Machine Learning (2023) 112:3151–3190
https://doi.org/10.1007/s10994-023-06334-9

Learning system parameters from turing patterns

David Schnörr[1] · Christoph Schnörr[2]

Received: 19 August 2021 / Revised: 14 March 2023 / Accepted: 30 March 2023 /
Published online: 13 June 2023

Abstract

The Turing mechanism describes the emergence of spatial patterns due to spontaneous symmetry breaking in reaction–diffusion processes and underlies many developmental processes. Identifying Turing mechanisms in biological systems defines a challenging problem. This paper introduces an approach to the prediction of Turing parameter values from observed Turing patterns. The parameter values correspond to a parametrized system of reaction–diffusion equations that generate Turing patterns as steady state. The Gierer–Meinhardt model with four parameters is chosen as a case study. A novel invariant pattern representation based on resistance distance histograms is employed, along with Wasserstein kernels, in order to cope with the highly variable arrangement of local pattern structure that depends on the initial conditions which are assumed to be unknown. This enables us to compute physically plausible distances between patterns, to compute clusters of patterns and, above all, model parameter prediction based on training data that can be generated by numerical model evaluation with random initial data: for small training sets, classical state-of-the-art methods including operator-valued kernels outperform neural networks that are applied to raw pattern data, whereas for large training sets the latter are more accurate. A prominent property of our approach is that only a *single* pattern is required as input data for model parameter predicion. Excellent predictions are obtained for single parameter values and reasonably accurate results for jointly predicting all four parameter values.

Keywords Vector-valued parameter prediction · Turing patterns · Resistance distance histograms

Mathematics Subject Classification 68T07 · 62M30 · 35B36 · 92C15

Editor: Sicco Verwer.

David Schnörr and Christoph Schnörr have contributed equally to this work.

✉ Christoph Schnörr
schnoerr@math.uni-heidelberg.de

David Schnörr
schnoerr@gmail.com

[1] School of Life Sciences, Imperial College, London, UK

[2] Institute for Applied Mathematics, Heidelberg University, Im Neuenheimer Feld 205, 69120 Heidelberg, Germany

1 Introduction

1.1 Motivation and overview

Reaction–diffusion models in the form of Eq. (3) are used to describe the dynamic behaviour of interacting and diffusing particles in various disciplines including biochemical processes (Murray, 2001), ecology (Holmes et al., 1994), epidemiology (Martcheva, 2015) and tumor growth (Gatenby & Gawlinski, 1996). Here, we are interested in systems where the interaction of particles can give rise to spontaneous symmetry breaking of a homogenous system by means of the so-called *Turing mechanism* which was first described by Alan Turing in 1952 (Turing, 1952). It describes the scenario where a stable steady state of a non-spatial system of ordinary differential equations becomes unstable due to diffusion (Murray, 1982; Pertham, 2015). This phenomeon is hence also referred to as *diffusion-driven instability*. Such instabilities typically give rise to stable non-homogenous spatial patterns. In two spatial dimensions, for example, these patterns can take various forms such as spots, stripes or labyrinths (Murray, 2001). This variety of patterns is generated by a reaction–diffusion model that is parametrised by a few parameters that represent physical quantities of the system, such as reaction and diffusion rate constants. Certain versions of the well-known Gierer–Meinhardt model, for example, comprises four effective parameters (cf. Sect. 2.2) (Gierer & Meinhardt, 1972; Murray, 2001).

It was not until almost 4 decades after Alan Turing's seminal work that the first experimental observation of a Turing pattern was realised in a chemically engineered system (Castets et al., 1990). Recently, as a first practical application, a chemical Turing system was engineered to manufacture a porous filter that can be used in water purification (Tan et al., 2018). In biological systems, Turing patterns are regarded as the main driving mechanism in the formation of spatial structures in various biological systems, including patterning of palatel ridges and digits, hair follice distribution, feather formation, and patterns on the skins of animals such as fish and zebras (Economou et al., 2012; Jung et al., 1998; Nakamasu et al., 2009; Raspopovic et al., 2014; Sick et al., 2006). For recent surveys, see Landge et al. (2020) and Vittadello et al. (2021). However, the biological and mathematical complexity has often prevented identification of the precise molecular mechanisms and parameter values underlying biological systems. One difficulty in fitting models to experimental data stems from the high sensitivity of the arrangement of local structure on the initial conditions that are usually unknown in practice.

In this paper, we focus on the *inverse* problem: given a *single* non-homogenous spatial pattern and a reaction–diffusion model, the task is to predict the parameter values that *generated* the pattern as steady state of the reaction–diffusion equation. To this end, we introduce a novel pattern representation in terms of resistance distance histograms that effectively represents the spatial structure of patterns, irrespective of the local variability stemming from different initial conditions. This enables to compute almost invariant distances between patterns, to compute clusters of patterns and, above all, to predict model parameter values using a single pattern only as input data.

Specifically, we focus on the Gierer–Meinhardt model as a case study and apply and compare state-of-the-art kernel-based methods and neural network architectures for parameter prediction using the aforementioned resistance distance histograms. Additionally, neural networks are also applied to the raw pattern data for comparison. All predictors are trained using various parameter values that generate diffusion-driven instabilities and corresponding spatial patterns that result from solving numerically the reaction–diffusion equations.

1.2 Related work

The problem studied in this paper, parameter prediction from observed Turing patterns, has been studied in the literature from various angles. We distinguish three categories and briefly discuss few relevant papers.

- *Turing parameter estimation by optimal control.* The work (Garvie et al., 2010) presents an approach for estimating parameter values by fitting the solution of the reaction–diffusion equation to a given spatial pattern. This gives rise to a PDE-constraint problem of optimal control requiring sophisticated numerics; see also Stoll et al. (2016). A similar approach is studied in Sgura et al. (2019).
  The authors of Garvie et al. (2010) show and demonstrate that the proposed control problem is solvable which indicates that the task studied in our work, i.e. learning *directly* the pattern-to-parameter mapping, is not unrealistic. The approach of Garvie et al. (2010) has been generalized by Garvie and Trenchea (2014) in order to handle also non-constant spatially-distributed parameters. In our work, we only consider constant parameter values.
  A strong property of approaches employing PDE-based control is that they effectively cope with *noisy* observed patterns, provided that the type of noise is known such that a suitable objective function can be set up. A weak point is that in some of these papers the initial conditions are assumed to be known which is not the case in realistic applications, and that sophisticated and expensive numerics is required.
  The problem to control PDEs that generate *time-varying* travelling wave patterns has been studied recently (Karasözen et al., 2020; Shangerganesh & Sowndarrajan, 2020; Uzunca et al., 2017). In these studies the focus lies on fitting the trajectory of the evolving pattern in function space, however, rather than on estimating model parameter values that are assumed to be given.
- *Turing parameter estimation by statistical inference.* The paper Campillo-Funollet et al. (2019) presents a Bayesian approach to parameter estimation using the reaction–diffusion equation as forward mapping and a data likelihood function corresponding to additive Gaussian zero mean noise superimposed on observed patterns. Given a pattern, the posterior distribution on the parameter space is explored using expensive MCMC computations. A weak point of this approach shared with the works in the former category discussed above is that the initial conditions are assumed to be known. This assumption is not required in our approach presented below.
  Closer to our work is the recent paper Kazarnikov and Haario (2020). These authors also study model parameter identification from steady-state pattern data only, without access to initial conditions or the transient pattern evolving towards the steady state. The key idea is to model statistically steady-state patterns of ‘the same class’, i.e. collections of patterns whose spatial structure differs only due to varying initial conditions. This is achieved by adopting a Gaussian model for the empirical distribution of discretized L_2 distances between spatial patterns, which can be justified theoretically in the large sample limit. Regarding inference, this approach requires a few dozen to hun-

dreds of novel test patterns to estimate model parameters, unlike our approach which only requires a single pattern as input data.

In our work, we proceed differently: an almost invariant representation of patterns of 'the same class' is developed. This is advantageous in practice since parameter prediction can be done *directly whenever a novel pattern is obtained in an application.*

- *Turing parameter estimation: other approaches.* The work (Murphy et al., 2018) focuses on the identification of parameter values through a linear stability analysis on various irregular domains, assuming that the corresponding predicted pattern is close to a desired or observed pattern. However, the authors admit that, in many cases, the steady-state pattern may *not* be an eigenfunction of the Laplacian on the given domain, since the nonlinear terms play a role in the resultant steady-state pattern. Our approach does not rely on such assumptions.

  A recent account of the broad variety of Turing pattern generating mechanisms and corresponding identifiability issues is given by Woolley et al. (2021). In our work, we focus on the well-known Gierer–Meinhardt model and study the feasibility of predicting points in the four-dimensional parameter space based on given steady-state patterns.

1.3 Contribution and organisation

We introduce a novel representation of the spatial structure of Turing patterns which is achieved by computing *resistance* distances *within each* pattern, followed by discretization and using the empirical distribution of resistance distances as class representative. Discretization effects are accounted for by using the Wasserstein distance and a corresponding kernel function for comparing *pairs* of patterns. Based on this representation we present results of a feasibility regarding the prediction of model parameter values from observed patterns. To our knowledge, this is the first paper that applies machine learning methods to the problem of mapping directly Turing patterns to model parameter values of a corresponding system of reaction–diffusion equations that generate the pattern as steady state. Adopting the Gierer–Meinhardt model as a case study, we demonstrate that about 1000 data points suffice for highly accurate prediction of single model parameter values using state-of-the-art kernel-based methods. The accuracy decreases for predictions of all four model parameter values but is still sufficiently good in terms of the normalized root-mean-square error and the corresponding pattern variation. In the large data regime ($\geq$ 20.000 data points) predictions by neural networks trained directly on raw pattern data outperform kernel-based methods. Since these models can be trained on simulated data, this approach allows to infer parameters from single data points obtained e.g. by experimental measurements.

Our paper is organized as follows. Section 2 summarizes the basics of Turing patterns that are required in the remainder of the paper: definition of diffusion-driven patterns; discretization and a numerical algorithm for solving a system of semi-linear reaction–diffusion equations whose steady states correspond to the patterns that are used as input data for model parameter prediction; the Gierer–Meinhardt PDE and its parametrization. Section 3 details the features that are extracted from spatial patterns in order to predict model parameter values. A key feature are histograms of resistance distances that represent spatial pattern structure in a proper invariant way. Section 4 introduces four methods for model parameter prediction from observed patterns: two kernel-based methods (basic SVM regression and operator-valued kernels) and neural networks are applied to either nonlocal pattern features or to the raw pattern data directly. Numerical results are reported and discussed in Sect. 5. We conclude in Sect. 6.

1.4 Basic notation

We set $[n] = \{1, 2, \dots, n\}$ and $\mathbb{1}_n = (1, \dots, 1)^\top \in \mathbb{R}^n$ for $n \in \mathbb{N}$. The Euclidean inner product is denoted by $\langle p, q\rangle$ for vectors $q, p \in \mathbb{R}^n$ with corresponding norm $\|q\| = \sqrt{\langle q, q\rangle}$. The ℓ_n^∞-norm is denoted by $\|p\|_\infty = \max\{|p_i| : i \in [n]\}$. $\langle A, B\rangle = \operatorname{tr}(A^\top B)$ is the canonical inner product of matrices A, B with the operations trace $\operatorname{tr}(\cdot)$ and transposition $A^\top$ of A. The symbol λ with matrix argument denotes an eigenvalue $\lambda(A)$ of the matrix. The spectral matrix norm is defined as $\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)}$, where $\lambda_{\max}(A^\top A)$ is the largest eigenvalue of $A^\top A$. $\mathbb{R}^n_+$ is the nonnegative orthant and $u > 0$ means $u_1 > 0, \dots, u_n > 0$ if $u \in \mathbb{R}^n$. $\operatorname{Diag}(u)$ is the diagonal matrix that has the components of a vector u as entries. Similarly, $\operatorname{Diag}(A_1, \dots, A_n)$ is the block diagonal matrix with matrices A_i, $i \in [n]$ as entries. The probability simplex is denoted by $\Delta_n = \{p \in \mathbb{R}^n_+ : \langle \mathbb{1}_n, p\rangle = 1\}$.

2 Turing patterns: definition and computation

This section provides the required background on Turing patterns: reaction–diffusion systems (Sect. 2.1), concrete examples based on the Gierer–Meinhard model (Sect. 2.2), Turing instability and patterns (Sect. 2.3) and a numerical algorithm for computing Turing patterns (Sect. 2.4).

We refer to Murray (1982, 2001) for an analysis of the Gierer–Meinhard model devised by Gierer and Meinhardt (1972), to Hairer et al. (2008 Section II.7) regarding the numerical implicit Euler method, to Murray (2001) and Pertham (2015) for comprehensive expositions of spatial pattern formation in biology and to Kondo and Miura (2010), Landge et al. (2020) and Vittadello et al. (2021) for recent reviews.

2.1 Reaction–diffusion models

Consider a system of N interacting species described by the state vector $u(t) = (u_1(t), \dots, u_N(t))$, where $u_i(t) \in \mathbb{R}_+$ is the time-dependent concentration of the ith species. We assume that the dynamics is governed by an autonomous system of ordinary differential equations

$$\frac{d}{dt}u(t) = f(u(t)), \quad u(0) = u_0 > 0, \tag{1}$$

where initial condition $u_0 \in \mathbb{R}^N_+$ is assumed to be positive and $f : \mathbb{R}^N \to \mathbb{R}^N$ encodes interactions of the species. We further assume that the functions f_i, $i \in [N]$ are continuously differentiable with bounded derivatives. ‘Autonomous’ means that f does not explicitly depend on the time t.

Next, model (1) is extended to a spatial scenario including diffusion. Concentrations $u_i(t)$, $i \in [N]$ are replaced by space-dependent concentration fields

$$u(r,t) = (u_1(r,t), \dots, u_N(r,t)), \qquad r = (r_1, \dots, r_M) \in S \subset \mathbb{R}^M, \tag{2}$$

where $r \in S$ denotes a point in a region S of $\mathbb{R}^M$. The dynamics of these fields is described by a system of reaction–diffusion equations

$$\frac{\partial}{\partial t} u(r,t) = f(u(r,t)) + D\Delta_N u(r,t), \tag{3a}$$

$$u(r,0) = u_0(r), \quad r \in S, \quad t \geq 0, \tag{3b}$$

where

$$D = \operatorname{Diag}(\delta_1, \dots, \delta_N) \in \mathbb{R}^{N \times N} \tag{3c}$$

$$\Delta_N = \operatorname{Diag}(\Delta, \dots, \Delta) \tag{3d}$$

with diffusion constants $\delta_i \in \mathbb{R}_+$, $i \in [N]$ of species $i \in [N]$. Δ_N denotes the block-diagonal differential operator that separately applies the ordinary Laplacian $\Delta = \partial^2/\partial r_1^2 + \cdots + \partial^2/\partial r_M^2$ to each component function $r \mapsto u_i(r,t)$, $i \in [N]$.

System (3) has to be supplied with boundary conditions in order to be well-posed. A common choice are homogeneous Neumann conditions. We choose *periodic* boundary conditions, however, because this considerably speeds up the generation of training data by numerical simulation (Sect. 2.4), yet does *not* facilitate or change in any essential way the *learning problem* studied in this paper.

2.2 The Gierer–Meinhardt nodel

As concrete examples, we consider evaluations of the Gierer–Meinhardt model (Gierer & Meinhardt, 1972) comprising two species: a slowly diffusing activator that promotes its own and the other species' production, and a fast diffusing inhibitor that suppresses the production of the activator. Regarding the representation of the model by means of a PDE as in Eq. (3), several different variants have been proposed in the literature (Gierer & Meinhardt, 1972; Murray, 2001). Here, we use the dimensionless version analysed in Murray (1982) and defined by

$$f(u) = \begin{pmatrix} a - bu_1 + \frac{u_1^2}{u_2(1+cu_1^2)} \\ u_1^2 - u_2 \end{pmatrix}, \quad D = s\begin{pmatrix} 1 & 0 \\ 0 & \delta \end{pmatrix}, \tag{4}$$

with parameters $a, b, c, \delta, s > 0$ and the shorthands [cf. Eq. (3)]

$$u_1 = u_1(r,t), \qquad u_2 = u_2(r,t). \tag{5}$$

Since only the ratio between the diffusion constants of the two species effects the stability of the system, the diffusion constant of the first species in (4) is normalised and we set $\delta_1 = 1, \delta_2 = \delta$. The overall scaling of D determines however the wavelength of an emerging pattern, and we accordingly multiply the diffusion matrix D in Eq. (4) with an additional scaling factor $s > 0$.

Figure 1 displays the eigenvalues of the Jacobian of this model in the context of Turing instabilities as described in Sect. 2.3, for one choice of parameters a, b, c, δ, and

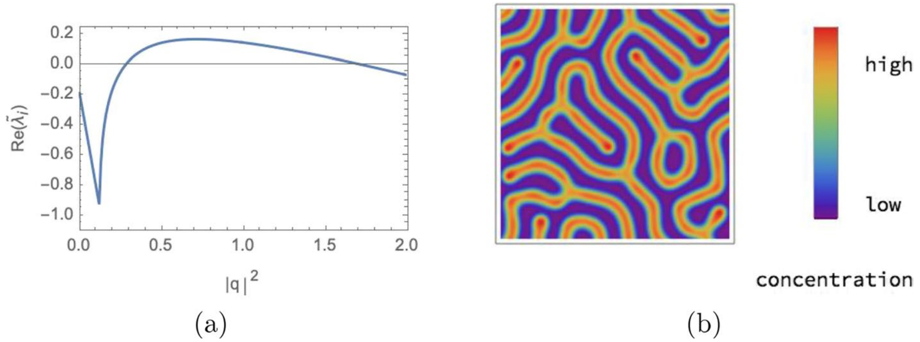


Fig. 1 Turing instability. Panel **a** displays the eigenvalue $\widetilde{\lambda}_i(q)$ of the Jacobian $\widetilde{J}(u^*, q)$ in Eq. (10) that gives rise to the Turing instability. The real part of $\widetilde{\lambda}_i(q)$ is shown, evaluated at an asymptotically stable equilibrium point u^* for the Gierer–Meinhardt model of Eq. (4), as a function of $\|q\|^2$. The parameters of the model were set to $a = 0.01$, $b = 1.2$, $c = 0.7$, $\delta = 40$ and the scaling parameter to $s = 1$. One has $\mathrm{Re}(\widetilde{\lambda}_i(q))|_{q=0} < 0$ due to an asymptotically stable equilibrium as explained after Eq. (8). We find that $\mathrm{Re}(\widetilde{\lambda}_i(q))$ becomes positive for an intermediate range of $\|q\|$ values, which indicates a Turing instability. **b** The first species field $u_1(r, t)$ of the solution to the system (3) computed on a 128×128 grid, as described in Sect. 2.4

the corresponding Turing pattern emerging when simulating the model numerically as described in Sect. 2.4. Figure 2 (page 10) shows simulation results for various parameter values c and δ. The model gives rise to different types of patterns, ranging from spots to labyrinths. The characteristic length scale, or 'wavelength', of the pattern varies with these parameters. This limits the ranges of parameter values that we analyse in the experiments: a too small wavelength leads to numerical artefacts when simulating

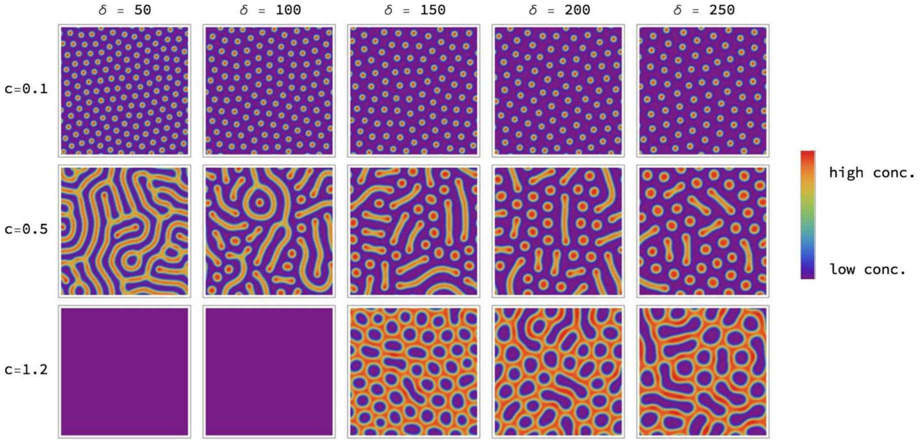


Fig. 2 Pattern generation by model evaluation. Simulation results of species u_1 in the Gierer–Meinhardt model defined by Eq. (4) on a 128×128 grid with final time $T = 5000$ for varying parameters c and δ and fixed parameters $a = 0.02$, $b = 1.0$ and $s = 0.5$. We observe that different parameter combinations give rise to different types of patterns and differing wavelengths. For $c = 1.2$ and $\delta = 50$ and $\delta = 100$ we find a homogeneous solution and no pattern, which illustrates that the system does not exhibit an Turing instability for these parameter values

the corresponding PDEs due to discretization errors; a too large wavelength on the other hand only yields a small section of the spatial pattern as 'close-up view'.

We hence only consider parameter values that exclude both extreme cases relative to a fixed grid graph that was used for numerical computation.

2.3 Turing patterns

We characterize *Turing instabilities* that cause *Turing patterns*. Suppose $u^* \in \mathbb{R}^N_+$ is an equilibrium point of (1) satisfying $f(u^*) = 0$. In order to assess the stability of u^*, we write

$$u(t) = u^* + \epsilon \widetilde{u}(t), \quad \epsilon > 0, \quad \widetilde{u}(t) \in \mathbb{R}^N \tag{6}$$

and compute a first-order expansion of the system (1),

$$\frac{d}{dt}\widetilde{u}(t) = J(u^*)\widetilde{u}(t) + \mathcal{O}(\epsilon), \tag{7}$$

with the Jacobian

$$J(u^*) = \big(J_{i,j}(u^*)\big)_{i,j\in[N]}, \qquad J_{i,j}(u^*) = \frac{\partial f_i(u)}{\partial u_j}\bigg|_{u=u^*} \tag{8}$$

Let $\lambda_1 = \lambda_1(J(u^*)), \dots, \lambda_N = \lambda_N(J(u^*)) \in \mathbb{C}$ be the eigenvalues of J at u^*. The equilibrium u^* is *asymptotically stable* if and only if $\mathrm{Re}(\lambda_i) < 0, i \in [N]$ (Schaeffer & Cain, 2016, Cor. 6.1.2), that is a region of attraction $U(u^*)$ containing u^* exists such that $u(t) \in U(u^*)$ implies $u(t) \to u^*$ as $t \to \infty$.

Assuming that u^* is asymptotically stable, we next consider the extended system (3) that involves spatial diffusion. Let $u^* = u^*(r)$ denote the spatially constant extension of the equilibrium point that neither depends on the time t nor on the spatial variable r: $u^*(r,t) = u^*(r) = u^*(r'),\ \forall r, r' \in S$. Hence $\Delta_N u^* = 0$. Due to the diffusion terms, this equilibrium of f may not be stable anymore for the system (3), however. To assess the stability of u^*, a linear stability analysis is conducted using the ansatz

$$u(r,t) = u^* + \epsilon \widetilde{u}(t) e^{i\langle q,r\rangle}, \tag{9}$$

where $i = \sqrt{-1}$, $\epsilon \in \mathbb{R}_+$, $\widetilde{u}(t) \in \mathbb{R}^N$, $q \in \mathbb{R}^M$. The perturbation $\widetilde{u}(t)e^{i\langle q,r\rangle}$ conforms to the eigenvalue problem of the *linearized* spatial system (3). Substituting this ansatz into (3) and expanding to the first order with respect to ϵ yields a linear system of equations for $\widetilde{u}(t)$, similar to Eq. (7), but with Jacobian $\widetilde{J}$ given by

$$\widetilde{J}(u^*, q) = J(u^*) - \|q\|^2 D. \tag{10}$$

Let $\widetilde{\lambda}_1(q) = \lambda(\widetilde{J}(u^*, q)), \dots, \widetilde{\lambda}_N(q) = \lambda(\widetilde{J}(u^*, q))$ be the eigenvalues of $\widetilde{J}(u^*, q)$. For $\|q\| = 0$, we have $\widetilde{\lambda}_i(0) = \lambda_i$, $i \in [N]$ [eigenvalues of $J(u^*)$ given by (8)] and hence $\mathrm{Re}(\widetilde{\lambda}_i(0)) < 0$, $i \in [N]$, since $u^* = u^*(r)$ is equal to the equilibrium of non-spatial system (1) for every r.

We say u^* is a *Turing instability* of the system (3) if there exists a finite $\|q\| > 0$ and some $i \in [N]$ for which $\mathrm{Re}(\widetilde{\lambda}_i(q)) > 0$, i.e. the steady state u^* becomes unstable for a certain wavevector q. Here, we are interested in the additional condition $\mathrm{Re}(\widetilde{\lambda}_j(q)) < 0$ for $\|q\| \to \infty$ for *all* $j \in [N]$, such that $\mathrm{Re}(\widetilde{\lambda}_i(q))$ has a global maximum for some finite $\|q\|$.

This type of instability typically leads to a *stable* pattern of a wavelength corresponding to q Murray (2001). In summary, the conditions for a Turing instability read

$$\mathrm{Re}(\widetilde{\lambda}_j(0)) = \mathrm{Re}(\lambda_j) < 0, \ \text{for all}\ j \in [N], \tag{11a}$$

$$\text{there exist}\ \|q\| > 0 \ \text{and}\ i \in [N] \ \text{for which}\ \mathrm{Re}(\widetilde{\lambda}_i(q)) > 0, \tag{11b}$$

$$\mathrm{Re}(\widetilde{\lambda}_j(q)) < 0 \ \text{if}\ \|q\| \to \infty, \ \text{for all}\ j \in [N]. \tag{11c}$$

For other types of instabilities, we refer the reader to Scholes et al. (2019). Figure 1 shows $\mathrm{Re}(\widetilde{\lambda}_i(q))$ of a two-species system ($N = 2$) with Turing instability and the Turing pattern resulting from solving numerically the system of equations (3).

2.4 Numerical simulation

This section describes the numerical algorithm used to simulate the system of PDEs (3).

2.4.1 Discretization

We consider reaction–diffusion systems of the form (3) with two spatial dimensions $M = 2$, spatial points and domain

$$r = (r_1, r_2) \in S = [0, n_r] \times [0, n_r], \quad n_r \in \mathbb{N}, \tag{12}$$

and their solutions within the time interval $t \in [0, T]$. We further assume doubly periodic boundary conditions, i.e., $u_i(0, r_2, t) = u_i(n_r, r_2, t)$ for each $r_2 \in [0, n_r]$ and $u_i(r_1, 0, t) = u_i(r_1, n_r, t)$ for each $r_1 \in [0, n_r]$, $t \in [0, T]$ and $i \in [N]$. The domain S is discretized into a regular torus grid graph $G = (V, E)$ of size

$$m = |V| = n_r \times n_r, \tag{13}$$

where each node $v \in V$ indexes a point $r_v \in S$. The edge set E represents the adjacency of each node to its four nearest neighbors on the grid and takes into account the doubly periodic boundary conditions. Each of these edges have length 1 corresponding to uniform sampling along each coordinate r_1 and r_2, respectively, of size 1.

2.4.2 Algorithm

Case $N = 1$. For simplicity, consider first the case of a single species $N = 1$, $u(r, t) = u_1(r, t)$, with diffusion constant δ, and let $v(t)$ denote the vector obtained by stacking the rows of the two-dimensional array of function values $\big(u(r_v, t)\big)_{v \in V}$ evaluated on the grid. We discretize time into equal intervals of length $h > 0$ and write $v^{(k)} = v(kh)$ for $k \in \mathbb{N}$. The discretized PDE of the single species case

$$\frac{\partial}{\partial t} u(r, t) = f\big(u(r, t)\big) + d\Delta u(r, t), \qquad u(r, 0) = u_0(r) \tag{14}$$

of Eq. (3) is solved by the implicit Euler scheme (Hairer et al., 2008, Section II.7)

$$\frac{v^{(k+1)} - v^{(k)}}{h} = f\left(v^{(k+1)}\right) + \delta L v^{(k+1)}, \tag{15}$$

where matrix L is the Laplacian discretized using the standard 5-point stencil. To perform a single time-step update according to (15), we rewrite this equation as fixed point iteration with an inner iteration index l

$$v^{(k_l)} = (I - h\delta L)^{-1}\left(v^{(k)} + hf\left(v^{(k_{l-1})}\right)\right), \quad l = 1, 2, \dots, \quad v^{(k_0)} = v^{(k)}, \tag{16}$$

where I is the identity matrix. This fixed point equation is iterated until the convergence criterion

$$\frac{\|v^{(k_l)} - v^{(k_{l-1})}\|}{\|v^{(k_{l-1})}\|} \leq \varepsilon_l \tag{17}$$

is met for some constant $0 < \varepsilon_l \ll 1$, followed by updating the outer iteration (16)

$$v^{(k+1)} = v^{(k_l)}. \tag{18}$$

The outer iteration is terminated when

$$\left\|f\left(v^{(k+1)}\right) + \delta L v^{(k+1)}\right\|_\infty \leq \varepsilon_k \tag{19}$$

for some constant $0 < \varepsilon_k \ll 1$.

Due to the doubly periodic boundary conditions, the matrix $I - h\delta L = W\Lambda W^*$ is a sparse, *block-circulant* and can hence be diagonalized using the unitary Fourier matrix W corresponding to the two-dimensional discrete Fourier transform of doubly periodic signals defined on the graph G. As a result, using the fast Fourier transform (2D-FFT), multiplication of the inverse matrix by some vector b,

$$(I - h\delta L)^{-1} b = W\Lambda^{-1} W^* b, \tag{20}$$

can be efficiently computed due to the convolution theorem (Bracewell, 2000) by

- computing the 2D-FFT of b: $\widehat{b} = W^* b$,
- pointwise multiplication with the inverse eigenvalues of the matrix: $\Lambda^{-1}\hat{b}$, where Λ^{-1} is a diagonal matrix and hence is inverted *elementwise*,
- transforming back using the inverse 2D-FFT: $W(\Lambda^{-1}\widehat{b})$.

The eigenvalues Λ of the matrix $I + h\delta L$ result from applying the 2D-FFT to the block-circulant matrix corresponding to the convolution stencil

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + h\delta \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}. \tag{21}$$

Case $N > 1$. This procedure applies almost unchanged to the case of *multiple species* ($N > 1$), because the diffusion operator $D\Delta_N$ of (3) is block-diagonal. It suffices to check the case $N = 2$: $v(t) = \binom{v_1(t)}{v_2(t)}$ denotes the stacked subvectors v_1, v_2 that result from stacking the rows of the two-dimensional arrays of function values $\left(u_1(r_v, t)\right)_{v\in V}, \left(u_2(r_v, t)\right)_{v\in V}$ evaluated on the grid. The fixed point iteration (16) then reads

$$\begin{pmatrix} v_1^{(k_l)} \\ v_2^{(k_l)} \end{pmatrix} = \begin{pmatrix} (I - h\delta_1 L)^{-1} & 0 \\ 0 & (I - h\delta_2 L)^{-1} \end{pmatrix} \left(\begin{pmatrix} v_1^{(k)} \\ v_2^{(k)} \end{pmatrix} + h \begin{pmatrix} f_1\left(v_1^{(k_{l-1})}, \; v_2^{(k_{l-1})}\right) \\ f_2\left(v_1^{(k_{l-1})}, \; v_2^{(k_{l-1})}\right) \end{pmatrix} \right), \tag{22}$$

where the mappings $(I - h\delta_i L)^{-1}$, $i = 1, 2$ can be applied in parallel to the corresponding subvectors. Note that the vector $f(v^{(k_{l-1})}) = \binom{f_1(v^{(k_{l-1})})}{f_2(v^{(k_{l-1})})}$ *couples* the species concentrations. The general case $N > 2$ is handled similarly.

2.4.3 Step size selection

Step size h has to be selected such that two conditions are fulfilled: Matrix $I - h\delta L_N$ should be invertible where δL_N means the block-diagonal matrix

$$dL_N = \operatorname{Diag}(d_1 L, \dots, d_N L), \tag{23}$$

and the fixed point iteration (16) should converge. We discuss these two conditions in turn.

The first condition holds if $I - h\delta_i L$ is invertible for every $i \in [N]$, which certainly holds if $\lambda_{\min}(I - h\delta_i L) > 0$ which yields

$$h < \frac{1}{\max\{\delta_i\}_{i\in[N]}\lambda_{\max}(L)}. \tag{24}$$

This also yields the estimate

$$\|(I - h\delta L_N)^{-1}\|_2 = \frac{1}{\lambda_{\min}(I - h\delta L_N)} \le \frac{1}{1 - h\max\{\delta_i\}_{i\in[N]}\lambda_{\max}(L)}. \tag{25}$$

$\lambda_{\max}(L)$ may be easily computed beforehand using the power method (Horn & Johnson, 2013, p. 81) or replaced by the upper bound due to Gerschgorin's circle theorem (Horn & Johnson, 2013, Section 6.1).

Now consider the fixed point iteration (16). Due to our assumptions stated after Eq. (1), the mapping f is Lipschitz continuous, i.e. there exists a constant $L_f > 0$ such that

$$\|f(v) - f(v')\| \le L_f\|v - v'\|, \qquad \forall v, v'. \tag{26}$$

Thus, writing $T_h(v) = (I - h\delta L_N)^{-1}\left(v^{(k)} - hf(v)\right)$, we obtain using (25) and (26)

$$\|T_h(v) - T_h(v')\| \le \frac{hL_f}{1 - h\max\{\delta_i\}_{i\in[N]}\lambda_{\max}(L)}\|v - v'\|, \quad \forall v, v'. \tag{27}$$

As a result, both above-mentioned conditions hold if h is chosen small enough to satisfy (24) and

$$\frac{hL_f}{1 - h\max\{\delta_i\}_{i\in[N]}\lambda_{\max}(L)} < 1. \tag{28}$$

Then the mapping T_h is a contraction and, by Banach's fixed point theorem (Pathak, 2018, Section 5.1), the iteration converges.

3 Extracting features from turing patterns

We extract two types of features from Turing patterns: *resistance distance histograms* (Sect. 3.1) efficiently encode the spatial structure of patterns due to their stability under spatial transformations. This almost invariant representation also includes few symmetries, however, which may reduce the accuracy of parameter prediction in certain scenarios. Hence *two additional features* are extracted that remove some of these symmetries (Sect. 3.2).

3.1 Resistance distance histograms (RDHs)

Resistance distance histograms (see Definition 1 below) require two standard preprocessing steps described subsequently: representing Turing patterns as weighted graphs and computing pairwise resistance distances.

3.1.1 Graph-based representation of Turing patterns

Let $u_{i,j}, i,j \in [n_r]$ be the concentration values of some species of a reaction–diffusion system at time $t = T$ on a regular torus grid graph $G = (V, E)$ of size $m = |V| = n_r \times n_r$, where each node $v \in V$ indexes a point $r_v \in S = [0, n_r] \times [0, n_r]$ (cf. Sect. 2.4.1). Let $u_v = u_{i,j}$ be the concentration value at $v = (i,j)$, obtained by simulating a system of PDEs (3) as described in Sect. 2.4, and denote by

$$\overline{u} = \frac{1}{m} \sum_{v \in V} u_v \tag{29}$$

the mean concentration. We assign weights $\omega_{vv'}$ to edges $(v, v') \in E$ between adjacent nodes $v, v' \in V$ by

$$\omega_{vv'} = \begin{cases} 1, & \text{if } (u_v \geq \overline{u} \text{ and } u_{v'} \geq \overline{u}) \text{ or } (u_v < \overline{u} \text{ and } u_{v'} < \overline{u}), \\ \epsilon, & \text{otherwise, where } 0 < \epsilon \ll 1, \end{cases} \tag{30}$$

that is edges between adjacent nodes receive the unit weight 1 if both concentrations are either larger or smaller than the mean concentration $\overline{u}$, and the weight ϵ otherwise.

3.1.2 Resistance distances and histograms

Based on (30), we define the weighted adjacency matrix Ω_G and graph Laplacian L_G of G,

$$\Omega_G = (\omega_{vv'})_{v,v' \in V}, \tag{31}$$

$$D_G = \operatorname{Diag}(\Omega_G \mathbb{1}_m), \tag{32}$$

$$L_G = D_G - \Omega_G, \tag{33}$$

where $\mathbb{1}_m$ is an m-dimensional column vector with all entries equal to 1. Using (33), in turn, we define the *Gram* or *kernel matrix* K

$$K = (J_m + L_G)^{-1} \in \mathbb{R}^{m\times m}, \qquad J_m = \mathbb{1}_m \mathbb{1}_m^T \in \mathbb{R}^{m\times m} \tag{34}$$

and the *resistance matrix* $R \in \mathbb{R}^{m\times m}$

$$R = (R_{vv'})_{v,v'\in V}, \qquad R_{vv'} = K_{vv} + K_{v'v'} - 2K_{v'v}, \qquad v, v' \in V. \tag{35}$$

Each entry $R_{vv'}$ is the *resistance distance* between v and v' that was introduced in Klein and Randić (1993). Its name refers to relations of the theory of electric networks (Doyle & Snell, 1984), (Bapat, 2014, Chapter 10), (Brémaud, 2017, Chapter 8). A *geometric interpretation* results from the relation

$$R_{vv'} \le d_G(v, v'), \tag{36}$$

where d_G denotes the length of the shortest weighted path connecting v and v' in G. The bound is tight if this path is unique. Conversely, if multiple paths connect v and v', then the resistance distance is strictly smaller than $d_G(v, v')$. This sensitivity to the connectivity between nodes in graphs explains its widespread use, e.g. for cluster and community detection in graphs (Fortunato, 2010).

A *probabilistic interpretation* of the resistance distance is as follows. Consider a random walk on G performing jumps along the edges in discrete time steps, and assume that the probability to jump along an edge is proportional to the edge's weight. Then $R_{vv'}$ is inversely proportional to the probability that the random walk starting at v visits v' before returning to v (Bapat, 2014, Section 10.3). In view of (30), this implies that the process jumps more likely between neighbouring nodes with both large (small) concentrations than between differing concentrations.

We add a third interpretation of the resistance distance from the viewpoint of kernel methods (Hofmann et al., 2008; Seto et al., 2014) and *data embedding*. Let

$$\mathcal{F}_G = \{f : V \to \mathbb{R}\} \cong \mathbb{R}^m \tag{37}$$

the space of functions on V that we identify with real vectors of dimension $m = |V|$, and consider the bilinear form

$$\mathcal{E} : \mathcal{F}_G \times \mathcal{F}_G \to \mathbb{R}, \qquad \mathcal{E}(f, g) = \frac{1}{2} \sum_{v,v'\in V} \omega_{vv'}(f_v - f_{v'})(g_v - g_{v'}) \tag{38a}$$

$$= \langle f, L_G g\rangle. \tag{38b}$$

Since G is connected, the symmetric and positive semidefinite graph Laplacian L_G has a single eigenvalue 0 corresponding to the eigenvector $\mathbb{1}_m$. Consequently, using $\mathcal{E}$, one defines the Hilbert space

$$\mathcal{H}_G = (\mathcal{F}_G, \langle\cdot,\cdot\rangle_G) \tag{39a}$$

with inner product

$$\langle f, g\rangle_G = \Big(\sum_{v\in V} f_v\Big)\Big(\sum_{v'\in V} g_{v'}\Big) + \mathcal{E}(f, g) = \big\langle f, (J_m + L_G) g\big\rangle. \tag{39b}$$

Since $\dim \mathcal{H}_G < \infty$, all norms are equivalent and the evaluation map $f \mapsto f_v$ is continuous. Hence $\mathcal{H}_G$ is a *reproducing kernel Hilbert space* (Paulsen & Raghupathi, 2016, Def. 1.1) with reproducing kernel

$$K : V \to V \to \mathcal{H}_G, \qquad K(v, v') = K_{vv'} = \langle K^v, K^{v'} \rangle_G, \quad v, v' \in V, \tag{40}$$

where $K_{vv'}$ denotes the entries of the Gram matrix (34), and $K^v, K^{v'}$ are the column vectors indexed by v, v' and interpreted as elements (functions) in $\mathcal{H}_G$. The resistance distance (35) then takes the form

$$R_{vv'} = \|K^v - K^{v'}\|_G^2, \qquad v, v' \in V, \tag{41}$$

where $\|\cdot\|_G$ denotes the norm induced by the inner product (39b). This makes explicit the *nonlocal* nature of the resistance distance $R_{vv'}$ between nodes $v, v' \in V$ in terms of the squared distance of the corresponding functions $K^v, K^{v'}$ in $\mathcal{H}_G$.

Overall, each of the three interpretations reveals how the resistance distance measures *nonlocal spatial structure* of Turing patterns. The second column of Fig. 3 visualises the resistance distances $\{R_{vv'}\}_{v' \in V}$ with respect to a single fixed node v. We

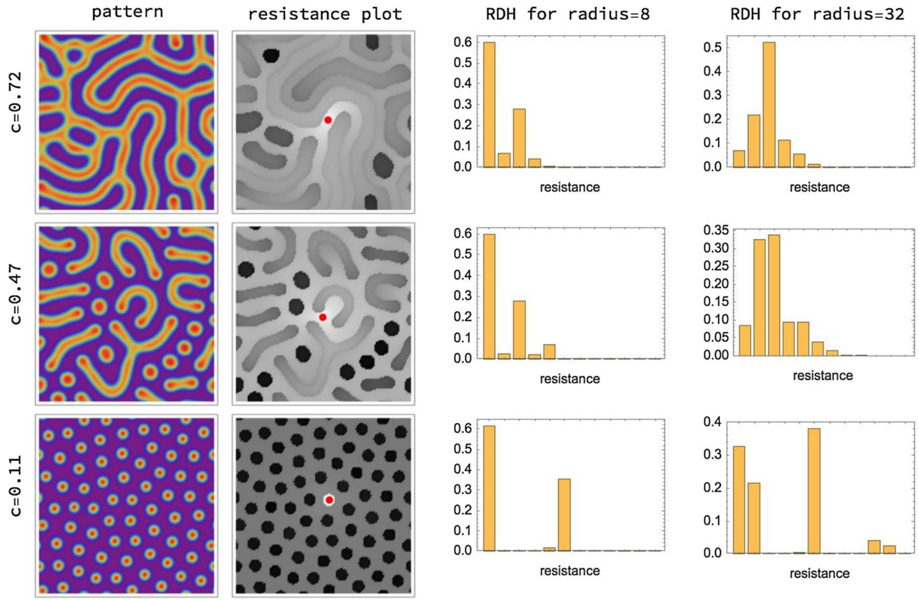


Fig. 3 Resistance distances and histograms (RDHs). Each row of the figure shows from left to right: a pattern, a resistance distance plot and the resistance distance histograms (RDH) for radii 8 and 32 of the first species, obtained from simulating the Gierer–Meinhardt model in Eq. (4) on a 128×128 grid. Rows correspond to different values of parameter c. Weights are assigned to the edges of the corresponding grid graph according to Eq. (30). The resistance plots visualise resistance distances between all nodes and one central node marked with red. These plots result from partitioning the column of the resistance matrix R (35) corresponding to the central node into an 128×128 array. The darker the colour of a node the larger its resistance towards the central node. One can observe how the resistance values vary depending on the local structure of the pattern. In particular, the RDHs (cf. Definition 1) differ substantially for the different types of patterns. The parameters are set to $a = 0.02$, $b = 1$, $\delta = 100$ and $s = 0.8$ for all three columns, and the parameter c is set to 0.72, 0.47 and 0.11 for the three columns, respectively. The final simulation time is $T = 5000$. The RDHs are computed for $B = 12$ bins and hypergraph spacing of $t = 1$

condense these data extracted from Turing patterns into features, in terms of corresponding histograms described next.

Definition 1 (*resistance distance histogram (RDH)*) Let $V_t \subset V,\ t \in \mathbb{N}$ denote the nodes of the subgraph induced by the subgrid that is obtained from the underlying grid graph $G = (V, E)$ by undersampling each coordinate direction with a factor t. Define the set of resistance distance values

$$\mathcal{R}_{t,r} = \{R_{vv'} : v \in V_t,\ v' \in V,\ \|r_v - r_{v'}\| \leq r\} \tag{42}$$

parametrized by t and a radius parameter $r \in \mathbb{R}$. The *resistance distance histogram (RDH)* $H_{r,t} \in \Delta_B$ is the normalized histogram of the resistance distance values $\mathcal{R}_{t,r}$ with respect to a uniform binning with bin number B of the interval $[0, R_{\max}]$, with a suitably chosen maximal value $R_{\max}$.

The radius parameter r specifies the spatial scale at which the local structure of Turing patterns is assessed through RDHs, in a way that is stable against spatial transformations of the local domains corresponding to (42). r is the only essential parameter, since RDHs are based on data $\mathcal{R}_{r,t}$ collected from nodes $v \in V_t$. This results in averaging of local pattern structure and makes RDHs only weakly dependent on the spacing t. In addition, the representation becomes robust against local noise in the pattern caused by random initial conditions. This is illustrated in Fig. 4 where patterns and corresponding RDHs are shown for fixed parameters and varying initial conditions. The influence of the grid size on RDHs (cf. Sect. 2.4.1) is visualized in Fig. 5. Figure 6 demonstrates the smooth dependence of RDHs on kinetic model parameters.

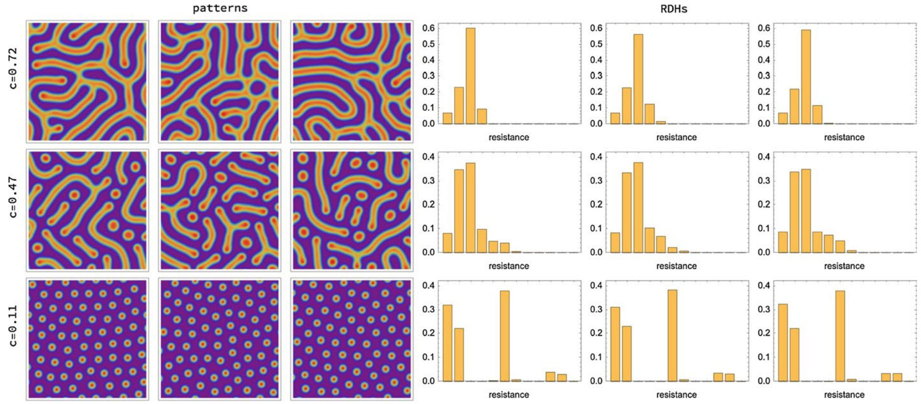


Fig. 4 Influence of initial conditions. Parameters and simulation details are the same as in Fig. 3. The left side shows simulated patterns for three different initial conditions and for different c values, with all other parameters fixed. The right side shows the corresponding resistance distance histograms (RDHs) for radius 32. We find that while the patterns for each value of c vary substantially, the corresponding RDHs are very similar to each other, in particular when applying the Wasserstein kernel for measuring similarity (Sect. 4.2.3). On the other hand substantial differences can be observed between RDHs for different values of c. This demonstrates the robustness ('quasi-invariance') of RDHs with respect to local variations in the patterns due to noise in initial conditions

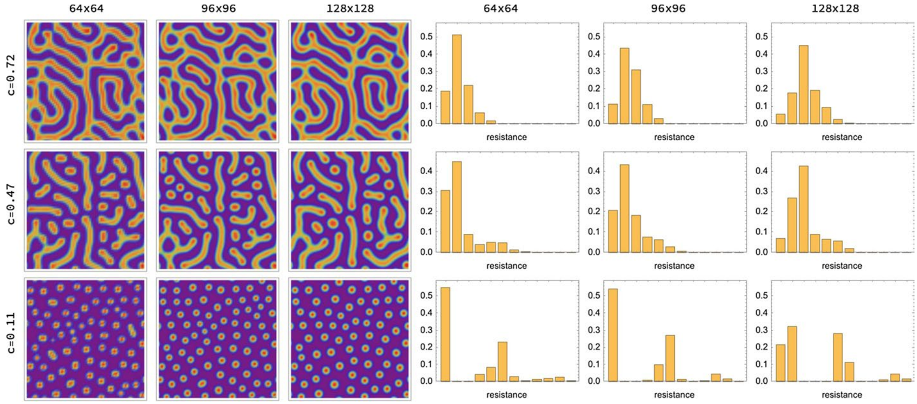


Fig. 5 Spatial resolution. Parameters and simulation details are the same as in Figs. 3 and 4 except that we here varied the discretization grid (cf. Sect. 2.4.1) and the radius parameter r [cf. (42)]. The left side shows simulated patterns for 64×64, 96×96 and 128×128 grids and radii $r = 16, 24$ and 32, respectively. These radii values ensure that resistance values are collected over the same physical distances for the different grid sizes. The with of grid cells has to be scaled accordingly to ensure that distances in the underlying graph, which upper bound resistance distances by (36), are comparable. The right side shows the corresponding resistance distance histograms (RDHs) which up to binning effects do not change significantly in each row. These plots demonstrate that up to such unavoidable binning effects which can be properly taking into account using Wasserstein kernels (Sect. 4.2.3), relatively coarse spatial resolutions already suffice to properly represent Turing patterns by RDHs

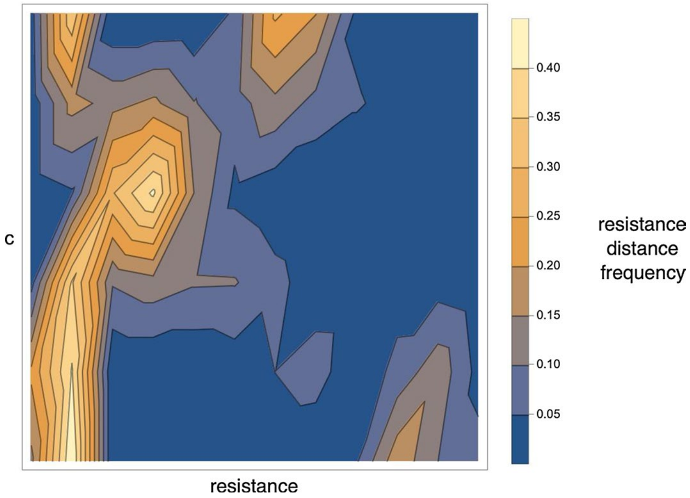


Fig. 6 Smooth parameter dependency of resistance distance histograms. Parameters and simulation details are the same as in Fig. 3 but we use a 64×64 grid here and a radius $r = 32$ and scaling $s = 0.2$. The model parameter value c is varied on the range [0.06, 1.06] and the corresponding patterns are simulated and RDHs computed. The figure shows these RDH values as a function of c in a contour density plot. Up to some minor fluctuations stemming from the noise in initial conditions, we find that the RDHs vary smoothly with varying c, demonstrating the systematic relation between RDHs and kinetic model parameters

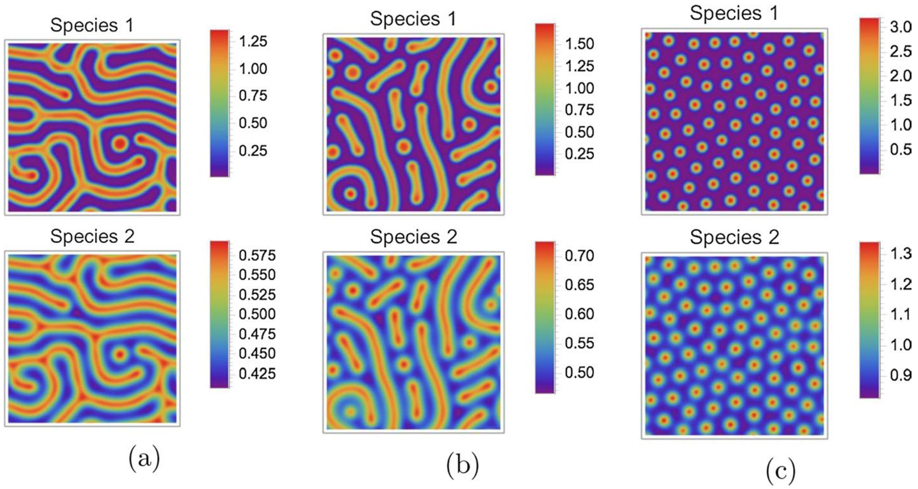


Fig. 7 Redundancy of two species. The patterns depicted by Fig. 3 are shown again, here for *both* species, however. **a**–**c** correspond to the three rows of Fig. 3. It is apparent that the patterns of the two species are qualitatively very similar: Basically, they are just rescaled and shifted versions of each other. Since resistance distance histograms (RDHs) are invariant under such operations, the resulting RDHs would be approximately equal. It is hence sufficient to use only the patterns of one species for computing RDHs and subsequent analysis

Remark 1 Typically, concentrations of different species in a Turing pattern are approximately scaled (and sometimes reflected) versions of each other, in particular for two-species systems like the Gierer–Meinhardt model studied here. See Fig. 7 for an illustration. The RDHs defined in Definition 1 hence contain redundant information when computed for different species. Therefore, we only use concentrations of one species to compute RDHs in the following.

3.2 Maximal concentration and connected components

RDHs according to Definition 1 represent the spatial structure of Turing patterns in a compact way. However, this representation also includes few symmetries such that certain properties of patterns are not captured, such as

1. *absolute concentration values* rescaling or shifting the concentration values of a pattern does not change the RDHs;
2. *range-reflection symmetry* reflection of a pattern on any plane of constant concentration, i.e. inverting the total order that defines the weights (30), does not change the RDHs.

To account for these two properties, we introduce the following two additional features.

1. *Maximal concentration* c_m We aim to estimate the concentration of areas in the pattern with large concentrations while disregarding local fluctuations that potentially arise from numerical inaccuracies. To this end, we bin the concentration values of a pattern into a histogram and define the maximal concentration c_m as the location of the right-most peak.

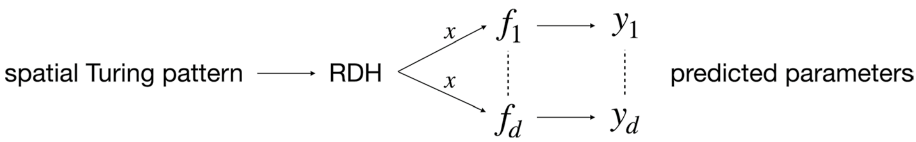


Fig. 8 The approach described in Sect. 4.2.1: resistance histograms (RDHs) x are extracted from spatial Turing patterns based on which support vector machines f_j, $j \in [d]$ separately predict the parameter values y_j, $j \in [d]$ of the underlying Gierer–Meinhard model

2. *Number of connected components* n_c To account for the above-mentioned range reflection symmetry, we define the graph $G' = (V, E')$ with the same nodes V as the original graph G but with only a subset of edges $E' \subset E$ between nodes of high concentration. We then compute the number n_c of connected components in G'.

This list of pattern properties not captured by RDHs is not exhaustive, of course. For example, RDHs do not effectively measure the steepness of transitions between areas of high to low concentrations. However, since RDHs turned out to be powerful enough for parameter estimation, as shown below, we did not use further features in this study.

4 Learning parameters from spatial turing patterns

This section concerns the problem of learning the kinetic parameters of Turing patterns described in Sect. 2.3. We start by formulating the learning problem in Sect. 4.1. Sections 4.2 and 4.3 introduce the approaches for parameter prediction studied in this paper, kernel-based predictors and neural networks, respectively.

The first simple kernel-based approach to parameter prediction is illustrated by Fig. 8. Each predictor f_j, $j \in [d]$ has the standard form (51) and is trained to predict a corresponding parameter y_j, $j \in [d]$ of the underlying Gierer–Meinhard model using the resistance distance histogram (RDH) x that is computed for a given spatial Turing pattern beforehand (Sect. 3.1). Section 4.2.1 describes how the predictors f_j are trained using a basic setup (Sect. 4.1.1). Suitable kernels for Hilbert space embeddings of RDHs are presented in Sect. 4.2.3.

Figure 9 illustrates a more ambitious kernel-based approach for *jointly* predicting model parameter values from Turing patterns. In addition to a kernel for RDHs in the input space, an output space kernel is used in order to also capture dependencies among the output variables. The predictor mapping has the factorized form (61) which parametrizes the

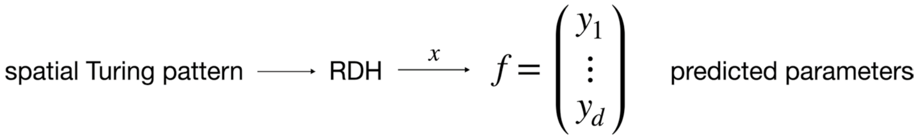


Fig. 9 The approach described in Sect. 4.2.2: resistance histograms (RDHs) x are extracted from spatial Turing patterns based on which an operator-valued kernel approach is employed in order to jointly predict the model parameter values y_j, $j \in [d]$. This approach not only aims to capture dependencies between input data and output variables but also the dependencies among the output variables themselves

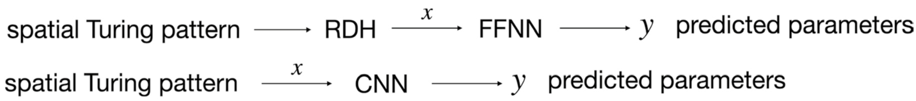


Fig. 10 The two approaches described in Sect. 4.3: a feedforward neural network (FFNN) predicts model parameter values based on RDHs (top row) whereas a CNN does the same including feature extraction from spatial Turing patterns (bottom row)

operator-valued kernel approach to parameter prediction. Accordingly, both training and inference are numerically more expensive than the basic SVM approach from Sect. 4.2.1.

Finally, two elementary neural network approaches for model parameter prediction are illustrated by Fig. 10 and described in Sect. 4.3. The first approach replaces the embeddings intro reproducing kernel Hilbert spaces (RKHS) of the above-mentioned approaches by a feedforward neural network. The second approach additionally ignores the resistance distance histograms as features and applies a convolutional neural network (CNN) directly to spatial Turing patterns for predicting model parameters.

4.1 Setup

4.1.1 Learning problem

We consider a multi-output learning problem with training set

$$\mathcal{D}_n = \{(x_i, y_i)\}_{i\in[n]} \subset \mathcal{X}\times\mathcal{Y}, \qquad n\in\mathbb{N}, \quad \mathcal{X}=\{H_{r,t}\}, \quad \mathcal{Y}=\mathbb{R}^d, \tag{43}$$

where each x_i is a resistance distance histogram $H_{r,t}$ (RDH) according to Definition 1, and vectors y_i comprise parameter values of a model, such as the parameters a, b, c, δ of the Gierer–Meinhardt model (4). Our goal is to learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that generalises well to points $(x, y) \notin \mathcal{D}_n$.

We distinguish *individual* parameter prediction (Sect. 4.2.1) corresponding to dimension $d = 1$, where for *each* model parameter a predictor function f is *separately* learned, and *joint* parameter prediction (Sect. 4.2.2) corresponding to $d > 1$, where a *single vector-valued* predictor function f is learned. In each of these cases, the output training data y_i in $\mathcal{D}_n$ (43) have to be interpreted accordingly.

Regarding individual parameter prediction, we employ basic support vector regression (Evgeniou et al., 2000; Smola & Schölkopf, 2004) in Sect. 4.2.1 and specify suitable kernel functions for resistance histograms as input data in Sect. 4.2.3. Regarding joint parameter prediction, we employ regression using an operator-valued kernel function in Sect. 4.2.2. For background reading concerning reproducing kernel Hilbert spaces (RKHS) and their use in machine learning, we refer to Berg et al. (1984), Berlinet and Thomas-Agnan (2004), Paulsen and Raghupathi (2016), Evgeniou et al. (2000), Cucker and Smale (2001) and Hofmann et al. (2008), respectively. We also utilize neural networks in Sect. 4.3 for both individual and joint parameter prediction.

4.1.2 Accuracy measure

A commonly used measure for the accuracy of an estimator $f : \mathcal{X} \to \mathcal{Y}$ on a test set $X \times Y = \{(x_i, y_i)\}_{i \in [m]}, m \in \mathbb{N}, X \subset \mathcal{X} \subset \mathbb{R}^k, Y \subset \mathcal{Y} = \mathbb{R}^d$ is the *root-mean-square error* (RMSE) defined as

$$\mathrm{RMSE} = \left(\frac{1}{md} \sum_{i \in [m]} \|f(x_i) - y_i\|^2 \right)^{1/2} = \left(\frac{1}{md} \sum_{i \in [m]} \sum_{j \in [d]} ((f_j(x_i) - y_{i,j})^2 \right)^{1/2}. \tag{44}$$

Additional normalisation by the empirical mean value of the nonnegative target variables yields the *normalised root-mean-square error* (NRSME)

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{\frac{1}{md} \sum_{i \in [m]} \sum_{j \in [d]} y_{i,j}}. \tag{45}$$

We use the NRMSE to measure the accuracy of predicted parameter values in this work. Figure 12 (page 34) illustrates visually the variation of patterns for various NRMSE values.

We consider as "good" model parameter predictions with accuracy values NRMSE ≤ 0.2 and as "excellent" predictions with accuracy values NRMSE ≤ 0.05.

4.2 Kernel-based parameter prediction

4.2.1 Individual parameter prediction using SVMs

In this section, we focus on the case $d = 1$ where along with a finite sample of RDHs x_i, values $y_i \in \mathbb{R}$ of some model parameter are given as training set (43). *Individual* parameter prediction means that, for each model parameter, an individual prediction function

$$f : \mathcal{X} \to \mathbb{R} \tag{46}$$

specific to this particular parameter is determined. We apply standard support vector regression.

Given a symmetric and nonnegative kernel function (see Sect. 4.2.3 for concrete examples)

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+, \tag{47}$$

the corresponding *reproducing kernel Hilbert space* (RKHS) with inner product $\langle \cdot, \cdot \rangle_k$ is denoted by $\mathcal{H}_k$. 'Reproducing' refers to the

$$k_x = k(x, \cdot) \in \mathcal{H}_k, \quad \forall x \in \mathcal{X}, \tag{48a}$$

$$f(x) = \langle f, k_x \rangle_k, \qquad \forall x \in \mathcal{X}, \quad \forall f \in \mathcal{H}_k. \tag{48b}$$

Using the training set $\mathcal{D}_n$ and a corresponding loss function, our objective is to determine a prediction function $f \in \mathcal{H}_k$ that maps a RDH x extracted from an observed pattern to a corresponding parameter value $y = f(x)$. We employ the ε-insensitive loss function (Evgeniou et al., 2000)

$$\ell_\varepsilon : \mathbb{R} \to \mathbb{R}_+, \qquad z \mapsto \ell_\varepsilon(z) = \max\{0, |z| - \varepsilon\}, \qquad \varepsilon \geq 0, \tag{49}$$

to define the training objective function

$$\sum_{i\in[n]} \ell_\varepsilon\big(y_i - f(x_i)\big) + \frac{\lambda}{2}\|f\|_k^2, \tag{50}$$

where the regularizing parameter controls the size of the set of prediction functions f in order to avoid overfitting. Since ℓ_ε is continuous and the regularizing term monotonically increases with $\|f\|_k$, the representer theorem (Whaba, 1990; Schölkopf et al., 2001) applies and implies that the function f^* minimizing (50) lies in the span of the functions generated by the training set

$$f^* \in \operatorname{Span}\{k_{x_1}, \dots, k_{x_n}\}, \qquad f^* = \sum_{i\in[n]} \alpha_i^* k_{x_i}. \tag{51}$$

Using nonnegative slack variables ξ_i, $i \in [n]$ in order to represent the piecewise linear summands $\ell_\varepsilon\big(y_i - f(x_i)\big)$ of (50) by

$$\min_{\mathbb{R}} \xi_i \qquad \text{subject to} \qquad \begin{cases} \xi_i \geq 0, \\ \xi_i \geq y_i - f(x_i) - \varepsilon, \\ \xi_i \geq f(x_i) - y_i - \varepsilon, \end{cases} \tag{52}$$

and substituting $f = \sum_{i\in[n]} \alpha_i k_{x_i}$ yields the training objective function (50) in the form

$$\min_{\alpha,\xi} \Big\{ \sum_i \xi_i + \frac{\lambda}{2}\langle \alpha, K_n \alpha\rangle \Big\}, \quad \lambda > 0, \qquad K_n = \big(k(x_i, x_j)\big)_{i,j\in[n]} \tag{53a}$$

$$\text{subject to} \qquad \xi_i \geq 0, \tag{53b}$$

$$\xi_i + \sum_j \alpha_j k(x_j, x_i) \geq y_i - \varepsilon, \tag{53c}$$

$$\xi_i - \sum_j \alpha_j k(x_j, x_i) \geq -y_i - \varepsilon, \qquad i \in [n]. \tag{53d}$$

Since K_n is positive definite, this is a convex quadratic program that can be solved using standard methods. Substituting the minimizing vector α^* into (51) determines the desired prediction function f^*.

This procedure is repeated to obtain prediction functions f_j^*, $j \in [d]$ for each parameter to be predicted, using the training sets $\{(x_i, y_{i,j})\}_{i\in[n]}$ for $j \in [d]$.

4.2.2 Joint parameter prediction using operator-valued kernels

In this section, we consider the general case $d \geq 2$: each vector y_i of the training set (43) comprises the values of a fixed set of model parameters and x_i is the RDH extracted from the corresponding training pattern. Our aim is to exploit dependencies between input variables $\{x_i\}_{i\in[n]}$ and output variables $\{y_i\}_{i\in[n]}$ as well as dependencies between the output variables. To this end, we use the method proposed by Kadri et al. (2013) for vector-valued parameter prediction that generalizes vector-valued kernel ridge regression as introduced by Evgeniou et al. (2005) and Micchelli and Pontil (2005). See Álvarez et al. (2012) for a review of vector-valued kernels and Brouard et al. (2016) and Minh et al. (2016) for general operator-valued kernels and prediction.

In addition to a kernel function k (47) for the input data, we additionally employ a kernel function

$$l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+, \qquad \mathcal{Y} = \mathbb{R}^d_+ \tag{54}$$

for the output data with corresponding RKHS $(\mathcal{H}_l, \langle\cdot,\cdot\rangle_l)$ and properties analogous to (48), and a operator-valued kernel function

$$K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{H}_l) \tag{55}$$

that takes values in the space $\mathcal{L}(\mathcal{H}_l)$ of bounded self-adjoint operators from $\mathcal{H}_l$ to $\mathcal{H}_l$. K enjoys the same properties as the more common scalar-valued kernel functions k, l, viz., it is the reproducing kernel of a RKHS $(\mathcal{H}_K, \langle\cdot,\cdot\rangle_K)$ of $\mathcal{H}_l$-valued functions

$$g : \mathcal{X} \to \mathcal{H}_l. \tag{56}$$

The properties analogous to (48) now read

$$K(x,\cdot)\varphi \in \mathcal{H}_K, \quad \forall x \in \mathcal{X},\ \forall \varphi \in \mathcal{H}_l, \tag{57a}$$

$$\langle g, K(x,\cdot)\varphi\rangle_K = \langle g(x), \varphi\rangle_l, \quad \forall g \in \mathcal{H}_K,\ \forall x \in \mathcal{X},\ \forall \varphi \in \mathcal{H}_l. \tag{57b}$$

In particular, the operator-valued kernel matrix

$$K^o_n = \big(K(x_i, x_j)\big)_{i,j\in[n]} \tag{58}$$

is positive definite, due to the positive definiteness of the kernel function K,

$$\sum_{i,j\in[m]} \langle \varphi_i, K(x_i, x_j)\varphi_j\rangle_l \geq 0 \tag{59}$$

for all $m \in \mathbb{N}$, $x_1, \dots, x_m \in \mathcal{X}$, $\varphi_1, \dots, \varphi_m \in \mathcal{H}_l$.

In order to capture dependencies among the output variables as well as between input and output variables, the prediction function

$$f : \mathcal{X} \to \mathcal{Y} \tag{60}$$

is not learned directly, unlike the individual predictors (46) in the preceding section [cf. (51)]. Rather, the optimal mapping (60) is parametrized by

$$x \mapsto f^*(x) = \phi_l^{-1} \circ g^*(x), \tag{61}$$

where $\phi_l : \mathcal{Y} \to \mathcal{H}_l$ is the feature map corresponding to the output kernel function (54) (see, e.g., Cucker & Smale, 2001, Section 3) satisfying

$$\langle \phi(y_i), \phi(y_j)\rangle_l = l(y_i, y_j), \tag{62}$$

and $g^* \in \mathcal{H}_k$ is determined by regularized least-squares on the transformed training data $(x_i, \phi_l(y_i))_{i\in[n]}$, that is by solving

$$g^* = \arg\min_{g\in\mathcal{H}_K} \left\{ \sum_{i\in[n]} \|g(x_i) - \phi_l(y_i)\|_l^2 + \lambda\|g\|_K^2 \right\}, \quad \lambda > 0. \tag{63}$$

Invoking again the representer theorem valid for the present more general scenario (Micchelli & Pontil, 2005), g^* admits the representation

$$g^* = \sum_{i\in[n]} K_{x_i}\psi_i^*, \quad \psi_i^* \in \mathcal{H}_l, \quad K_{x_i} = K(x_i, \cdot), \tag{64}$$

which makes explicit how the approach generalizes the individual parameter predictors (51).

It remains to specify a kernel function K and the computation of ϕ_l^{-1} in order to evaluate the prediction map (61). As for K, our choice is

$$K(x_i, x_j) = k(x_i, x_j)C_{\mathcal{YY}|\mathcal{X}} \tag{65a}$$

with the input kernel function k (47) and the conditional covariance operator

$$C_{\mathcal{YY}|\mathcal{X}} = C_{\mathcal{YY}} - C_{\mathcal{YX}}C_{\mathcal{XX}}^{-1}C_{\mathcal{XY}} \tag{65b}$$

on $\mathcal{H}_l$. Since K is evaluated on the training data, $C_{\mathcal{YY}|\mathcal{X}}$ is replaced in practice by evaluating the empirical covariance operators on the right-hand side, i.e.

$$C_{n;\mathcal{YY}} = \frac{1}{n}\sum_{i\in[n]} l_{y_i} \otimes l_{y_i}, \tag{66}$$

with output kernel function l (54), $l_{y_i} = l(y_i, \cdot)$ and $(l_{y_i} \otimes l_{y_j})\varphi = \langle l_{y_j}, \varphi\rangle_l l_{y_i}$, and similarly for the remaining mappings on the right-hand side of (65b). The kernel function (65) together with the predictor (64) in the *output feature space* reveals how the dependencies are taken into account of both the output variables and between the input and output variables.

In order to obtain for some test input (RDH) x the predicted parameter vector

$$\hat{y} = f^*(x) = \phi_l^{-1} \circ g^*(x) \tag{67}$$

from the predicted *embedded* output value $g^*(x)$, the mapping ϕ_l^{-1} of (61) has to be evaluated. This is an instance of the so-called *pre-image problem* (Honeine & Richard, 2011; Schölkopf et al., 1999). In the present scenario, putting together (63), (64), (65) and (57), this yields after a lengthy computation (Kadri et al., 2013, Appendix) the optimization problem

$$\hat{y} = \arg\min_{y\in\mathcal{Y}} \left\{ l(y, y) - 2l_y^\top v(x;\mathcal{D}_n) \right\} \tag{68a}$$

where

$$v(x;\mathcal{D}_n) = \left(k_x^\top \otimes T_n\right)\left(K_n \otimes T_n + n\lambda I_{n^2}\right)^{-1}\mathrm{vec}(I_n), \quad \lambda > 0, \tag{68b}$$

$$T_n = L_n - (K_n + n\varepsilon I_n)^{-1}K_nL_n, \qquad 0 < \varepsilon \ll 1, \tag{68c}$$

$$K_n = \big(k(x_i,x_j)\big)_{i,j\in[n]}, \qquad k_x = \big(k(x,x_1),\dots,k(x,x_n)\big)^\top, \tag{68d}$$

$$L_n = \big(l(y_i,y_j)\big)_{i,j\in[n]}, \qquad l_y = \big(l(y,y_1,\dots,l(y,y_n)\big)^\top. \tag{68e}$$

Here, λ in (68b) is the regularization parameter of (63), ε in (68c) is a small constant regularizing the numerical matrix inversion, K_n, L_n are the input and output kernel matrices corresponding to the training data (43), x is a novel unseen test pattern represented as described in Sect. 3, and y is the parameter vector variable to be optimized.

Unlike the input kernel function k that is applied to RDHs (see Sect. 4.2.3), the output kernel function l applies to the common case of parameter vectors and hence choosing the smooth Gaussian kernel function as l is a sensible choice. Therefore, once the vector $v(x;\mathcal{D}_n)$ has been computed for a test pattern x, the optimization problem (68a) can be solved numerically by iterative gradient descent with adaptive step size selection by line search.

Regarding the computation of the vector (68b) that defines the objective function of (68a), the matrix T_n given by (68c) can be directly computed for numbers n up to few thousands data points using off-the-shelf solvers. This is not the case for the linear system of (68b) involving the Kronecker product $K_n \otimes T_n$, however, which is dense and has the size $n^2 \times n^2$. Therefore, we solve the linear system

$$\big(K_n \otimes T_n + n\lambda I_{n^2}\big)u = \mathrm{vec}(I_n) \tag{69}$$

in a memory-efficient way using the global-GMRES algorithm proposed by Bouhamidi and Jbilou (2008) that iteratively constructs Krylov matrix subspaces and approximates the solution by solving a sequence of low-dimensional least-squares problems. Having computed u, the vector (68b) results from computing

$$v(x;\mathcal{D}_n) = \mathrm{vec}(T_n \mathrm{vec}^{-1}(u) k_x). \tag{70}$$

4.2.3 Kernels for resistance distance histograms

In this section, we specify kernel functions (47) that we evaluated for parameter prediction. Below, $x, x' \in \mathcal{H}_{r,t}$ denote two RDHs.

- *Symmetric χ^2 -kernel.* This kernel is member of a family of kernels generated by Hilbertian metrics on the space of probability measures on $\mathcal{X}$ (Hein & Bousquet, 2005) and defined by

$$k_\gamma(x,x') = \sum_{i\in[B]} \frac{x_i x'_i}{x_i + x'_i}. \tag{71}$$

- *Exponential χ^2 -kernel.* The exponential χ^2-kernel reads

$$k_\gamma(x,x') = \exp\left(-\frac{1}{\gamma}\sum_{i\in[B]} \frac{(x_i - x'_i)^2}{x_i + x'_i}\right), \quad \gamma > 0. \tag{72}$$

- *Wasserstein kernel.* We define a cost matrix

$$C = (C_{i,j})_{i,j \in [B]}, \qquad C_{i,j} = (i-j)^2, \qquad i,j \in [B] \tag{73}$$

and the squared discrete Wasserstein distance between x and x'

$$d_W^2(x,x') = \langle C, M^* \rangle, \tag{74}$$

where M^* solves the discrete optimal transport problem (Peyré & Cuturi, 2019)

$$\min_M \langle C, M \rangle \quad \text{subject to} \quad M \geq 0, \quad M\mathbb{1}_n = x, \quad M^\top \mathbb{1}_n = x'. \tag{75}$$

M is a doubly stochastic matrix, and the minimizer M^* is the optimal transport plan for transporting x to x' with respect to the given costs C. The Wasserstein kernel is defined as

$$k_W(x,x') = \exp\left(-\frac{1}{\gamma} d_W^2(x,x')\right), \quad \gamma > 0, \tag{76}$$

and can be shown to be a valid kernel for generating a RKHS and embedding (Bachoc et al., 2018). For measures defined on the real line $\mathbb{R}$, it is well known that the distance d_W between two distributions can be evaluated in terms of the corresponding cumulative distributions. This carries over to discrete measures x, x' and the distance $d_W(x,x')$ considered here, provided the implementation takes care of monotonicity and hence invertibility of the discrete cumulative distributions; we refer to Santambrogio (2015, Section 2) for details.

4.3 Neural networks

4.3.1 Feedforward neural networks

Let n_k and n_d be the dimensions of the input and output space, respectively. A feedforward neural network (FFNN) of depth $L \in \mathbb{N}$ is a function $f : \mathbb{R}^{n_k} \to \mathbb{R}^{n_d}$ that can be written as the composition $f(x) = f^{(o)}(f^{(L)}(\dots f^{(1)}(x)))$ of L hidden layers $f^{(j)} : \mathbb{R}^{n_i^{(j)}} \to \mathbb{R}^{n_o^{(j)}}, j \in [L]$, where $n_i^{(1)} = n_k$, and a final output layer $f^{(o)} : \mathbb{R}^{n_o^{(L)}} \to \mathbb{R}^{n_d}$. Each hidden layer is in turn the composition of a linear transformation and an activation function. We use the rectified linear unit (ReLU) as activation function defined as

$$\mathrm{ReLU}(x) = \max(0,x), \quad x \in \mathbb{R}. \tag{77}$$

The hidden layers can accordingly be written as

$$f^{(j)}(x) = \mathrm{ReLU}\big(W^{(j)}x + b^{(j)}\big), \quad j \in [L], \tag{78}$$

for an input vector $x \in \mathbb{R}^{n_i^{(j)}}$, weight matrix $W^{(j)} \in \mathbb{R}^{n_o^{(j)} \times n_i^{(j)}}$ and bias $b^{(j)} \in \mathbb{R}^{n_o^{(j)}}$. The ReLU function in Eq. (78) acts independently on each element of its argument. For the final output function $f^{(o)}$ we use a linear transformation without activation function:

$$f^{(o)}(x) = W^{(o)}x + b^{(o)}, \tag{79}$$

with $W^{(o)} \in \mathbb{R}^{n_d} \times \mathbb{R}^{n_o^{(L)}}$ and bias $b^{(o)} \in \mathbb{R}^{n_d}$. The values $n_o^{(j)}$ are called the numbers of "hidden units" or "neurons" of the jth layer. Since for a general $W^{(j)}$ all neurons of the $(j-1)$th layer are connected to all neurons of the jth layer, hidden layers as in Eq. (78) are also called "fully-connected layers". To characterise a FFNN we specify the numbers of hidden units as $(n_o^{(1)}, \dots, n_o^{(L)})$. For example, (10, 20, 5) denotes a FFNN of depth $L = 3$ with $n_o^{(1)} = 10, n_o^{(2)} = 20$ and $n_o^{(3)} = 5$, respectively. Accordingly, () denotes a FFNN without any hidden layers, i.e., $L = 0$.

4.3.2 Convolutional neural networks

Convolutional neural networks (CNNs) have been used for learning problems on image data in various different applications, in particular for classification tasks (Gu et al., 2018). We will consider CNNs that were trained on raw pattern data to learn the kinetic parameters of the model, as a benchmark for the results obtained from training models on resistance distance histograms.

Various different CNN architectures have been used in the literature. The majority consist of three basic types of layers: convolutional, pooling, and fully connected layers. The convolutional layer's function is to learn feature representations of the inputs. This is achieved by convolving the inputs with learnable kernels, followed by applying an element-wise activation function. Convolutional layers are typically followed by pooling layers which reduce the dimension by combining the outputs of clusters of neurons into a single neuron in the next layer. Local pooling combines small clusters, typically of size 2×2, while global pooling acts on all neurons of the previous layer. A sequence of convolutional and pooling layers is then typically followed by one or several fully-connected layers as in Eq. (78). These are then followed by a final output layer chosen according to the specific learning task such as a softmax layer for classification tasks (Gu et al., 2018).

For the applications in this paper, we found the best performance for minimalistic CNNs consisting of only one convolutional and one fully-connected layer. The ReLU activation function in Eq. (77) was applied to the output of both layers. We denote the architecture by $(n_k/n_p/n_f)$ where n_k is the used number of kernels of size $n_p \times n_p$ and n_f denotes the number of neurons in the fully connected layer.

5 Experiments and discussion

5.1 implementation details

5.1.1 Simulation details

According to Sect. 2.4.3, setting the step size h properly requires to estimate (an upper bound of) the Lipschitz constant of f. It turned out, however, that applying standard calculus (Rockafellar & Wets, 2009, Ch. 9) to the concrete mappings f (4) yields too loose upper bounds of L_f and hence quite small step sizes h, which slows down the numerical computations unnecessarily. Therefore, in practice, we set h to a value that is 'reasonable' for the backward Euler method and monitored the fixed point iteration (16) in order to check every few iterations if the method diverges, in which case h was replaced by $h/2$. We found $h = 0.2$ to be a reasonable choice for all applications studied here.

The threshold ε_l for the convergence criterion of the inner iteration in Eq. (17) was set to $\varepsilon_l = 0.001$. The outer iteration was terminated if either the convergence criterion in Eq. (19) was met with threshold $\varepsilon_k = 10^{-6}$, which we checked after time intervals of $\delta t = 100$, or when a fixed maximal time T_f was reached. We chose $T_f = 2000$ for domain sizes of 32×32 and 64×64, and $T_f = 5000$ for a domain size of 128×128. We found that the patterns do not change substantially beyond these time values even if the convergence criterion in Eq. (19) was not satisfied.

5.1.2 Initial conditions

For simulating the Gierer–Meinhardt model described in Sect. 2 we need to specify initial conditions for both species. Given a spatial discretisation of size $n_r \times n_r$ and given parameters a, b and c, we first find the equilibrium point u^* of the according non-spatial system given in Eq. (1). For each species and each spatial grid point we then sample a random number in the interval $[0.9 \times u_i^*, 1.1 \times u_i^*]$, where u_i^* is the equilibrium value of the respective species. This choice of initial conditions leads to faster convergence of PDE simulations while giving rise to large spatial variations of computed patterns as illustrated by Fig. 4.

5.1.3 Colour scaling of pattern plots

Since the analysed patterns vary substantially in their absolute concentration values, the colour scaling of patterns in figures is *not* normalised between different patterns. Some systems can possess a Turing instabilty for certain parameter values but the resulting pattern can have a vanishingly small amplitude making it irrelevant for real applications (Scholes et al., 2019). The patterns in this work all have a non-trivial amplitude, with 99% of patterns having an amplitude larger than 50% of the mean value of the pattern. Here, we define the amplitude as the difference between the largest and smallest concentration value in the pattern.

5.1.4 Resistance distance histograms

As pointed out in Remark 1, the resistance distance histograms (RDHs) for different species are typically redundant. We hence used only the first species' simulation results for computing the RDHs from simulations of the Gierer–Meinhardt model in Eq. (4) studied here.

For computing RDHs, we had to specify the edge weight parameter ϵ of (30) which penalises paths from high to low concentrations and vice versa. Choosing ϵ too small (corresponding to a large penalty) leads to a saturation effect of large resistance values between nodes at large distances, preventing to resolve the geometry of a pattern on such larger scales. Similarly, a large ϵ fails to resolve the geometry on small scales. We empirically found $\epsilon = 0.003$ to be a good compromise.

For the parameter t in Definition 1 determining the undersampling of the graph, we found $t = 1$ to give the most accurate results. Thus, all results presented in this study were produced using $t = 1$. Note that $t = 1$ means that the original graph was used without undersampling.

When simulating a model for varying parameters, we found that some patterns led to few occurences of very large resistance values, while the majority of patterns had maximal resistance values substantially below these outliers. We believe that these large values arise from numerical inaccuracies in the PDE solver. Rather than including all resistance values which would cause most RDHs having only zeros for large values, we disregarded values beyond a certain threshold. To specify this threshold, we computed the 99% quantile across all patterns and picked the maximal value.

Finally, we set the bin number B introduced in Definition 1 to the value $B = 12$. We found empirically that smaller bin numbers give more accurate results for small data sets, while larger numbers perform better for larger data sets. $B = 12$ appears as a good tradeoff between these two regimes.

5.1.5 Additional features

In Sect. 3.2 we discussed the maximal concentration as an additional feature. Due to numerical inaccuracies when simulating a system, a few pixels might have an artificially large concentration. We aim here to disregard such values and instead estimate the concentration value of the highest plateau in a given pattern. To this end, we collected the concentration values of the pattern into a histogram of 25 bins and defined the maximal concentration as the location of the peak with the highest concentration value.

5.1.6 Data splitting

Consider the learning problem described in Sect. 4.1.1: given a data set $\mathcal{D}_m = \{(x_i, y_i)\}_{i\in[m]} \subset \mathcal{X} \times \mathcal{Y}$ with RDHs x_i and vectors y_i comprising parameter values of the model, we aimed to learn a prediction function $f : \mathcal{X} \to \mathcal{Y}$. In practice, we did not use the whole data set $\mathcal{D}_m$ for training, but split it into mutually disjoint training, test and validation sets, which respectively comprised 60%, 20% and 20% of $\mathcal{D}_m$. The training set was used to train a model for given hyperparameters, while the validation set was in turn used to optimise the hyperparameters. The NRMSE of the trained model was subsequently computed on the test set.

For small data sets $\mathcal{D}_m$ with $m \leq 500$, we observed large variations in the resulting NRMSE values. To obtain more robust estimates we split a total data set of 1000 points into subsets $\mathcal{D}_m$ of size m for $m \leq 500$, performed training and computed the NRMSE value for each $\mathcal{D}_m$ as described above, and took the average over these NRMSE values. For example, if $m = 100$, then we averaged over $1000/100 = 10$ data sets. For data sets of size $m > 500$, the procedure above was performed on the single data set without averaging.

For convolutional neural networks trained on raw pattern data we found the results to be substantially more noisy than for the other learning methods trained on RDHs. Accordingly, we here averaged the NRMSE values over more training sets: for $m \leq 1000$ points, we split a total set of 5000 points into data sets $\mathcal{D}_m$. For $m = 2000$, 5000 and 10,000 we used total data sets of 20,000 points. Finally, for $m = 20{,}000$ we trained the models three times with random initialization on the same data set and averaged subsequently.

5.1.7 Target variable preprocessing

We normalised each component of the target variable by its maximal value over the whole data set, i.e.,

$$y'_{i,j} = \frac{y_{i,j}}{\max\{y_{l,j}\}_{l\in[n]}}, \quad i \in [n], \quad j \in [d], \tag{80}$$

where d is the dimension of the target variable corresponding to the number of parameters to be learned, and n is the number of data points. We use these normalised target variables for both the regression task in Sect. 5.3 and for clustering in Sect. 5.4.

5.1.8 Support-vector regression

The training procedure for learning a single parameter, i.e. a scalar-valued target variable, using support-vector regression (SVR) is described in Sect. 4.2.1. We choose the hyperparameters γ [cf. Eq. (72) for the exponential χ^2-kernel and Eq. (76) for the Wasserstein kernel] and λ [cf. Eq. (53a)] by minimising the NRMSE on the validation set on a grid in the two parameters [note that the χ^2-kernel of Eq. (71) does not contain any hyperparameter]. In some cases, we performed a second optimisation over a finer grid centered around the optimal parameters from the first run. We found this to lead to only minor improvements, however. The model was then evaluated on the test set for the optimal parameters and the resulting NRMSE value is reported.

For learning multiple parameters we applied the SVR approach separately to each target parameter and subsequently computed the joint NRMSE value.

5.1.9 Operator-valued kernels

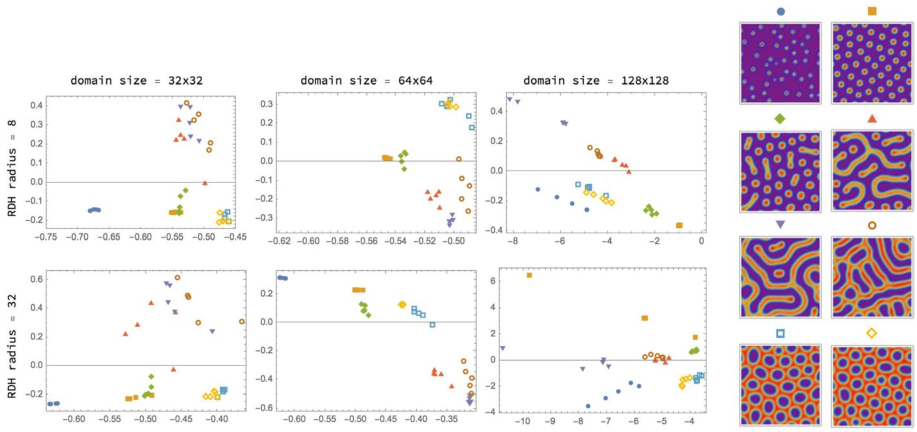
For learning multiple parameters jointly, i.e. a vector-valued target variable, we used the operator-valued kernel method described in Sect. 4.2.2. In addition to the input kernel parameter γ and regression parameter λ used for support-vector regression, we here also had to optimise the scale parameter of the output kernel [cf. the discussion after Eq. (68e)]. Optimisation of these hyperparameters was performed on a grid as in the SVR case, but this time jointly for all target parameters.

5.1.10 Feedforward neural networks

We employed feedforward neural networks both for learning a single parameter as well as learning multiple parameters jointly from RDHs. We used Mathematica's© build-in `NetTrain` function with the Adam optimization algorithm and the mean-squared loss function for training (Wolfram Research, 2021). The network architecture that gives the minimal loss on the validation set along the training trajectory was selected and evaluated on the test set to obtain the NRMSE value. Training was performed for T_f training steps with early stopping if the error on the validation set does not improve for more than T_e steps. For the data set sizes 20 and 50 we used $(T_f, T_e) = (4 \times 10^5, 10^5)$, for data set sizes 100, 1000 and 2000 we used $(T_f, T_e) = (2 \times 10^5, 5 \times 10^4)$, and for data set sizes ≥ 5000 we used $(T_f, T_e) = (10^5, 2 \times 10^4)$.

5.1.11 Convolutional neural networks

Convolutional neural networks were trained on the raw simulation data of the first species, i.e. *not* on RDHs. We used Abadi et al. (2015) and Chollet et al. (2015) for this purpose.



(a) (b)

Fig. 11 Visualization of sets of RDHs. **a** Dimensionality reduction of simulation results of the Gierer–Meinhardt model defined in Eq. (4) for eight equally-distanced values of parameter c on the interval [0.01, 1.15] indicated by different symbols, and five different random initial conditions each. The other parameters are fixed to $a = 0.02, b = 1, \delta = 100$. The resulting RDHs are embedded into the two-dimensional plane using latent semantic analysis (Berry et al., 1995). Results are shown for domain sizes 32×32, 64×64 and 128×128 and radii 8 and 32. We found that while points corresponding to different patterns do not separate into distinct clusters for a domain size of 32×32, they do so for a domain size of 64×64. For a domain size of 128×128, this separation appears even more pronounced. This demonstrates that resistance distance histograms successfully encode characteristic features of patterns while averaging out noise, if the domain size is chosen large enough. **b** Patterns for the eight different c values shown in (**a**) for one initial condition each

We employed the same training procedure as for feedforward neural networks. For the number of training steps we used $(T_f, T_e) = (500, 100)$.

5.2 Robustness of resistance distance histograms

Ideally, the RDHs should be characteristic of patterns of different types, while being robust to noise in the patterns due to noise in the initial conditions, and being invariant under immaterial spatial pattern transformations (translation, rotation). In other words, patterns arising from simulations of the same model for different initial conditions should give rise to RDHs that do not differ substantially, while patterns generated by different models should lead to larger deviations. Figure 4 visualizes these properties by means of a few examples and they will be assessed more quantitatively in the following.

To this end, we simulated the Gierer–Meinhardt model introduced in Sect. 2.2 for eight different values for c with the other parameters fixed. For each value of c we simulate the model for five random initial conditions. We subsequently embed the corresponding RDHs into two dimensions. Figure 11 shows the results for differing domain sizes. We observe that the points are reasonable well clustered for a domain size of 32×32 pixels, with a substantial improvement when increasing the domain size to 64×64 pixels. This is to be expected, since a larger snapshot of a pattern should reduce the noise in the corresponding RDHs. The clustering appears to improve slightly upon further increase of domain size to 128×128 pixels.

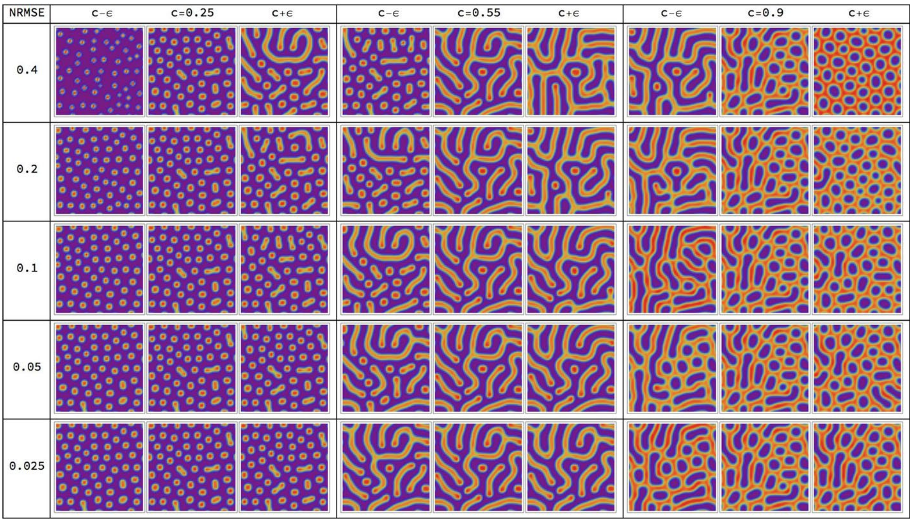


Fig. 12 Pattern accuracy. Simulation results of the Gierer–Meinhardt model in Eq. (4) for fixed parameters $a = 0.02, b = 1$ and $\delta = 100$ and varying values for parameter c on a 64×64 domain. The c values are varied around a central value such that they correspond to a certain NRMSE value, and different rows correspond to different NRMSE values. We find that for an NRMSE value of 0.4 the patterns deviate quite substantially from each other, while they look relatively similar for a value of 0.2 already. Decreasing the NRMSE value further successively decreases the deviations in the patterns. For NRMSE values smaller than 0.05 different patterns are hardly distinguishable anymore. This illustrates the criteria defined in Sect. 4.1.2 for rating parameter prediction as "good" or "excellent", respectively

The fact that the different noisy realisations of the patterns separate well indicates that the RDHs average out this noise while encoding the characteristic features of the patterns to a large degree.

In view of these results, we only consider domain sizes of 64×64 and 128×128 in the following.

5.3 Gierer-Meinhardt model: Learning a single parameter

In this section we consider the prediction problem of learning a map from RDHs (potentially combined with the additional features described in Sect. 3.2) generated from simulations of the Gierer–Meinhardt model in Eq. (4) as described in Sect. 2.4 onto the corresponding kinetic parameters. The training data thus consists of pairs (x_i, y_i) with x_i being an RDH and y_i being a set of kinetic parameters of the model. As outlined in Sect. 5.1 we split the data into a training, validation and test set, where the former two are used to train the models and learn hyperparameters, and the latter is used to evaluate the model's error in terms of the normalised root-mean squared error (NRMSE) (cf. Sect. 4.1.2).

Figure 12 visualises how patterns vary for varying parameters corresponding to different NRMSE values. We observe that the patterns look reasonably similar for NRMSE values of 0.2, while they are hardly distinguishable anymore for values below 0.05.

We start by varying the parameter c and fixing the other parameters to $a = 0.02, b = 1$ and $\delta = 100$ [cf. Eq. (4)]. We randomly sample 2×10^4 values for c on the interval [0, 1.15], solve the corresponding PDE in Eq. (4) and compute the resulting RDHs as described in

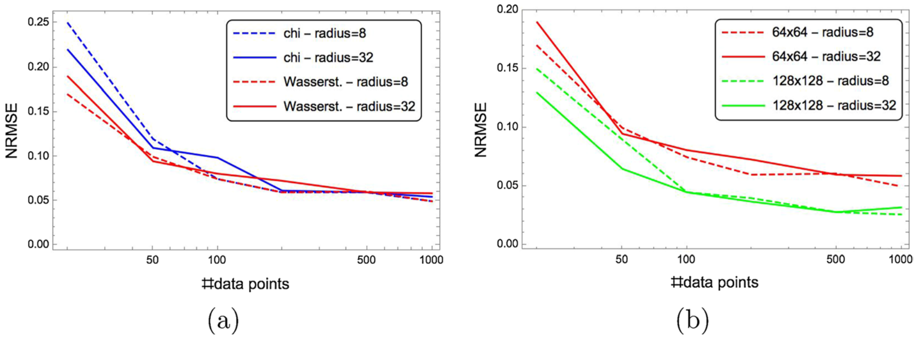


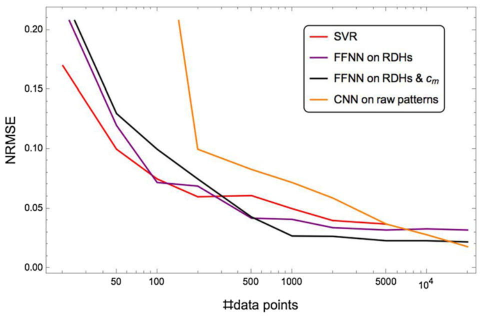
Fig. 13 Kernel-based parameter prediction. NRMSE values for support-vector regression of parameter c of the Gierer–Meinhardt model in Eq. (4). We vary c uniformly on the interval [0, 1.15] and fix the other kinetic parameters to $a = 0.02, b = 1$ and $\delta = 100$, the scaling parameter to $s = 0.25$. The figures show the NRMSE values for varying data set sizes and for the two RDH-radii 8 and 32 (cf. Definition 1). **a** The figure shows the results for both the exponential χ^2—and Wasserstein kernel [cf. Eqs. (71) and (76), respectively] and a domain size of 64×64. We find that even for small data sets of only 20 data points a reasonable good NRMSE value of about 0.2 is achieved (cf. Fig. 12). This value successively decreases for increasing data set sizes down to a value of about 0.05. While the results for the two different RDH-radii do not vary substantially, the Wasserstein kernel outperforms the χ^2 kernel for small data sets. **b** The figure shows the results for the Wasserstein kernel in Eq. (76) and the two domain sizes 64×64 and 128×128. We observe about about 10–50% better results for the 128×128 domain. These results show that about 1000 data points suffice to reach the NRMSE value 0.05, which is quite accurate according to the scale of NRMSE values discussed and fixed in Fig. 12 and in Sect. 4.1.2, respectively

Sect. 3.1. Several different types of patterns emerge in this range of c values as can be seen in Fig. 11b.

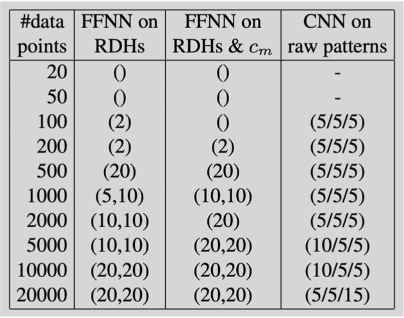
5.3.1 Support-vector regression

Fig. 13a shows the NRMSE obtained by training the support-vector regression model with both the exponential χ^2-kernel and the Wasserstein kernel as introduced in Sect. 4.2.3, for the two RDH-radii 8 and 32. We find that small data sets of only 20 data points allow to learn the parameter c reasonable well with NRMSE values in the range 0.17–0.25, which indicates that the RDHs average out noise in the patterns to a large degree (as already noted in Sect. 5.2).

Increasing the number of data points successively reduces the NRMSE down to values of 0.059–0.055 for 1000 data points. We observe that even for relatively small snapshots of the patterns of only 64×64 pixels, the RDHs allow to learn the parameter c with quite high accuracy. While we don't observe a substantial difference between the two analysed RDH-radii we do find that the Wasserstein kernel gives more accurate results for small data sets, while the two kernels perform similar for larger data sets. This finding is plausible because unavoidable binning effects like slightly shifted histogram entries impact RDHs more when the data set is small, but are reasonably compensated through 'mass transport' by the Wasserstein kernel. We ran the same experiments for the symmetric χ^2 kernel introduced in Sect. 4.2.3 and obtained worse results than for the other two kernels (results not shown). Therefore, in the rest of this paper, we will use the Wasserstein kernel.



(a)



#data points	FFNN on RDHs	FFNN on RDHs & c_m	CNN on raw patterns
20	()	()	-
50	()	()	-
100	(2)	()	(5/5/5)
200	(2)	(2)	(5/5/5)
500	(20)	(20)	(5/5/5)
1000	(5,10)	(10,10)	(5/5/5)
2000	(10,10)	(20)	(5/5/5)
5000	(10,10)	(20,20)	(10/5/5)
10000	(20,20)	(20,20)	(10/5/5)
20000	(20,20)	(20,20)	(5/5/15)

(b)

Fig. 14 Kernel- versus NN-based parameter prediction. **a** NRMSE values for the same setting as in Fig. 13, for a RDH radius of 8 and a domain size of 64 × 64. The figure shows the results obtained using support-vector regression (SVR) trained on RDHs, feedforward neural networks (FFNNs) once trained on RDHs and once trained on RDHs combined with the maximal concentration c_m as additional feature (cf. Sect. 3.2), as well as convolutional neural networks (CNNs) trained on the raw patterns, as described in in Sect. 4.3. We observe that the FFNNs perform slightly worse than the support-vector regression for small data sets, similar for intermediate data set sizes of 100–200 points, and slightly better for larger data sets. The FFNNs trained on RDHs and the maximal concentration c_m perform slightly worse for less than 500 data points and slightly better for larger sets. The NRSMSE value seems to level off and not decrease any further for larger data sets. For the CNNs trained on the raw patterns we find that NRMSE values are substantially larger than the corresponding FFNN and SVR values, which one may expect due to overfitting. The NRMSE values lie outside of the shown plot range for data sets smaller than 200 points. The difference decreases for increasing data sets until the CNNs eventually become more accurate for $10^4 - 2 \times 10^4$ data points. Note that no SVR results for $\geq 10^4$ data points are shown since our basic QP-solver failed to converge. However, the SVR method only outperforms the other methods for quite small data sets anyway. **b** Architectures for both the FFNNs and CNNs that gave the best performance and whose results are shown in (**a**) (see Sect. 4.3 for the used notation)

Figure 13b shows the results for an increased domain size of 128 × 128. This larger domain leads to improved NRMSE values of about 10–50%, with a larger improvement for larger data sets. As already noted in Sect. 5.2, this improvement is to be expected since a larger snapshot of a pattern allows the RDHs to average out local fluctuations more efficiently.

For simplicity and computational convenience, however, we use only 64 × 64 domain sizes in the following.

5.3.2 Neural networks

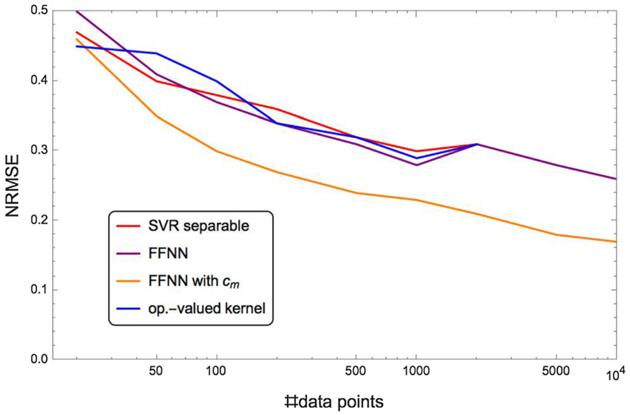
Figure 14 shows the regression results for learning the parameter c using feedforward neural networks (FFNN) for data sizes of up to 2×10^4 points. As one may expect, the FFNNs perform worse than support-vector regression for small data sets and better for larger data sets. Using FFNNs it is feasible to use data sets beyond the maximum of 5000 points used for support-vector regression. However, we find that the NRSME value appears to not improve any further beyond about 2000 points. This may be expected since in the computation of the RDHs some information about the patterns is inevitably lost, meaning there is a lower bound of how accurate the parameter can be learned in the limit of an infinitely large data set. We point out, however, that this saturation effect happens at NRSME values ≤ 0.03 which is very accurate (cf. Fig. 12).

5.3.3 Additional features

In Sect. 3.2 we introduced two additional features to account for certain symmetries of the RDHs, namely the maximal concentration c_m and the number of connected components n_c of a pattern. Figure 14 shows the results obtained by training FFNNs on RDHs with c_m as an additional feature. We find slightly larger NRMSE values for small data sets with less than 500 points, and slightly smaller NRMSE values for larger data sets. In contrast, using the number of connected components n_c as an additional feature did not give rise to notably more accurate results (results not shown).

5.3.4 Benchmark: CNN on raw data

Figure 14 also shows the NRMSE values obtained from training convolutional neural networks (CNNs) *directly on the raw pattern data* obtained through simulation (cf. Sect. 4.3). For data set sizes of ≤ 200 data points we found substantially larger NRMSE values than from FFNNs trained on RDHs. This shows once again that the RDHs efficiently encode most of the relevant information while averaging out noise, allowing for more accurate parameter learning for data sets of small and medium size. As one might except, the difference between the CNN and FFNN results becomes smaller for larger data set sizes since



(a)

#data points	FFNN architecture
20	()
50	()
100	()
200	()
500	(5,5)
1000	(5,5)
2000	(10,10)
5000	(20,20)
10000	(20,20)

(b)

Fig. 15 Joint parameter prediction. **a** NRMSE values for learning all four kinetic parameters a, b, c and δ of the Gierer–Meinhardt model in Eq. (4). The figures show the NRMSE values for varying data set sizes and for the RDH-radius 8 (cf. Definition 1) and a domain size of 64 × 64, for both cases of kernel-based learning the four parameters individually and jointly as outlined in Sects. 4.2.1 and 4.2.2, respectively, using the Wasserstein kernel [cf. Eqs. (76)]. In addition, the NRMSE values of feedforward neural networks (FFNNs) (cf. Sect. 4.3) are shown, once trained only on RDHs and once trained using the maximal concentration c_m as additional feature (cf. Sect. 3.2). As one may expect, the NRMSE values are substantially larger here than in the scalar case of learning just one parameter for the same number of data points (cf. Fig. 14) and once again decrease for increasing data sets. We find that all three methods trained on RDHs perform very similar. This implies, in particular, that no correlation among the output parameter values could be exploited for prediction. In contrast, including the maximal concentration c_m as an additional feature leads to substantially improved NRMSE values with an improvement of up to 35% for large data sets. **b** Architectures for the FFNNs that gave the best performance and whose results are shown in (**a**). We found the same optimal architectures for both training the FFNNs on the RDHs only and on the RDHs together with the maximal concentration c_m

the CNNs can effectively average out the noise themselves when a sufficient large number of data points are provided. Consequently, we found that CNNs become more accurate than the other methods for large data sets of about $10^4 - 2 \times 10^4$ data points.

5.4 Gierer-Meinhardt model: Jointly predicting four parameters

We next consider the problem of learning all four kinetic parameters a, b, c and δ of the Gierer–Meinhardt model in Eq. (4). We vary a, b, c and δ uniformly on the interals [0.01, 0.7], [0.4, 2], [0.02, 7] and [20, 200], respectively. The scaling parameter was set to $s = 0.4$ and assumed to be known. Parameter combinations for which the system does not possess a Turing instability and hence does not produce a pattern in simulations are disregarded.

Figure 15 shows the NRMSE results for the separable SVR model, for the joint kernel-based model, and for feedforward neural networks (FFNNs) trained on RDHs of radius $r = 8$ and a domain size of 64×64. We find that the three methods perform similar which means, in particular, that no correlation among the output parameters could be exploited for joint prediction in order to outperform separable parameter prediction (the SVR methods we trained only for data sets of up to 2×10^3 points). We further observe the NRMSE values to be substantially higher than in the scalar case (cf. Fig. 14) for the same number of data points, which is to be expected when learning more parameters. The NRMSE value again successively decreases for increasing data set sizes, with a minimal value of about 0.26 for FFNNs and 10^4 data points, which is substantially larger than the minimal value of 0.033 obtained in the scalar case for the same data set size and radius (cf. Fig. 14). However, here the curve does not appear to have levelled off yet for 10^4 data points as in the scalar case, and increasing the data set size further should further increase accuracy.

We performed the same experiment as shown in Fig. 15 but for RDHs radius $r = 32$ and found similar results (results not shown).

5.4.1 Combining RDHs of different radii

In Sect. 3 we argued that the RDH radius r determines the scale at which the local structure of a Turing pattern is resolved. We used the two radii $r = 8$ and $r = 32$ in the results presented so far in Figs. 13, 14 and 15 and found similar results for the two. However, since RDHs with radius $r = 8$ should more accurately capture characteristics of patterns on small scales and $r = 32$ should be able to capture larger-scale characteristics, one might expect that combining the two should provide more information than each of them individually and might therefore give rise to more accurate results. We trained feedforward neural networks on the RDHs of the two radii taken together as features for the same setting as in Fig. 15, but did not obtain notably more accurate results (results not shown).

5.4.2 Additional features

While we found in Sect. 5.3 that using the maximal concentration c_m as an additional feature did only slightly improve NRMSE values and only for large data sets when learning a single parameter (cf. Fig. 14), we here find a substantial improvement for all data set sizes, with improvements of up to 35% for large data set sizes as can be seen in Fig. 15. As in the scalar case, we find that using the number of connected components n_c does not improve results (results not shown).

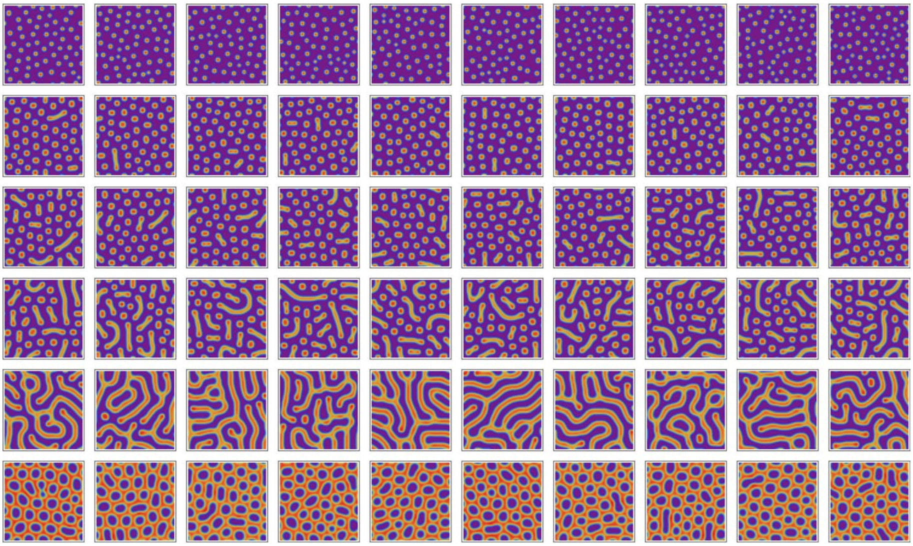


Fig. 16 Clustering Turing patterns. Cluster of patterns obtained by varying parameter c as described in Sect. 5.3. Each row shows 10 sample patterns from a cluster that comprises a large number of patterns within a small radius, measured by the squared Wasserstein distance (74) between the corresponding resistance distance histograms (RDHs). This result demonstrates how RDHs and a corresponding distance function represent distinct types of patterns

5.5 Cluster of patterns

We explored the geometry of patterns represented by resistance distance histograms (RDHs) and the squared Wasserstein distance (74). Parameter c in the Gierer–Meinhardt model in Eq. (4) was varied as described in Sect. 5.3, and 1000 patterns and corresponding RDHs were computed as detailed in Sect. 2.4. Next, we examined the neighborhood graph in which two patterns with corresponding RDHs x, x' are adjacent if $d_W^2(x, x') \leq 0.05$. As a result, about 84% of all patterns were contained in one of the six clusters corresponding to the connected components with the largest number of patterns.

Figure 16 depicts 10 sample patterns taken from each cluster. This result shows that RDHs according to Definition 1, together with an appropriate distance function, are suited for clustering patterns into qualitatively different categories.

6 Conclusion

We introduced a novel learning-based approach to Turing pattern parameter prediction. A key difference to existing work is that any *single* observed pattern is *directly* mapped to a predicted model parameter value, allowing to infer model parameters from single data points. Major ingredients of the method are (1) the (almost) invariant representation of Turing patterns by histograms of resistance distances computed within patterns, and (2) a kernel-based pattern similarity measure based on the Wasserstein distance that takes properly into account minor but unavoidable binning effects.

We compared classical reproducing kernel Hilbert space methods using basic kernels for single parameter prediction and operator-valued kernels for jointly predicting all parameters. These methods performed best for small and medium-sized training data sets. In addition, we evaluated various feedforward neural network architectures for prediction. As for single parameter prediction, these methods performed best for larger training data sets and but were on a par only with kernel-based methods in the case of joint parameter prediction.

Finally, we applied convolutional neural networks to raw pattern data directly. We found that, for very large training data sets with $\geq 2 \times 10^4$ data samples, they outperformed all other methods. However, it remains unexplained what internal pattern representations are used.

Overall, we observed excellent parameter prediction of single parameters even for small data sets with $\leq$ 1000 data samples, and fairly accurate joint prediction of all parameters for large data sets. Our results indicate that the latter predictions should further improve when even larger data sets can be used for training. We leave such experiments for future work.

We suggest to focus on two aspects in future work. In this paper, the Gierer–Meinhardt model was chosen in order to conduct a representative case study. In practical applications, selecting also the model among various candidates, besides estimating its parameters, might be desirable. Furthermore, our current approach cannot quantify the uncertainty of parameter prediction and in this respect falls short of statistical approaches like Campillo-Funollet et al. (2019) and Kazarnikov and Haario (2020). On the other hand, our approach can be applied to single patterns, rather than to ensemble of patterns like the approach Kazarnikov and Haario (2020), and further input data like initial conditions as in Campillo-Funollet et al. (2019) that are unknown in practice, are not required. Resolving these pros and cons defines an attractive program for future research.

Acknowledgements DS gratefully acknowledges support from a “Life?” programme grant from the Volkswagen Stiftung and the Biotechnology and Biological Sciences Research Council (Grant Number BB/P028306/1).

Author contributions DS and CS designed and performed the research. Both authors wrote the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Availability of data and material Not applicable.

Code availability No.

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., & Ghemawat, S. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Software available from https://www.tensorflow.org/

Álvarez, M. A., Rosasco, L., & Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning, 4*(3), 195–266.

Bachoc, F., Gamboa, F., Loubes, J.-M., & Venet, N. (2018). A Gaussian process regression model for distribution inputs. *IEEE Transactions on Information Theory, 64*(10), 6620–6637.

Bapat, R. B. (2014). *Graphs and matrices*. Springer.

Berg, C., Christensen, J. P. R., & Ressel, P. (1984). *Harmonic analysis on semigroups: Theory of positive definite and related functions*. Springer.

Berlinet, A., & Thomas-Agnan, C. (2004). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer.

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review, 37*(4), 573–595.

Bouhamidi, A., & Jbilou, K. (2008). A note on the numerical approximate solutions for generalized matrix equations with applications. *Applied Mathematics and Computation, 206*(2), 687–694.

Bracewell, R. N. (2000). *The Fourier transform and its applications* (3rd ed.). McGraw-Hill.

Brémaud, P. (2017). *Discrete probability models and methods*. Springer.

Brouard, C., Szafranski, M., & d'Alché Buc, F. (2016). Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research, 17*, 1–48.

Campillo-Funollet, E., Venkataraman, C., & Madzvamuse, A. (2019). Bayesian parameter identification for Turing systems on stationary and evolving domains. *Bulletin of Mathematical Biology, 81*(1), 81–104.

Castets, V., Dulos, E., Boissonade, J., & De Kepper, P. (1990). Experimental evidence of a sustained standing Turing-type nonequilibrium chemical pattern. *Physical Review Letters, 64*(2953), 24.

Chollet, F. et al. (2015). *Keras*. https://keras.io

Cucker, F., & Smale, S. (2001). On the mathematical foundations of learning. *Bulletin of AMS, 39*(1), 1–49.

Doyle, P. G., & Snell, J. L. (1984). *Random walks and electric networks*. Cambridge University Press.

Economou, A. D., Ohazama, A., Porntaveetus, T., Sharpe, P. T., Kondo, S., Basson, M. A., Gritli-Linde, A., Cobourne, M. T., & Green, J. B. A. (2012). Periodic stripe formation by a Turing mechanism operating at growth zones in the mammalian palate. *Nature Genetics, 44*(3), 348–351.

Evgeniou, T., Miccelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research, 6*, 615–637.

Evgeniou, T., Pontil, M., & Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics, 13*, 1–50.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports, 486*, 75–174.

Garvie, M. R., Maini, P. K., & Trenchea, C. (2010). An efficient and robust numerical algorithm for estimating parameters in Turing systems. *Journal of Computational Physics, 229*(19), 7058–7071.

Garvie, M. R., & Trenchea, C. (2014). Identification of space-time distributed parameters in the Gierer–Meinhardt reaction–diffusion system. *SIAM Journal on Applied Mathematics, 74*(1), 147–166.

Gatenby, R. A., & Gawlinski, E. T. (1996). A reaction–diffusion model of cancer invasion. *Cancer Research, 56*(24), 5745–5753.

Gierer, A., & Meinhardt, H. (1972). A theory of biological pattern formation. *Kybernetik, 12*(1), 30–39.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77*, 354–377.

Hairer, E., Nørsett, S. P., & Wanner, G. (2008). *Solving ordinary differential equations I* (3rd ed.). Springer.

Hein, M., & Bousquet, O. (2005). Hilbertian metrics and positive definite kernels on probability measures. In *AISTATS: Proceedings*.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *Annals of Statistics, 36*(3), 1171–1220.

Holmes, E. E., Lewis, M. A., Banks, J. E., & Veit, R. R. (1994). Partial differential equations in ecology: Spatial interactions and population dynamics. *Ecology, 75*(1), 17–29.

Honeine, P., & Richard, C. (2011). Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine, 28*(2), 77–88.

Horn, R. A., & Johnson, C. R. (2013). *Matrix analysis* (2nd ed.). Cambridge University Press.

Jung, H.-S., Francis-West, R. B., Widelitz, P. H., Jiang, T.-X., Ting-Berreth, S., Tickle, C., Wolpert, L., & Chuong, C.-M. (1998). Local inhibitory action of BMPs and their relationships with activators in feather formation: Implications for periodic patterning. *Developmental Biology, 196*(1), 11–23.

Kadri, H., Ghavamzadeh, M., & Preux, P. (2013). A generalized kernel approach to structured output learning. *Proceedings of Machine Learning Research, 28*, 471–479.

Karasözen, B., Uzunca, M., & Küçükseyhan, T. (2020). Reduced order optimal control of the convective FitzHugh–Nagumo equations. *Computers & Mathematics with Applications, 79*(4), 982–995.

Kazarnikov, A., & Haario, H. (2020). Statistical approach for parameter identification by Turing patterns. *Journal of Theoretical Biology, 501*, 110319.

Klein, D. J., & Randić, M. (1993). Resistance distance. *Journal of Mathematical Chemistry, 12*, 81–95.

Kondo, S., & Miura, T. (2010). Reaction–diffusion model as a framework for understanding biological pattern formation. *Science, 329*(5999), 1616–1620.

Landge, A. N., Jordan, B. M., Diego, X., & Müller, P. (2020). Pattern formation mechanisms of self-organizing reaction–diffusion systems. *Developmental Biology, 460*(1), 2–11.

Martcheva, M. (2015). *An introduction to mathematical epidemiology*. Text in applied mathematics, 61. Springer.

Micchelli, C. A., & Pontil, M. (2005). On learning vector-valued functions. *Neural Computation, 17*, 177–204.

Minh, H. Q., Bazzani, L., & Murino, V. (2016). A unifying framework in vector-valued reproducing kernel Hilbert spaces for manifold regularization and co-regularized multi-view learning. *Journal of Machine Learning Research, 17*(25), 1–72.

Murphy, L., Venkataraman, C., & Madzvamuse, A. (2018). Parameter identification through mode isolation for reaction–diffusion systems on arbitrary geometries. *International Journal of Biomathematics, 11*(04), 1850053.

Murray, J. D. (1982). Parameter space for Turing instability in reaction diffusion mechanisms: A comparison of models. *Journal of Theoretical Biology, 98*(1), 143–163.

Murray, J. D. (2001). *Mathematical biology II: Spatial models and biomedical applications*. Springer.

Nakamasu, A., Takahashi, G., Kanbe, A., & Kondo, S. (2009). Interactions between zebrafish pigment cells responsible for the generation of Turing patterns. *Proceedings of the National Academy of Sciences, 106*(21), 8429–8434.

Pathak, H. K. (2018). *An introduction to nonlinear analysis and fixed point theory*. Springer.

Paulsen, V. I., & Raghupathi, M. (2016). *An introduction to the theory of reproducing kernel Hilbert spaces*. Cambridge University Press.

Pertham, B. (2015). *Parabolic equations in biology*. Springer.

Peyré, G., & Cuturi, M. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning, 11*(5–6), 355–607.

Raspopovic, J., Marcon, L., Russo, L., & Sharpe, J. (2014). Digit patterning is controlled by a Bmp-Sox9-Wnt Turing network modulated by morphogen gradients. *Science, 345*(6196), 566–570.

Rockafellar, R. T., & Wets, R.J.-B. (2009). *Variational analysis* (3rd ed.). Springer.

Santambrogio, F. (2015). *Optimal transport for applied mathematicians*. Birkhäuser.

Schaeffer, D. G., & Cain, J. W. (2016). *Ordinary differential equations: Basics and beyond*. Springer.

Scholes, N. S., Schnoerr, D., Isalan, M., & Stumpf, M. P. H. (2019). A comprehensive network atlas reveals that Turing patterns are common but not robust. *Cell Systems, 9*(3), 243–257.

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). *A generalized represent theorem, computational learning theory* (Vol. 2111, pp. 416–426). Springer.

Schölkopf, B., Mika, S., Burges, C. J. C., Knirsch, P., Müller, K.-R., Rätsch, G., & Smola, A. J. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks, 10*(5), 1000–1017.

Seto, M., Suda, S., & Taniguchi, T. (2014). Gram matrices of reproducing kernel Hilbert spaces over graphs. *Linear Algebra and its Applications, 445*, 56–68.

Sgura, I., Lawless, A. S., & 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.

Shangerganesh, L., & Sowndarrajan, P. T. (2020). An optimal control problem of nonlocal Pyragas feedback controllers for convective FitzHugh–Nagumo equations with time-delay. *SIAM Journal on Control and Optimization, 58*(6), 3613–3631.

Sick, S., Reinker, S., Timmer, J., & Schlake, T. (2006). WNT and DKK determine hair follicle spacing through a reaction–diffusion mechanism. *Science, 314*(5804), 1447–1450.

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing, 14*, 199–222.

Stoll, M., Pearson, J. W., & Maini, P. K. (2016). Fast solvers for optimal control problems from pattern formation. *Journal of Computational Physics, 304*, 27–45.

Tan, Z., Chen, S., Peng, X., Zhang, L., & Gao, C. (2018). Polyamide membranes with nanoscale Turing structures for water purification. *Science, 360*(6388), 518–521.

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

Uzunca, M., Küçükseyhan, T., Yücel, H., & Karasözen, B. (2017). Optimal control of convective FitzHugh–Nagumo equation. *Computers & Mathematics with Applications, 73*(9), 2151–2169.

Vittadello, S. T., Leyshon, T., Schnoerr, D., & Stumpf, M. P. H. (2021). Turing pattern design principles and their robustness. *Philosophical Transactions of the Royal Society A, 379*(2213), 20200272.

Whaba, G. (1990). *Spline models for observational data*. SIAM.

Wolfram Research. (2021). *Mathematica, version 12.3.1*.

Woolley, T. E., Krause, A. L., & Gaffney, E. A. (2021). Bespoke Turing systems. *Bulletin of Mathematical Biology, 83*(5), 1–32.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.