 2010), crowd evacuation (Yang et al., 2011), traffic flow (Tian et al., 2021), and microstructure evolution in metal forming (Yang et al., 2011). Within the framework of a CA, distinct states are assigned to so-called cells. These states may change according to prescribed transition rules depending on the states of the neighboring cells (e.g., within the neighborhood of a specific size).

Typically, several parameters influence a CA's rules, significantly determining the patterns the CA produces. Thus, it is crucial to determine these parameters since CAs produce significantly flawed simulations if they are set incorrectly. Exemplary, if we try to model the growth of microaggregates in soil, wrong parameters can lead to very large aggregates in computer simulations. Still, these large aggregates cannot be reproduced in *in vitro* experiments. If we model cancer development with CAs (Cooper et al., 2020), wrong parameters can produce inaccurate growth and decay rates.

However, reasonable parameter choices are often hard to identify. An apparent reason is the inherent randomness of CAs that leads to stochastic cost functions in the parameter identification scheme. While the literature on cellular automaton applications is huge, the literature on parameter calibration of CA models is much more sparse. To calibrate a model in urban dynamics, i.e., the spread of cities, parameter estimation is conducted via a genetic algorithm in Li et al. (2007). In this case, the transition rules depend on geographical variables, physical constraints, and uncertainty. Knowledge about the parameters can be used to improve urban planning towards compact cities, e.g., concerning energy and sustainable land usage. Likewise, neural networks were used to predict parameters in urban planning (Yeh & Xia, 2004) based on satellite remote sensing data and GIS (Geographic Information System). Finally, parameter estimation for CAs for predicting wildfire in Africa is found in Couce and Knorr (2010). The fire propagation was assumed to depend on environmental (vegetation, fuel/litter load, wind) and climatic factors. Here, parameter estimation was implemented by minimizing the KL (Kulback–Leibler) distance between modeled and observed fire extension histograms.

A common issue in the CA model's parameter estimation approaches is the lack of statistics. A more or less ad-hoc cost function is formulated and optimized, but no uncertainty quantification is presented. In this research, we develop a sound statistical approach to the CA parameter estimation problem. The starting point is an analogy of CA patterns with Turing models. These continuous models were proposed by Alan Turing in the seminal work (Turing, 1952) as a hypothetical mechanism describing the symmetry-breaking phenomenon at the early stage of morphogenesis. They later found applications in different areas, including modeling chemical reactions (Lengyel & Epstein, 1991, 1992), describing population dynamics (Zhu et al., 2024), social interactions (Ke et al., 2022; Yuan et al., 2023; Zhu & Yuan, 2023; Li & Zhu, 2024), and even morphochemical processes (Bozzini et al., 2015; Lawless et al., 2019; Frittelli

et al., 2024). For both Turing models and cellular automata, randomized initial values lead to random patterns. Our proposed statistical approach was earlier used to identify parameters of Turing models in continuous and network domains (Kazarnikov et al., 2020a; Zhu & He, 2022a, b), and initially introduced to calibrate chaotic dynamical systems (Haario et al., 2015; Springer et al., 2019). It is designed for systems with stochastic outcomes that may be due to unknown, randomized initial conditions or stochasticity of the model itself. The approach is based on creating statistics for scalar-valued characteristics computed directly from the patterns. The decisive difference to the earlier applications in Kazarnikov et al. (2020a, 2023) is that in this work, both the CA rules and the resulting CA patterns are discrete, while the algorithm of Kazarnikov et al. (2020a) has been developed for and applied to partial differential equation (PDE) based models, which yield continuous functions as solutions, that depend continuously on the model parameters. Thus, the previous algorithm took advantage of these facts, e.g., by using different Lebesgue and Sobolev norms to characterize properties of the patterns produced by the forward model. As opposed to this, CA models yield completely discrete and discontinuous results. If these results are interpreted as functions, these functions only take values in discrete subsets, such as in $\{0, 1\}$, prohibiting the evaluation of spatial gradients.

Consequently, their discrete analogs must replace the norms/metrics on continuous function spaces. On the other hand, various measures such as the Minkowski characteristics (Armstrong et al., 2019) or the mean particle size/particle size distribution are widely used to characterize structures. Our statistical approach can directly employ those measures. Indeed, considering such measures leads to sharper estimates than those adopted from the continuous model norms.

The main novelties of our algorithm comprise the ability to use arbitrary quantities in the parameter estimation process. Thus, we can build our parameter estimation process on relevant quantities in the respective scientific field. In soil science, such quantities are, e.g., the total surface, number of particles, average particle size, compactness ratio, and particle size distribution (which is not a scalar); see (Rupp et al., 2019) for a detailed overview. We illustrate this feature using the average particle size and particle size distribution.

As the CA simulation is inherently stochastic, standard likelihood constructs are not available, and we here indeed deal with a situation often discussed under the title 'intractable likelihood.' The statistical analysis for such cases is typically carried out with 'likelihood-free inference' methods, among which the ABC (Approximate Bayesian Computation) approach is the most common. Our approach's main difference and novelty is that we create a likelihood, even an empirically Gaussian one, by considering eCDF vectors derived from data. For scalar-valued data discussed above, the eCDF vectors can be directly computed. Still, we need to map to scalars for the 2D patterns created by CA simulations before eCDF vectors can be computed. Here, we use the L^1 distance between pairs of patterns. The particle size distribution is a nice feature, directly providing an eCDF vector. The concatenations of Gaussian vectors are again Gaussian, and the joint mean and covariance can be readily numerically estimated.

The paper is structured as follows: In Sect. 2, we introduce our cellular automaton method, which is used as "forward model" to produce patterns for our parameter

estimation method. The parameter estimation method is outlined in detail in Sect. 3. In Sect. 4, we discuss the results of our parameter estimation method. This includes a sensitivity analysis concerning the parameters of the CA and the parameter estimation method. We conclude the paper with an outlook to future research.

Notably, all used software for this paper is made publicly available in two software projects:

- The implementation of the CA model can be found in Rupp et al. (2022c). It is performed in C++ with a MATLAB interface using the mex compiler and a Python interface using the just-in-time compilation provided by HyperHDG Rupp et al. (2022a); Rupp and Kanschat (2021), which builds on Cython.[1]
- The package for parameter estimation can be found in Rupp et al. (2022b). It contains a Python implementation of the presented empirical cumulative distribution function (eCDF) based approach. This package can also be obtained from PyPI.[2]

2 Cellular automaton method

We now describe the cellular automaton method and illustrate it in two spatial dimensions, although it is implemented to work accordingly in any positive-integer dimensional setting. The complete, C++-based implementation can be found in Rupp et al. (2022c).

2.1 Setting of CA model

The CA model consists of a discretized domain, typically a d-dimensional cube, comprising N^d non-overlapping small cubes (so-called *cells*) c_m, $m = 1, \dots, N^d$, each having identical volumes. Within this domain, the spatiotemporal distribution of two phases ⓪ (e.g., *void*, white in Fig. 1) and ① (e.g., *solid*, black in Fig. 1) is considered. We write $c_m = 0$ if cell c_m attains state ⓪ and $c_m = 1$ if c_m attains ①. At the initial time, to all cells, either of the values ⓪ or ① is assigned, e.g., randomly. After that, the cells are redistributed within the domain in every time step according to specific parameter-dependent jumping rules, see Sect. 2.2. This results in the two-phase system's temporal evolution (self-organization) and a final arrangement of the cells (pattern).

In this study, we prescribe the porosity

$$\theta = \frac{\sum_{m=1}^{N^d}(1 - c_m)}{N^d} = \frac{\text{number of cells with state ⓪}}{\text{number of cells}} \in (0, 1)$$

of the system and derive the number of cells of type ①. These are then randomly distributed in the cubic domain N^d at initial time $t_0 = 0$. The remaining cells are associated with phase ⓪, i.e., the initial state s_0 is an element of the pattern space

[1] https://cython.org/

[2] https://pypi.org/project/ecdf-estimator/

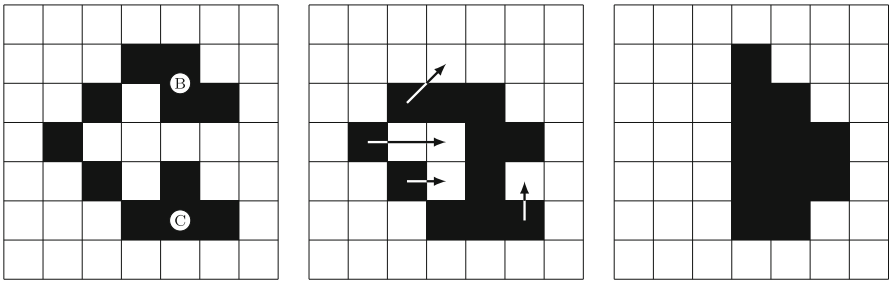


Fig. 1 The application of CA jumping rules within one time-step for the case $\sigma = 2$: First, all agglomerates move in random order to the positions with the highest number of neighbors: (B) changes its position, but (C) remains in its place because every possible movement decreases the number of neighbors (left picture). Then, all single cells (part of agglomerates or not) move to maximize the number of their neighbors. Arrows indicate the latter movement (center picture). The right picture shows the resulting structure (colour figure online)

$P = \{0, 1\}^{N^d}$. In fact, s_0 is generated by randomly selecting $N^d\theta$ cells out of the N^d cells (simple random sampling without replacement), i.e., $s_0 \sim U^\theta\{0, 1\}$. Moreover, we assume the domain to be periodic, i.e., we identify the left and the right boundary and the top and the bottom boundary with each other and likewise in higher dimensions.

2.2 Jumping rules for the CA model and related parameters

The jumping rules of our CA are designed such that phase (1) is compacted; see Fig. 1 for an illustration. The jumping rules depend on the choice of the neighborhood (see Fig. 2), which in turn depends on the jump parameter σ in some parameter set $\Pi \subset \mathbb{N}$, and the evaluation of the attractivity of new spots. Thereby, the parameter choice highly influences the self-organization of the system and the pattern obtained; see also illustration in Figs. 3 and 4.

2.2.1 Von-Neumann neighborhoods

First, the parameter $\sigma \in \Pi$ determines the size of the neighborhood, which is considered to decide about possible jumps of single cells with state (1) to more attractive spots. It describes the range of the von Neumann neighborhood (VNN), which is the most commonly used distance in CA applications, given by

$$\text{range VNN(cell)} = \max\{1, \sigma\}.$$

As illustrated in Fig. 2, for a single cell (A), the VNN of size 1 ($\sigma = 1$) consists of (A) and its face-wise neighbors, i.e., four neighbors in the two-dimensional space (illustrated in black in Fig. 2). A VNN of size 2 ($\sigma = 2$) consists of the VNN of size 1 and all face-wise neighbors of all cells contained in the VNN of size 1, i.e., 12 neighbors in the two-dimensional space (illustrated in black and red in Fig. 2), etc. Depending on the choice of the parameter σ, the single cells of type (1) can move within a smaller or larger region to find more attractive spots; see Sect. 2.2.2 below.

More formally, the VNN of range r around cell c consists of all cells that can be reached by c when it performs at most r consecutive moves into one of its face-wise neighbors. Analogously, the VNN of range r of a set of cells C comprises all cells that can be reached by any cell in C when it conducts at most r consecutive moves into its face-wise neighbors.

Second, the parameter σ is used to determine the size of the VNN, in which agglomerates (ag), i.e., composites of face-wise connected cells of type ①, are allowed to move. The following definition realizes this:

$$\text{range VNN(ag)} = \max\left\{1, \left\lfloor \frac{\sigma}{\sqrt[d]{\mu(\text{ag})}} \right\rfloor\right\},$$

where $\lfloor\cdot\rfloor$ indicates the floor function, i.e., rounding down to the next integer, d is the spatial dimension (e.g., two), and $\mu(\text{ag}) > 1$ is the size of ag. It is defined as the number of cells of which ag consists. In the case of Ⓑ or Ⓒ in Fig. 1, for instance $\mu(Ⓑ) = \mu(Ⓒ) = 4$ holds.

From the above equations, it is obvious that σ is the decisive parameter when it comes to the question of how far a cell or aggregate can move and what the patterns that the CA produces look like, see Fig. 3. The equations themselves stem from the reasoning that soil aggregates diffusion is proportional to the inverse of their diameter, which is ~ 1 for single cells, and $\sim \sqrt[d]{\mu(\text{ag})}$ for a ball-like aggregate.

2.2.2 Movement of agglomerates and single cells

Algorithm 1 Cellular automaton, the definition of non-linear forward operator $F_{\text{CA}}(\sigma)\colon P \to P$, where $P = \{0,1\}^{N^d}$ denotes the pattern space. This operator implements a single time-step of the CA model. In this work, we define as a pattern the state vector $s \in P$ obtained by applying operator $F_{\text{CA}}(\sigma)$ n^* times to the initial random state $s_0 \in P$, i.e. $s = [F_{\text{CA}}(\sigma)]^{n^*} s_0$.

Input: $s \in P$, the current state of the CA model
Input: $\sigma \in \Pi$, the jumping rate parameter
Construct set of agglomerates A_s in state s.
for each agglomerate ag $\in A_s$ (in random order) **do**
    Calculate VNN(ag) depending on jump parameter σ
    Evaluate the attractivity of all possible positions in VNN(ag)
    Choose a new position for ag, maximizing the attractivity
    Move ag to the new position
end for
Construct the set of all cells with state ① C_s.
for each cell $c \in C_s$ (in random order) **do**
    Calculate VNN(c) depending on jump parameter σ
    Evaluate the attractivity of the members of the VNN(c)
    Choose a new position for c, maximizing the attractivity
    Move cell c to the new position
end for
Output: Updated state vector s, i.e., $F_{\text{CA}}(\sigma)s$

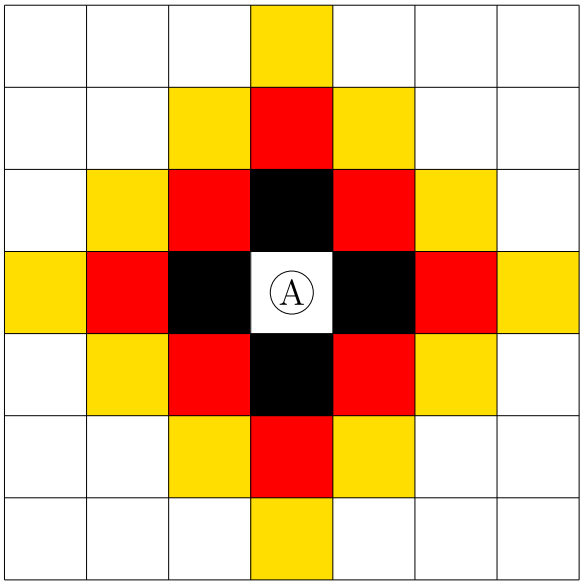


Fig. 2 Illustration of VNN around Ⓐ of range 1 (black), 2 (red), and 3 (yellow) in two spatial dimensions (colour figure online)

Starting from an initial state s_0 in the pattern space P, a new pattern s in the pattern space P is eventually created - first due to the movement of single cells and then due to the subsequent movement of single cells and agglomerates to new positions for a prescribed amount of time steps, see Algorithm 1. The attractivity of potential new spots within the VNN is evaluated in each time step for the actual movement of the agglomerates and single cells. Maximizing attractivity involves trying all possible moves and selecting the one with the highest attractivity values.

First, all agglomerates within the domain N^d are identified. These are Ⓑ and Ⓒ in the two-dimensional example as illustrated in Fig. 1. The agglomerates are randomly ordered, and their potential movement within VNN(ag) is evaluated successively. Since the CA should compactify phase ①, each agglomerate's jumping is chosen to maximize the number of its direct neighbors. If several equally attractive new spots exist for an agglomerate, one of them is randomly selected.

Let us assume that Ⓑ is the first agglomerate to move in the two-dimensional example of Fig. 1, and that it may move in a VNN of 1 for the parameter choice $\sigma = 2$. This means the agglomerate can remain in its actual position, moving to the left, right, upwards, or downwards. The most attractive new spot can here be achieved by moving downwards. After Ⓑ has moved, Ⓒ may move, but the number of neighbors will decrease if Ⓒ changes its position. Thus, it does not move to a new spot.

After all the agglomerates have moved, all single cells (part of agglomerates or not) may move within their VNN to find new and more attractive spots. Again, the order of the movement of the single cells is random, and the single cells move such that they end up with a maximum amount of direct neighbors. If there are two equally beneficial moves, one of those is randomly selected. In the two-dimensional example

illustrated in Fig. 1, these movements are indicated in the middle picture by arrows for the parameter choice $\sigma = 2$.

Moving agglomerates and single cells is repeated several times (CA steps). In this sense, the forward model maps an initially disordered state vector $s_0 \sim U^{\theta}\{0, 1\}$ into a pattern $s = F_{\mathrm{CA}}^{n^*}(\sigma)s_0$. This process naturally depends on the jumping parameter $\sigma \in \Pi$ and porosity parameter θ. We will assume that θ is fixed and use the parameter estimation method to recover parameter $\sigma \in \Pi$ from patterns. This is outlined below in Sect. 3. Note that due to the random origin of s_0, distinct patterns are emerging even for a given maximal size of the VNN (prescribed by the value of σ), and the inverse problem becomes a non-trivial task.

2.3 Application of cellular automaton method for different parameter sets

We apply the cellular automaton as introduced in Sect. 2.2 to illustrate the corresponding pattern formation for different choices of parameters. Starting from dispersed, randomly created structures, according to the CA jumping rules, single cells and agglomerates attract each other and finally form larger clusters and potentially connected structures. This process is illustrated in Fig. 3 for different choices of the jump parameter

$$\sigma \in \{1, 5, 10, 15\},$$

and different porosities $\theta \in \{0.3, 0.5, 0.7, 0.9\}$. The dynamic structure development concerning time is shown in Fig. 4 for two different porosities $\theta \in \{0.5, 0.9\}$ and jump parameters $\sigma \in \{1, 5\}$. Distinct patterns emerge depending on the specific parameter choice. Larger porosities lead to dispersed structures, while larger jump parameters lead to blocky patterns. On the other hand, smaller porosities and smaller jump parameters induce card-house-type structures.

Besides the illustrations of the cellular automaton model for two spatial dimensions in Figs. 3 and 4, the method can also be applied to model self-organization in three spatial dimensions as shown in Fig. 5. Likewise, it can also be applied in higher spatial dimensions using the implementation of Rupp et al. (2022c).

The resulting patterns are input for further analysis using parameter estimation methods outlined below in Sect. 3.

3 Parameter estimation method

We now introduce the parameter estimation method we apply to the results generated by the CA as outlined in Sect. 2. Our method for parameter identification allows us to map a training set of patterns to a Gaussian distribution. This will enable us to define a statistical likelihood and use it as a cost function during the parameter identification. Our approach relies on emerged pattern data only, without using the information about the initial data.

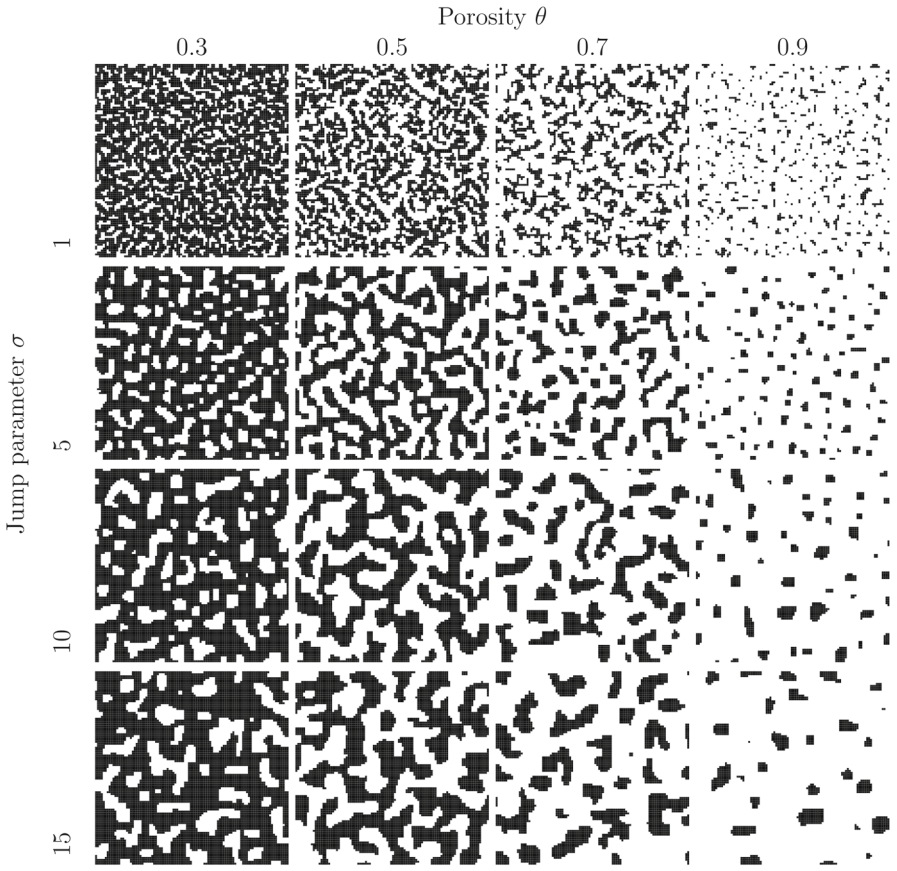


Fig. 3 Illustration of the domain consisting of 100×100 cells after five steps of the CA for varying porosity θ and jump parameter σ (colour figure online)

3.1 Background

According to the central limit theorem, the average of random variables with finite expected value and variance converges to the normal distribution. This allows for using a Gaussian likelihood for parameter estimation if enough repeated measurements are available. However, the mean may be rather uninformative, as quite different distributions can have the same mean (and higher moments). The cumulative distribution function (CDF) can create more accurate statistics. In probability theory, Donsker's theorem is a functional extension of the central limit theorem. In this work, we build on generalizations of the Donsker theorem (Donsker, 1952), which states that the cumulative distribution function of independent and identically distributed (i.i.d) scalar samples converges towards a Gaussian vector.

Theorem 3.1 (Donsker, Skorokhod, Kolmogorov) *Let F_n be the empirical distribution function of the sequence of i.i.d. random variables $X_1, X_2, \ldots, X_n$ with distribution*

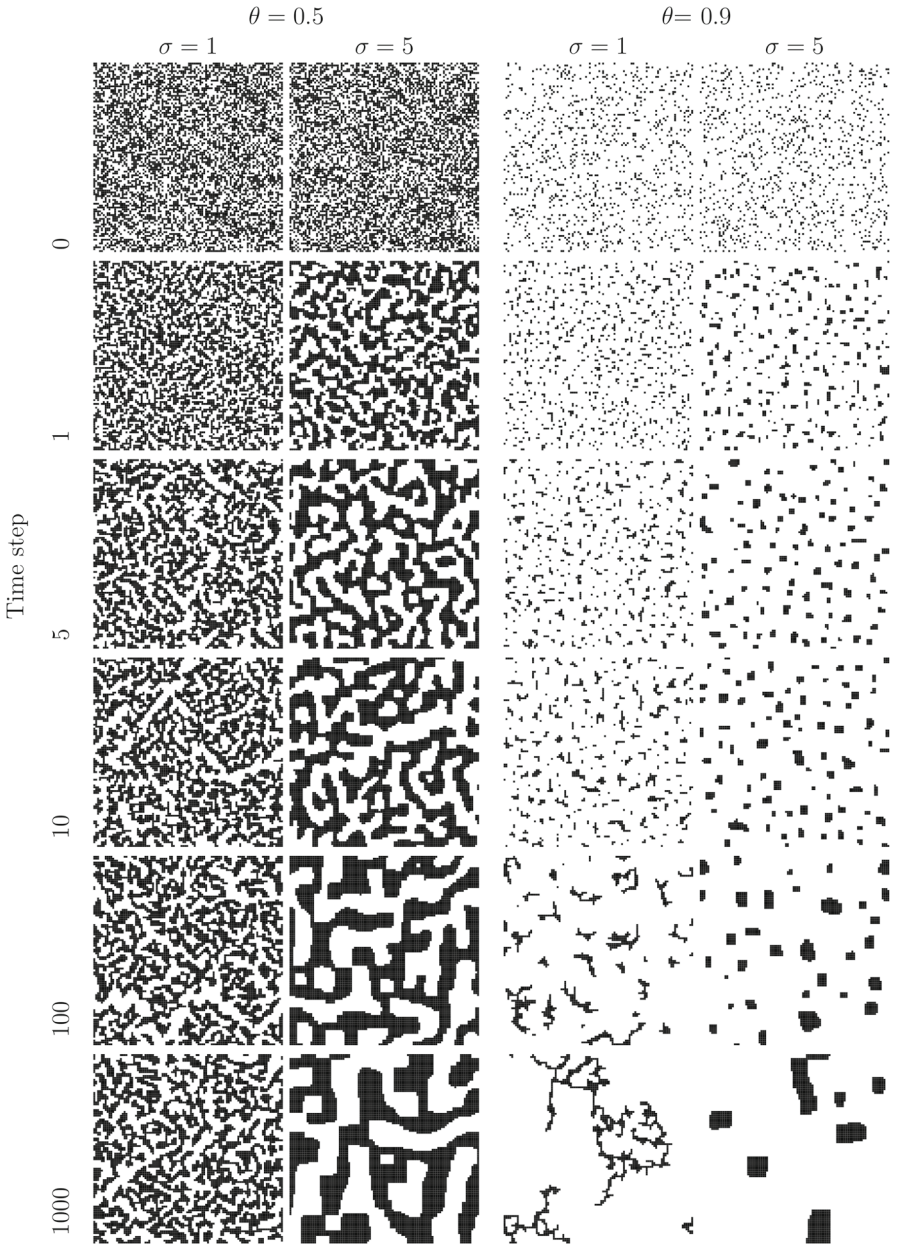


Fig. 4 Illustration of the time evolution of the domain consisting of 100×100 cells after steps 0, 1, 5, 10, 100 and 1000 of the CA for porosity $\theta \in \{0.5, 0.9\}$ and jump parameter $\sigma \in \{1, 5\}$ (colour figure online)

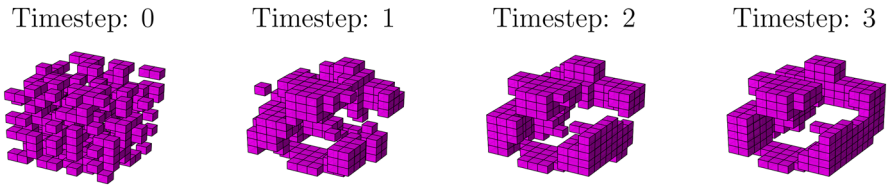


Fig. 5 Illustration of the time evolution of the domain consisting of $10 \times 10 \times 10$ cells after steps 0, 1, 2 and 3 of the CA for porosity $\theta = 0.7$ and jump parameter $\sigma = 5$ (colour figure online)

function F. Define the centered and scaled version of F_n by

$$G_n(x) = \sqrt{n}(F_n(x) - F(x)).$$

The sequence of $G_n(x)$ converges in distribution to a Gaussian process G with zero mean and the covariance is given by

$$\mathrm{cov}[G(s), G(t)] = E[G(s)G(t)] = \min\{F(s), F(t)\} - F(s)F(t).$$

We use the theorem in an approximative form for finite data. For i.i.d. scalar data with sample size N, the empirical distribution function (eCDF) computed at selected bin values x_i, $i = 1, 2, \ldots, M$, becomes a M-dimensional Gaussian vector, with mean $\boldsymbol{F}_0 \in \mathbb{R}^M$ and covariance given by

$$(\boldsymbol{\Sigma}_0)_{ij} = (\min((\boldsymbol{F}_0)_i, (\boldsymbol{F}_0)_j) - (\boldsymbol{F}_0)_i(\boldsymbol{F}_0)_j)/N, \quad i, j = 1, \ldots, M.$$

The basic form of the Donsker theorem applies to i.i.d scalar situations. In our application, the data is not i.i.d. The covariance formula cannot be used then, but data or simulated eCDF vectors can estimate the covariance matrix. The Gaussianity still holds, assuming that conditions on weakly dependent data hold (Borovkova et al., 2001; Neumeyer, 2004).

Also, our data is inherently high dimensional, so a scalar-valued mapping must first be used to construct eCDF vectors; see (Kazarnikov et al., 2020a; Springer et al., 2019; Haario et al., 2015) for earlier examples. We discuss the construction of such scalar-valued mappings below.

Note the approximative character of the approach in a finite setting. As eCDF vectors are strictly limited in the interval (0, 1), the normality cannot hold close to the tails. In numerical applications, standard scalar normality tests can be used to verify the normality at the bin values used, and the M-dimensional χ^2 test can be used for the Gaussianity of the eCDF vectors.

3.2 Construction of the approach

Let us represent the CA model as an abstract pattern formation model depending on a p-dimensional vector of model parameters $\boldsymbol{\sigma} \in \Pi \subset \mathbb{R}^p$. In our specific case, $\boldsymbol{\sigma} = \sigma \in \mathbb{N}$, thus $p = 1$, but the construction of the approach remains the same

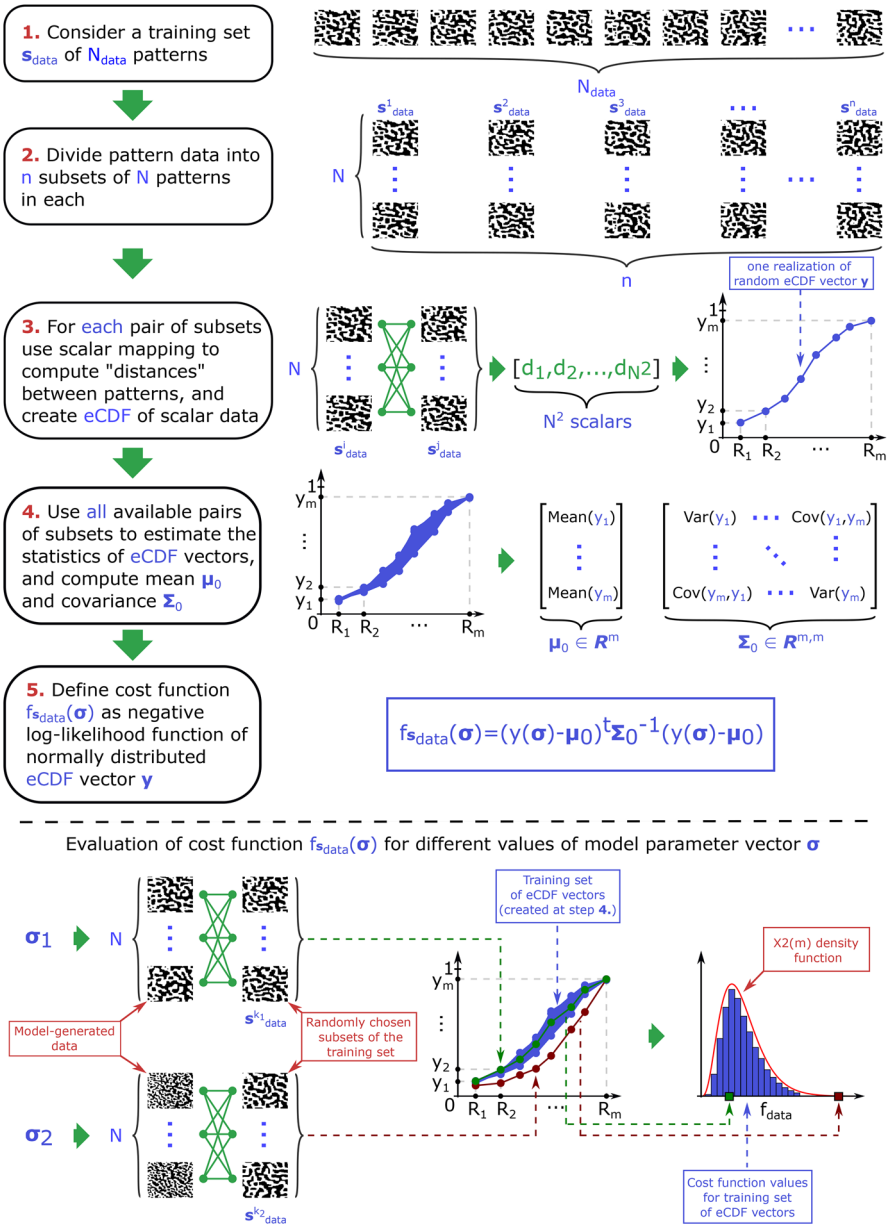


Fig. 6 Construction and evaluation of the cost function for parameter identification by pattern data (colour figure online)

for a larger number of parameters. The output of each "forward model" run is a pattern $s(\boldsymbol{\sigma}) \in P$, which corresponds to the final state after a prescribed number n^* of applications of the cellular automaton forward operator $F_{\mathrm{CA}}(\sigma)$ as introduced in Sect. 2: $s(\boldsymbol{\sigma}) = [F_{\mathrm{CA}}(\sigma)]^{n^*} s_0$, $s_0 \sim U^{\theta}\{0, 1\}$. Applying the forward model for various choices of initial random state s_0 and fixed parameter(s) $\boldsymbol{\sigma}$, a set of patterns $\boldsymbol{s}(\boldsymbol{\sigma})$ is obtained. These patterns naturally change due to the variation of the model parameter(s) but additionally change even for the fixed model parameter(s) due to the random distribution of the two phases at an initial time and the randomness included in the CA steps. We aim to distinguish this internal variability from the systematic changes due to varying model parameters.

More precisely, we want to find all the model parameters that fit a given training data, i.e., a set of patterns $\boldsymbol{s}_{\mathrm{data}} \subset \boldsymbol{s}(\boldsymbol{\sigma})$, within the accuracy allowed by the data. To do so, we define a minimization problem in terms of a stochastic cost function

$$f_{\boldsymbol{s}_{\mathrm{data}}}(\boldsymbol{\sigma}) \to \min \qquad \Longleftrightarrow \qquad \bar{\boldsymbol{\sigma}} = \operatorname{argmin}_{\boldsymbol{\sigma}} f_{\boldsymbol{s}_{\mathrm{data}}}(\boldsymbol{\sigma}), \tag{3.1}$$

and consider any argument $\bar{\boldsymbol{\sigma}}$ that solves (3.1) as a model parameter vector that corresponds to the training data set $\boldsymbol{s}_{\mathrm{data}}$. The remainder of this section is devoted to constructing $f_{\boldsymbol{s}_{\mathrm{data}}}$ step-by-step.

First, we specify what we mean by a solution to problem (3.1). Due to the stochasticity of the model, a given model parameter corresponds to a distribution of solutions. We thus distinguish different model parameters by the respective distributions they produce. As the pattern data is high-dimensional, we define some measures to quantify the "distance" between two samples. For this purpose, we employ the training data to construct a statistical likelihood function that quantifies the variability within the data, i.e., gives a distribution of acceptable solutions. The basic idea is to define a "distance" mapping ρ, e.g., a scalar mapping, which compares two patterns $\boldsymbol{s}^+, \boldsymbol{s}^- \in P$ (possible choices of ρ will be discussed below). The full statistics of ρ are then used to produce a Gaussian likelihood based on/from the training data.

We next show how to construct the function $f_{\boldsymbol{s}_{\mathrm{data}}}$ for a given distance and all the training data pairs. To employ the training data statistically, we divide the set of patterns into n subsets. We define a function that accepts two arguments: the sets of patterns $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ containing N^+ and N^- patterns, respectively. Apart from the two arguments, it depends on two parameters: a radius $R > 0$ and the "distance" ρ:

$$C(\boldsymbol{s}^+, \boldsymbol{s}^-; R, \rho) = \frac{1}{N^+ \times N^-} \sum_{i=1}^{N^+} \sum_{j=1}^{N^-} \#(\rho(\boldsymbol{s}_i^+, \boldsymbol{s}_j^-) < R), \tag{3.2}$$

where # denotes the discrete indicator function.

The radius $R > 0$ is used to create a vector $\boldsymbol{y}$ that encodes the similarity of two sets of patterns. Hence, we define a vector of bin values $(R_i)_{i=1}^m$, and set

$$\boldsymbol{y}(\boldsymbol{s}^+, \boldsymbol{s}^-) = \boldsymbol{y}(\boldsymbol{s}^+, \boldsymbol{s}^-; (R_i)_{i=1}^m, \rho) = \left(C(\boldsymbol{s}^+, \boldsymbol{s}^-; R_i, \rho)\right)_{i=1}^m .$$

This vector represents the eCDF of the set of values $\rho(\boldsymbol{s}_i^+, \boldsymbol{s}_j^-)$ evaluated at the bin values $(R_i)_{i=1}^m$.

We quantify the statistics (mean and variance) of $\boldsymbol{y}(\boldsymbol{s}^+, \boldsymbol{s}^-)$ among subsets $\boldsymbol{s}^+, \boldsymbol{s}^-$ of the whole training set $\boldsymbol{s}_{\text{data}}$: For each subset pair, we receive N^2 scalar distance values, from which a single eCDF vector is computed. Repeating this for all distinct $n(n-1)/2$ pairs

$$\boldsymbol{y}^{k,l} = \boldsymbol{y}(\boldsymbol{s}_{\text{data}}^k, \boldsymbol{s}_{\text{data}}^l) \in \mathbb{R}^m \qquad 0 \le k < l \le n,$$

we can evaluate the mean $\boldsymbol{\mu}_0 \in \mathbb{R}^m$ and the covariance $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{m,m}$ of all the pairs of distinct eCDF vectors.

As for most applications, the normality is numerically verified here as follows: We test for Gaussianity of the ensemble of vectors using the χ^2-test (or scalar normality tests for the vector components at the bin values). For this purpose, we evaluate all the values of the negative log-likelihood function

$$(\boldsymbol{y}^{k,l} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{y}^{k,l} - \boldsymbol{\mu}_0),$$

and compare the resulting histogram against the density function of the distribution χ_M^2 with m degrees of freedom.

To evaluate the cost function at a new parameter value $\boldsymbol{\sigma}$, we simulate the CA model N times using $\boldsymbol{\sigma}$ and denote the collection of patterns by $\boldsymbol{s}(\boldsymbol{\sigma})$. The eCDF vector

$$\boldsymbol{y}(\boldsymbol{\sigma}) = \boldsymbol{y}(\boldsymbol{s}(\boldsymbol{\sigma}), \boldsymbol{s}_{\text{data}}^k) \tag{3.3}$$

can then be computed for one randomly selected $k \in \{1, 2, \ldots, n\}$, and the likelihood value is evaluated as

$$f_{\boldsymbol{s}_{\text{data}}}(\boldsymbol{\sigma}) = (\boldsymbol{y}(\boldsymbol{\sigma}) - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{y}(\boldsymbol{\sigma}) - \boldsymbol{\mu}_0). \tag{3.4}$$

To summarize, the 'forward model' is given by the CA algorithm that produces the collection of patterns by $\boldsymbol{s}(\boldsymbol{\sigma})$ for a given value of $\boldsymbol{\sigma}$. The observation operator is defined by applying formula (3.3) that maps the set of model-generated patterns $\boldsymbol{s}(\boldsymbol{\sigma})$ to an eCDF vector $\boldsymbol{y}(\boldsymbol{\sigma})$. The 'inverse problem' of parameter estimation is solved by evaluating the stochastic cost function $f_{\boldsymbol{s}_{\text{data}}}$ for a given value of $\boldsymbol{\sigma}$ by using expression (3.4) and finding the minimum of it concerning $\boldsymbol{\sigma}$.

Note that in the examples of the present work, we only work with one parameter, the integer-valued σ, so the parameter estimation can be performed simply with a direct search.

3.3 Numerical implementation

3.3.1 Choice of bins

The bin values for the eCDF vectors can be selected in various ways. Here, we use the following approach: We first use the first two subsets of the training data and determine

the minimum and maximum of the function C as defined in (3.2) (concerning R). Next, we uniformly split the respective interval into 50 possible bin values. This allows us to represent the shape of the eCDF curve. However, in the numerical application, such dense coverage might result in an unwanted correlation between neighboring bin values. Therefore, from these preliminary bins, we choose a smaller subset of n_{bin} values, which we select by an inverse CDF method: to get the bin values on the x-axis, a set of n_{bin} linearly spaced values on the y-axis are mapped to the x-axis by the inverse CDF (quantile) function, using the mean of the preliminary CDF vectors. Additionally, a cut-off parameter of 0.1 is used to step away from the CDF function's range [0, 1] boundaries. This allows us to exclude bins with possibly prohibitively small variabilities, which might result in singularities in the covariance matrix of the underlying Gaussian distribution. After that, the Gaussianity of both bin configurations (the preliminary and the selected sparse one) can be evaluated.

3.3.2 Choice of characteristics

The algorithm of Kazarnikov et al. (2023) employs several norms to characterize the distance between two patterns in continuous-valued images. As the present setting is discrete, with binary-valued patterns, we use the L^1-norm, i.e., we define the distance function $\rho(\boldsymbol{s}^+, \boldsymbol{s}^-) = \|\boldsymbol{s}^+ - \boldsymbol{s}^-\|_{L^1}$ between two patterns $\boldsymbol{s}^+, \boldsymbol{s}^-$.

However, various other candidates for characteristics are reasonable for the discrete CA patterns. Here, we use the particle size distribution (PSD) and the average particle size (APS), which are frequently considered in soil science (Rupp et al., 2019). For the former case, let us consider a vector containing the sizes of all agglomerates $\boldsymbol{\mu}(\mathrm{ag})$.

Let us consider the first two patterns in the example of Fig. 1. The PSD for the left figure (pattern s^+) reads (1, 1, 1, 4, 4) as three single cells and two aggregates of size four are present. In the middle figure (pattern s^-), two single cells and one agglomerate of size 9 are present, i.e., the PSD reads (1, 1, 9). The average particle size is the mean of the particle size distribution, i.e., it is defined as the arithmetic average of the sizes of all particles, i.e., single cells and agglomerates. In our example this leads to $\mathrm{APS}(s^+) = 2.2$ and $\mathrm{APS}(s^-) = 3.67$, respectively. The distance function is given by

$$\rho(\boldsymbol{s}^+, \boldsymbol{s}^-) = |\mathrm{APS}(\boldsymbol{s}^+) - \mathrm{APS}(\boldsymbol{s}^-)|.$$

As our approach is based on the statistical distribution of the CDF functions of scalar data, we can also employ the particle size distribution directly by forming its empirical CDF. Although the particle size distribution could be computed for each pattern separately, we compute the particle size distributions of all pairwise combinations of patterns. In this case, the "distance" is the concatenated vector,

$$\rho(\boldsymbol{s}^+, \boldsymbol{s}^-) = \big(\mathrm{PSD}(\boldsymbol{s}^+), \mathrm{PSD}(\boldsymbol{s}^-)\big)$$

and the indicator function in (3.2) must be replaced by a counting function. The concatenation of the pattern from our example leads to (1, 1, 1, 4, 4, 1, 1, 9). For the choice of $R = 2$, the evaluation of ρ, for instance, leads to 5 counts. This has the

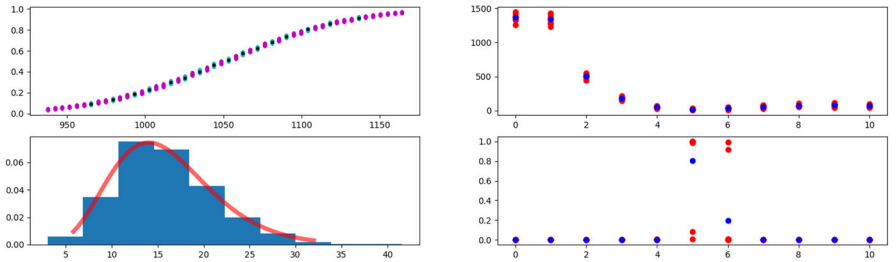


Fig. 7 Construction of the statistical likelihood for basic experimental setup and its evaluation for different values of jump parameter σ. On the top left is the distribution of the eCDF curves for dense bin values (purple) and selected bins only (light blue). Bottom left: Gaussianity test by χ^2 criterion for selected bin values. Top right: repetitive evaluation of the negative log-likelihood (cost function) for integer values of jump parameter σ on the interval [0, 10]. Here, cost function values are plotted in red, while blue dots denote the average overall evaluations. Bottom right: normalized likelihood values (red) and averaged values over evaluations (blue) (colour figure online)

advantage of producing more eCDF vectors, stabilizing the numerical estimation of the mean and covariance of the likelihood.

3.3.3 Application of the eCDF method to the base setup

We perform numerical experiments to demonstrate how our parameter estimation method can be applied to identify parameters of the CA model from synthetic (model-generated) data. We first consider a basic experimental setup, which is defined as follows:

- The CA is run on a two-dimensional domain of size 50×50 with porosity $\theta = 0.7$ and jump parameter $\sigma = 5$.
- The training set of patterns is obtained by running five iterations of the CA model with random initial data ($4000 = 40 \times 100$ realizations).

For this basic experimental setup, we create the L^1 likelihood and estimate the model parameters using the L^1 norm.

Following the procedure outlined in Sect. 3 and illustrated in Fig. 6, we create a statistical likelihood from the training data. We split the training set into 40 subsets with 100 samples in each. Next, for every pair, we compute distances between the respective patterns in terms of L^1-norm and compute the eCDF of the individual scalar data. Here, we successively select bins for the eCDF vectors using the algorithm described in Sect. 3.3.1. The selection is illustrated in Fig. 7 (top left).

Next, we check the Gaussianity of the selected bins as outlined in Sect. 3, which is illustrated in the bottom left part of Fig. 7. We evaluate the negative log-likelihood for integer values of the jump parameter σ on the interval [0, 10], as is shown in Fig. 7 (top right). Here, the red dots represent the evaluation of this value 100 times, and the blue dots highlight the average values of the red dots. The minimum negative log-likelihood values are achieved at $\sigma = 5$. Finally, the proper probabilistic interpretation is obtained by the normalized likelihood values in Fig. 7 (bottom right)—with the same understanding of red and blue dots. Thus, our approach can properly distinguish patterns corresponding to different values of σ for this setup.

4 Results

Next, we investigate the robustness of the parameter estimation method as introduced in Sect. 3 concerning the number of bins, the domain size, and the number of time steps (iterations) in the CA model. Here, we change only one quantity at a time, while the other ones are fixed to the default values as prescribed by the basic experimental setup in the previous Section. Finally, we analyze the impact of including additional CA-specific characteristics to the scheme of the parameter estimation method, which is discussed in Sect. 4.4.

4.1 Varying number of bins

We first study the robustness of our parameter estimation method for the number of selected bins (dimension of eCDF vectors). We repeat the experiments described in Sect. 3.3.3 for varying bins between 6 and 26 while keeping the other parameters fixed. The results were statistically identical to the ones shown in Fig. 7 for all considered cases. From this result, we conclude that the approach allows for relatively large flexibility concerning the number of bins once the numerical stability considerations discussed in Sect. 3.3.1 are considered.

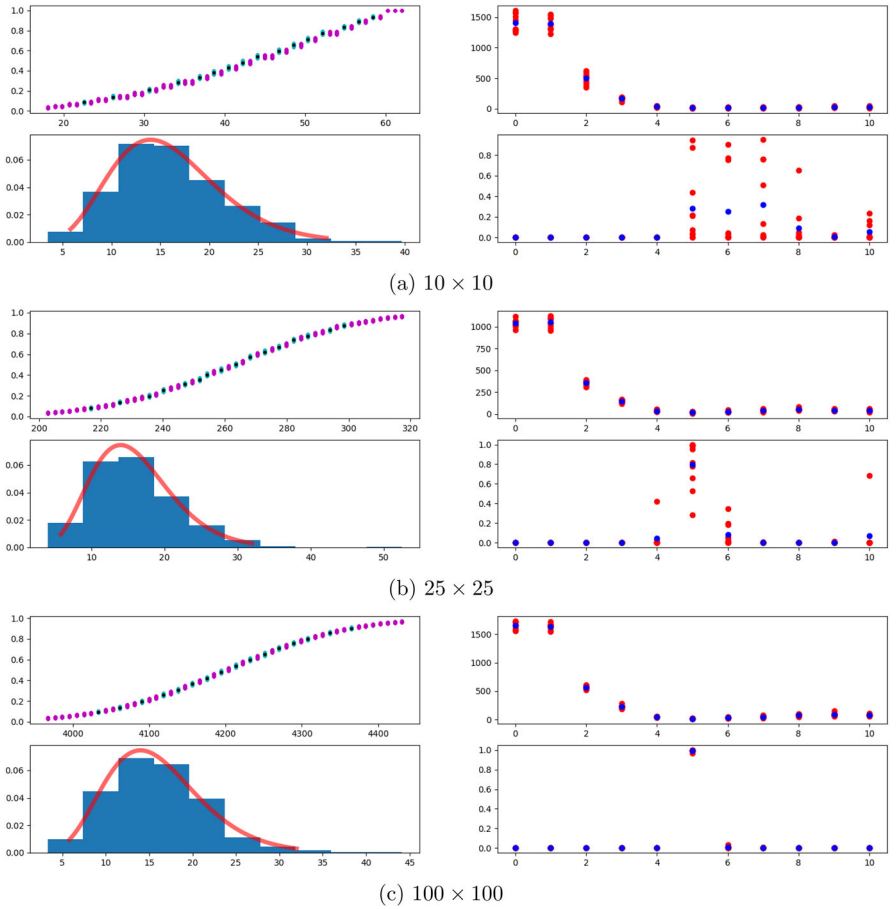
4.2 Varying domain sizes

We now consider different two-dimensional domain sizes, i.e., we perform the procedure of Sect. 3.3.3 for domains of size $N^2 = 10 \times 10$, 25×25, and 100×100. A simulation conducted on a small domain provides less information than a large one but keeps the porosity fixed. This is underpinned by the numerical experiments illustrated in Fig. 8.

Considering first the 10×10 domain, our algorithm can correctly identify the true parameter $\sigma = 5$. However, the minimum is not very pronounced, so the accuracy is relatively low. For domain sizes of 25×25, 50×50 (basic experimental setup, see Fig. 7), and 100×100, we observe smooth eCDF shapes again. Thus, the parameter estimation method works accurately and successfully. The increase in domain size leads to a more pronounced minimum at $\sigma = 5$. Therefore, the algorithm is more confident in identifying the correct jump parameter if the domain size and available information increase.

4.3 Varying number of CA iterations

The patterns generated by our CA model are not stationary since initially fragmented particles eventually attract each other and self-organize into larger, connected structures. Thus, depending on how many model iterations were used to create a pattern, it can be more or less clustered; see also illustration in Fig. 4. Here, we study how this factor affects our ability to perform parameter identification. The results are illustrated in Fig. 9. We start by considering the trivial case of zero iterations. In this case, the



(a) 10×10

(b) 25×25

(c) 100×100

Fig. 8 Construction of the statistical likelihood for various domain sizes and its evaluation for different values of jump parameter σ. The layout of sub-figures is identical to Fig. 7 (colour figure online)

dispersed initial state is independent of σ; thus, we can construct the likelihood without any problems. Still, we naturally cannot detect any difference between different values of σ. However, the parameter identification works without issues for all other considered cases: 1, 5 (base experimental setup, see Fig. 7), 10, 25, or 50 iterations. This indicates that the number of iterations does not significantly influence the quality of our parameter estimation method.

4.4 Multiple features in parameter estimation method

In this section, we again consider the basic experimental setup and discuss how the parameter identification procedure can be improved by considering additional features typical for characterizing structures of CA models as outlined in Sect. 3. Computing the distance between pattern data using L^1-norm allowed us to correctly identify

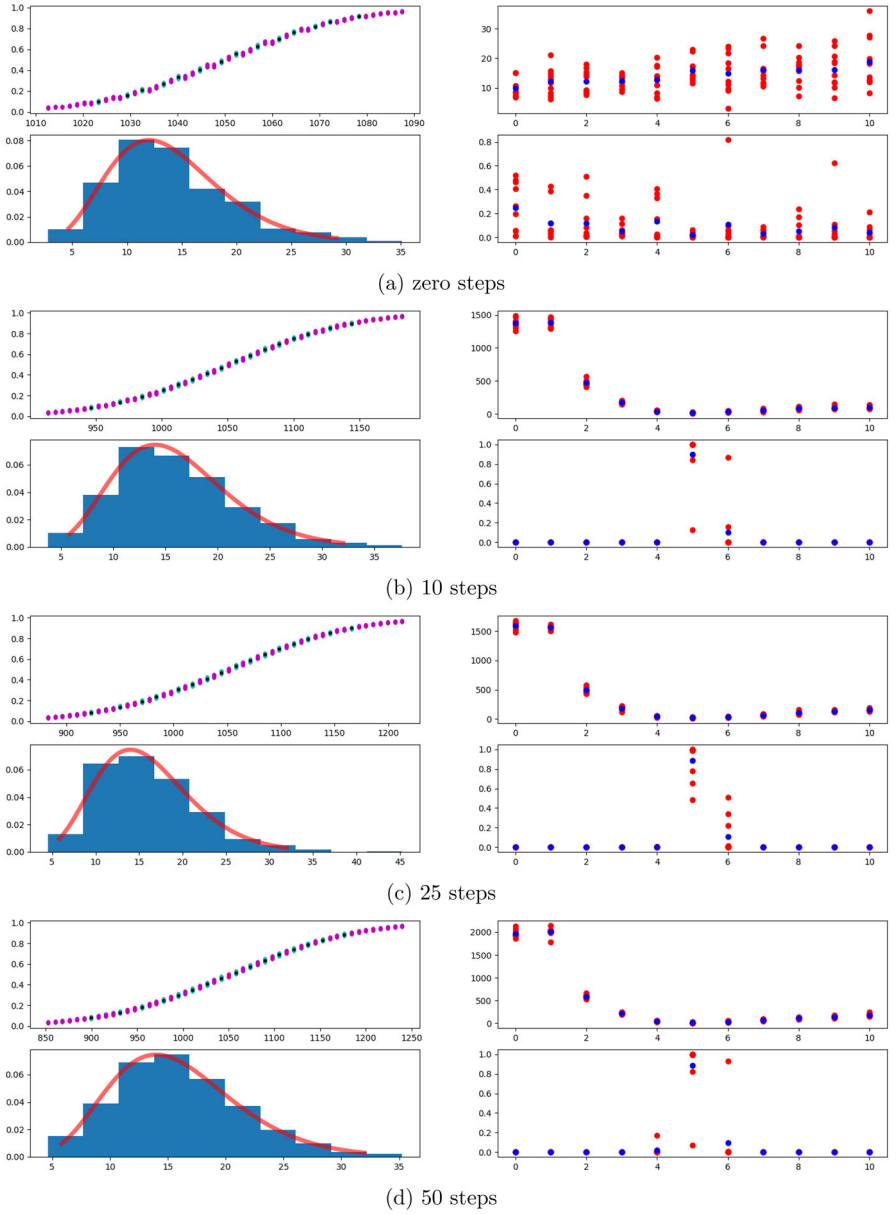
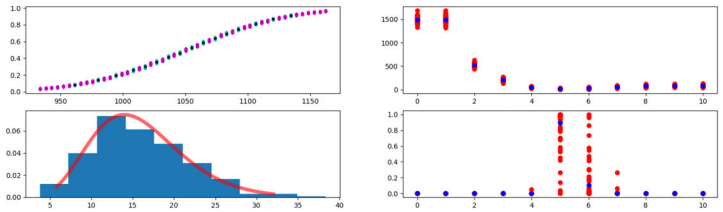
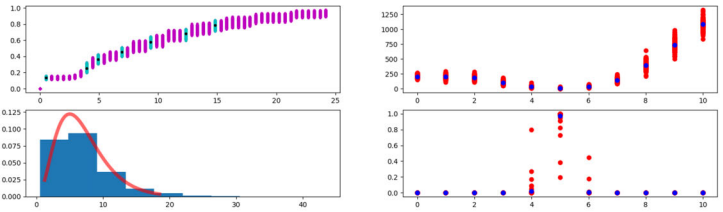


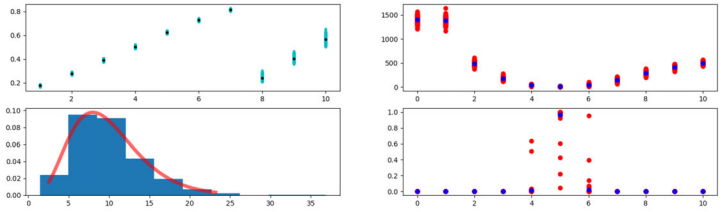
Fig. 9 Construction of the statistical likelihood for various CA iterations, and its evaluation for different values of jump parameter σ. The layout of sub-figures is identical to Fig. 7 (colour figure online)



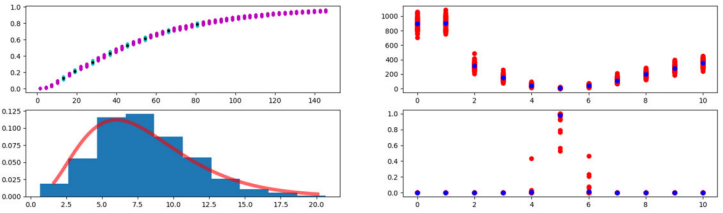
(a) L^1-distance



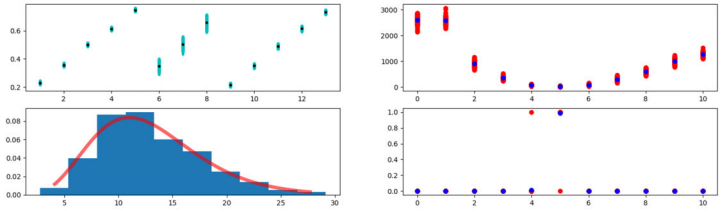
(b) absolute difference of average particle sizes



(c) L^1-distance + absolute difference of average particle sizes



(d) distribution of particle sizes



(e) L^1-distance + both characteristics

Fig. 10 Construction of the statistical likelihood for multiple characteristics in parameter estimation method, and its evaluation for different values of jump parameter σ. The layout of sub-figures is identical to Fig. 7 (colour figure online)

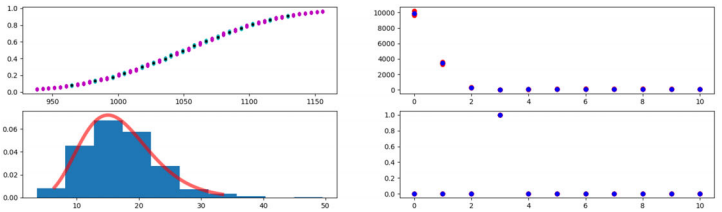
the correct value of jump parameter σ (see Fig. 10a). However, the minimum cost is not always well pronounced. This may result in higher uncertainty in parameter identification and can be seen from the wide spread of the red dots. We can, however, significantly reduce the uncertainty by additionally using the characteristics introduced in Sect. 3.

The first feature we use here is the average particle size. This means that instead of computing the L^1-distance between two patterns, we use the absolute difference of the respective average particle sizes. We observe that if we replace the L^1-distance with this new mapping, the variability of the cost function is nicely reduced, and the minimum becomes more pronounced (see Fig. 10b). An even better effect, however, can be achieved by combining these characteristics with the previously used L^1-distance (see Fig. 10c). Since the concatenation of the respective eCDF vectors is again Gaussian, the same approach can be applied directly.

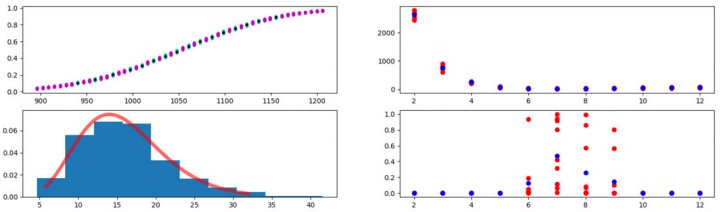
Next, we consider a different mapping from a pair of subsets of pattern data to one eCDF vector. That is, we do not use a distance that produces a scalar from a pair of patterns but directly create a distribution that holds the information of several scalars. Here, we employ the fact that an eCDF vector is directly available in the form of the particle size distribution of each CA pattern. Thus, we pool the particle sizes from each pair of pattern subsets (each containing N patterns) and construct one eCDF vector of all the emerging scalar values. This gives us a large amount of scalar data, which results in a smooth eCDF curve. Again, all the pairs yield $n(n-1)/2$ eCDF curves. This gives us another separate feature for the parameter estimation (see Fig. 10d). Naturally, the best results are obtained when all those features are combined. This case is shown in Fig. 10e. We mainly observe the minimal variability of the cost function.

4.5 The impact of σ

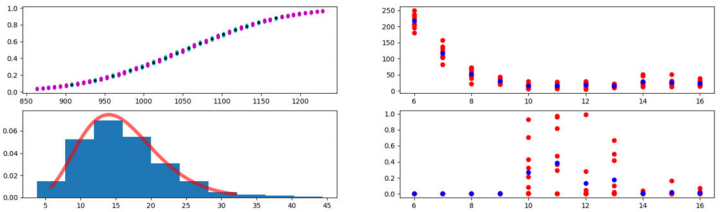
Finally, we discuss the impact of the parameter that we want to identify itself. To this end, we consider $\sigma \in \{3, 7, 11\}$ for our base setup of Sect. 3.3.3 in Fig. 11a–c. Our method always identifies σ correctly but becomes less confident as σ increases as the value for the correct σ decreases in the bottom right panels of (a) to (c). This effect stems from our CA's property: patterns obtained for larger σ become increasingly similar (cf. Fig. 3). This can be compensated to some extent by increasing the sample size, illustrated in Fig. 11d. However, as σ increases, it cannot be identified at some point since its precise value no longer influences the simulation. This stems from the spatial domain's periodicity; in this case, any particle can jump to any location independent of the precise value of σ. Thus, our method works for different σ, but the sample sizes needed to obtain confident parameter estimates vary depending on the actual values of σ. However, the limitation of identifying too large σ as compared to the domain size remains and cannot be cured by using more data since changes in σ do have no effect if σ is chosen such that all particles can jump anywhere in the domain. This effect is illuminated in Fig. 11's panel (e), demonstrating that our method correctly excludes the possibility of $\sigma < 12$ and correctly shows that large enough sigma values (here $\sigma = 15$) can not be identified (by any method) since other (large enough) σ produce the same patterns.



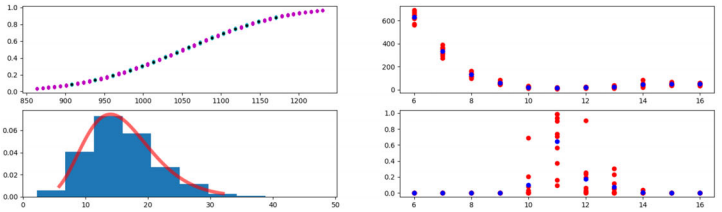
(a) $\sigma = 3$



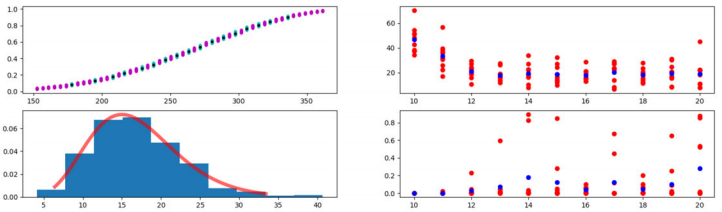
(b) $\sigma = 7$



(c) $\sigma = 11$



(d) $\sigma = 11$ with larger sample size



(e) $\sigma = 15$ for domain of size 25×25

Fig. 11 Construction of the statistical likelihood for multiple jump parameters σ and an increased sample size in **d**. Panel **e** demonstrates that σ cannot be identified if it is too large compared to the domain size. The layout of sub-figures is identical to Fig. 7 (colour figure online)

5 Conclusions

In this work, we introduced a parameter identification that works well for discrete problems, as in the context of CA models. Our parameter estimation method allowed us to identify model parameters using pattern data only, without knowing about initial states. We demonstrated the method's applicability by successfully identifying the value of jump parameter σ from a set of 4000 patterns. Moreover, we proved the robustness of our approach for different configurations of the model for the number of selected bins, the domain size, and the number of CA steps. Finally, we showed that the accuracy could be drastically improved if commonly used CA pattern data features were incorporated into our method's scheme.

Possible limitations of our approach stem from two main sources: CAs as base models and the statistical origin of the method. The main drawback of CAs is the loose physical motivation of their rules, making the validation of model predictions against experimental data more challenging. Especially when the complexity of the physical process being modeled by CA grows, the problem of ruling out the wrong model mechanisms becomes more challenging.

The second potential limitation of our approach stems from its statistical formulation. Indeed, a single evaluation of the statistical cost function $f_{s_{\text{data}}}(\boldsymbol{\sigma})$ requires computing repeated simulations of the underlying model. The numerical code we use in this work can very efficiently compute a single simulation of the CA. This means that the computational time needed for evaluating the N model grows linearly concerning the number of simulations. While it is not a problem for the situation where the cost of a single model run is low, it might become prohibitive for more complex CA models, which require more time to run. One possible solution to this problem could be the development of solvers optimized for batched simulations, as was done in Kazarnikov et al. (2020a, 2023b).

Future work may include applying the parameter estimation method to more than one parameter as already outlined in Haario et al. (2015) and combined with various characteristics/norms. Additionally, more sophisticated cellular automaton models may be used. This approach could address realistic problems, improve understanding of the underlying process, and draw conclusions relevant to real-life problems.

Acknowledgements We kindly acknowledge the fruitful discussions with Alexander Prechtel and Simon Zech on cellular automaton methods in soil science. A. Rupp has been supported by the Research Council of Finland's decision numbers 350101, 354489, 359633, 358944, and Business Finland's project number 539/31/2023. This research has also been supported by the German research foundation's research unit 2179 MAD Soil, and the DAAD PPP Finnland under grant number 57610378

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability All used datasets can be created using the codes in Rupp et al. (2022c, b), freely available software packages.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Armstrong, R., Mcclure, J., Robins, V., Liu, Z., Arns, C., Schlüter, S., & Berg, S. (2019). Porous media characterization using Minkowski functionals: Theories, applications and future directions. *Transport in Porous Media, 130*, 10. https://doi.org/10.1007/s11242-018-1201-4

Borovkova, S., Burton, R., & Dehling, H. (2001). Limit theorems for functionals of mixing processes with applications to u-statistics and dimension estimation. *Transactions of the AMS, 353*, 4261–4318.

Bozzini, B., Gambino, G., Lacitignola, D., Lupo, S., Sammartino, M., & Sgura, I. (2015). Weakly nonlinear analysis of turing patterns in a morphochemical model for metal growth. *Computers & Mathematics with Applications, 70*(8), 1948–1969. https://doi.org/10.1016/j.camwa.2015.08.019

Cooper,F.R., Baker, R.E., Bernabeu, M.O., Bordas, R., Bowler, L., Bueno-Orovio, A., Byrne, H.M., Carapella, V., Cardone-Noott, L., Cooper, J., Dutta, S., Evans, B.D., Fletcher, A.G., Grogan, J.A., Guo, W., Harvey, D.G., Hendrix, M., Kay, D., Kursawe, J., Maini, P.K., McMillan, B., Mirams, G.R., Osborne, J.M., Pathmanathan, P., Pitt-Francis, J. M., Robinson, M., Rodriguez, B., Spiteri, R.J., & Gavaghan, D.J. (2020) Chaste: Cancer, heart and soft tissue environment. *Journal of Open Source Software*, *5*(47):1848. https://doi.org/10.21105/joss.01848

Couce, E., & Knorr, W. (2010). Statistical parameter estimation for a cellular automata wildfire model based on satellite observations. *WIT Transactions on Ecology and the Environment, 137*, 47–55.

Donsker, M. D. (1952). Justification and extension of doob's heuristic approach to the Kolmogorov–Smirnov theorems. *Annals of Mathematical Statistics, 23*(2), 277–281. https://doi.org/10.1214/aoms/1177729445

Frittelli, M., Sgura, I., & Bozzini, B. (2024). Turing patterns in a 3d morpho-chemical bulk-surface reaction–diffusion system for battery modeling. *Mathematics in Engineering, 6*(2), 363–393. https://doi.org/10.3934/mine.2024015

Ghosh, P., Mukhopadhyay, A., Chanda, A., Mondal, P., Akhand, A., Mukherjee, S., Nayak, S. K., Ghosh, S., Mitra, D., Ghosh, T., et al. (2017). Application of cellular automata and markov-chain model in geospatial environmental modeling—A review. *Remote Sensing Applications: Society and Environment, 5*, 64–77.

Haario, H., Kalachev, L., & Hakkarainen, J. (2015) Generalized correlation integral vectors: A distance concept for chaotic dynamical systems. *Chaos*, *25*(6). https://doi.org/10.1063/1.4921939

Kazarnikov, A., & Haario, H. (2020). Statistical approach for parameter identification by Turing patterns. *Journal of Theoretical Biology, 501*, 110319. https://doi.org/10.1016/j.jtbi.2020.110319

Kazarnikov, A., Ray, N., Haario, H., Lappalainen, J., & Rupp, A. (2023) Parameter estimation for cellular automata. arXiv:2301.13320

Kazarnikov, A., Scheichl, R., Haario, H., & Marciniak-Czochra, A. (2023). A Bayesian approach to modeling biological pattern formation with limited data. *SIAM Journal on Scientific Computing, 45*(5), B673–B696. https://doi.org/10.1137/22M1485553

Ke, Y., Zhu, L., Wu, P., & Shi, L. (2022). Dynamics of a reaction–diffusion rumor propagation model with non-smooth control. *Applied Mathematics and Computation, 435*, 127478. https://doi.org/10.1016/j.amc.2022.127478

Lawless, A. S., Sgura, I., & Bozzini, B. (2019). Parameter estimation for a morphochemical reaction–diffusion model of electrochemical pattern formation. *Inverse Problems in Science and Engineering, 27*(5), 618–647. https://doi.org/10.1080/17415977.2018.1490278

Lengyel, I., & Epstein, I. R. (1991). Modeling of Turing structures in the chlorite-iodide-malonic acid-starch reaction system. *Science, 251*(4994), 650–652. https://doi.org/10.1126/science.251.4994.650

Lengyel, I., & Epstein, I. R. (1992). A chemical approach to designing turing patterns in reaction-diffusion systems. *Proceedings of the National Academy of Sciences, 89*(9), 3977–3979. https://doi.org/10.1073/pnas.89.9.3977

Li, X., Yang, Q. S., & Liu, X. P. (2007). Genetic algorithms for determining the parameters of cellular automata in urban simulation. *Science in China Series D: Earth Sciences, 50*(12), 1857–1866.

Li, B., & Zhu, L. (2024). Turing instability analysis and parameter identification based on optimal control and statistics method for a rumor propagation system. *Chaos: An Interdisciplinary Journal of Nonlinear Science, 34*(5), 053143. https://doi.org/10.1063/5.0207411

Moreira, J., & Deutsch, A. (2002). Cellular automaton models of tumor development: A critical review. *Advances in Complex Systems, 5*(02n03), 247–267.

Menshutina, N., Kolnoochenko, A., & Lebedev, E. (2020). Cellular automata in chemistry and chemical engineering. *Annual Review of Chemical and Biomolecular Engineering, 11*, 87–108.

Neumeyer, N. (2004). A central limit theorem for two-sample u-processes. *Statistics & Probability Letters, 67*(1), 73–85. https://doi.org/10.1016/j.spl.2002.12.001

Rupp, A., & Kanschat, G. (2021). HyperHDG: Hybrid discontinuous Galerkin methods for PDEs on hypergraphs. Published online. https://github.com/HyperHDG

Rupp, A., Guhra, T., Meier, A., Prechtel, A., Ritschel, T., Ray, N., & Totsche, K. (2019). Application of a cellular automaton method to model the structure formation in soils under saturated conditions: A mechanistic approach. *Frontiers in Environmental Science, 7*(170), 11. https://doi.org/10.3389/fenvs.2019.00170

Rupp, A., Gahn, M., & Kanschat, G. (2022). Partial differential equations on hypergraphs and networks of surfaces: Derivation and hybrid discretizations. *ESAIM: Mathematical Modelling and Numerical Analysis, 56*(2), 505–528. https://doi.org/10.1051/m2an/2022011

Rupp, A., Haario, H., & Kazarnikov, A. (2022) ECDF estimator: Python implementation for parameter estimation based on correlation integral likelihoods and empirical cumulative distribution functions. Published online. https://github.com/AndreasRupp/ecdf_estimator

Rupp, A., Zech, S., & Lappalainen, J. (2022) Cellular automaton: C++ implementation of a simple cellular automaton method. Published online. https://github.com/AndreasRupp/cellular-automaton

Ray, N., Rupp, A., & Prechtel, A. (2017). Discrete-continuum multiscale model for transport, biomass development and solid restructuring in porous media. *Advances in Water Resources, 107*, 393–404. https://doi.org/10.1016/j.advwatres.2017.04.001

Santé, I., García, A., Miranda, D., & Crecente, R. (2010). Cellular automata models for the simulation of real-world urban processes: A review and analysis. *Landscape and urban planning, 96*(2), 108–122.

Springer, S., Haario, H., Shemyakin, V., Kalachev, L., & Shchepakin, D. (2019). Robust parameter estimation of chaotic systems. *Inverse Problems & Imaging, 13*, 1189. https://doi.org/10.3934/ipi.2019053

Turing, A. M. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B, 237*(641), 37–72.

Tian, J., Zhu, C., Jiang, R., & Treiber, M. (2021). Review of the cellular automata models for reproducing synchronized traffic flow. *Transportmetrica A: Transport Science, 17*(4), 766–800.

Xiao, X., Wang, P., & Chou, K.-C. (2011). Cellular automata and its applications in protein bioinformatics. *Current Protein and Peptide Science, 12*(6), 508–519.

Yuan, T., Guan, G., Shen, S., & Zhu, L. (2023). Stability analysis and optimal control of epidemic-like transmission model with nonlinear inhibition mechanism and time delay in both homogeneous and heterogeneous networks. *Journal of Mathematical Analysis and Applications, 526*(1), 127273. https://doi.org/10.1016/j.jmaa.2023.127273

Yang, H., Wu, C., Li, H. W., & Fan, X. G. (2011). Review on cellular automata simulations of microstructure evolution during metal forming process: Grain coarsening, recrystallization and phase transformation. *Science China Technological Sciences, 54*(8), 2107–2118.

Yeh, A., & Xia, L. (2004). Integration of neural networks and cellular automata for urban planning. *Geospatial Information Science, 7*(1), 6–13. https://doi.org/10.1007/BF02826669

Zhu, L., & He, L. (2022). Pattern dynamics analysis and parameter identification of time delay-driven rumor propagation model based on complex networks. *Nonlinear Dynamics, 110*(2), 1935–1957. https://doi.org/10.1007/s11071-022-07717-8

Zhu, L., & He, L. (2022). Two different approaches for parameter identification in a spatial-temporal rumor propagation model based on turing patterns. *Communications in Nonlinear Science and Numerical Simulation, 107*, 106174. https://doi.org/10.1016/j.cnsns.2021.106174

Zech, S., Schweizer, S. A., Bucka, F. B., Ray, N., Kögel-Knabner, I., & Prechtel, A. (2022). Explicit spatial modeling at the pore scale unravels the interplay of soil organic carbon storage and structure dynamics. *Global Change Biology*. https://doi.org/10.1111/gcb.16230

Zhu, L., & Yuan, T. (2023). Optimal control and parameter identification of a reaction–diffusion network propagation model. *Nonlinear Dynamics, 111*(23), 21707–21733. https://doi.org/10.1007/s11071-023-08949-y

Zhu, L., Tao, X., & Shen, S. (2024). Pattern dynamics in a reaction-diffusion predator-prey model with allee effect based on network and non-network environments. *Engineering Applications of Artificial Intelligence, 128*, 107491. https://doi.org/10.1016/j.engappai.2023.107491