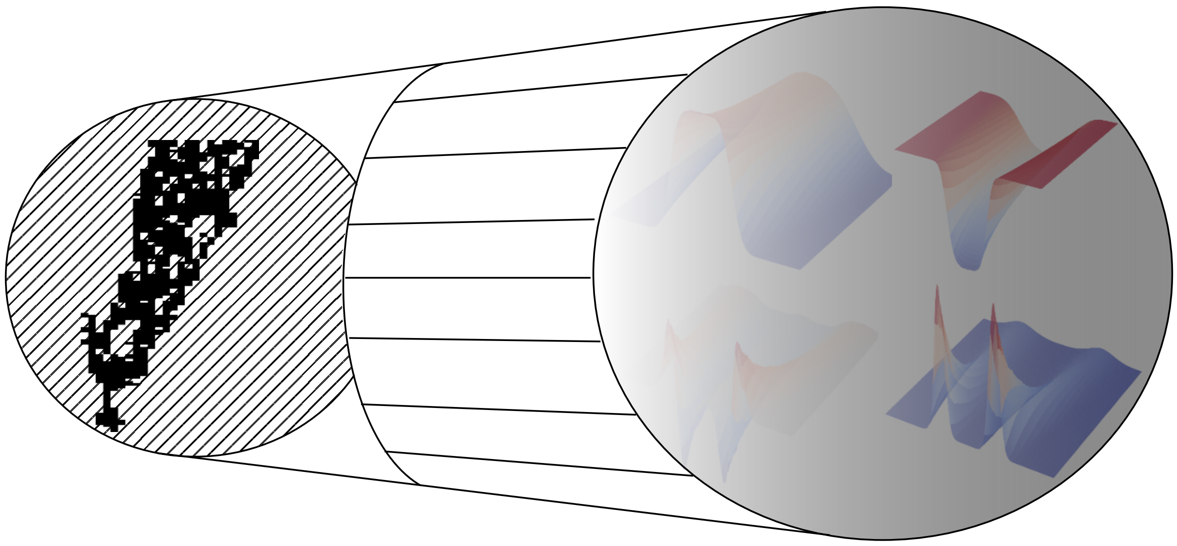
Theoretical Physics

Identifying PDEs in Interacting Particle Systems using Data-Driven Techniques

Master Thesis

Hanne Leeuwis



Dr. Joost de Graaf First supervisor
Institute for Theoretical Physics - Utrecht University

Dr. Deb Panja Second supervisor
Department of Information and Computing Sciences - Utrecht University

Jeroen Roberts Daily supervisor
Institute for Theoretical Physics - Utrecht University

July 10, 2022

Acknowledgement

I would like to thank my supervisor Joost de Graaf for all of the feedback and support he has given me throughout the year. I learned a lot of new things from him, and I could not be happier with the topic of the thesis. Special thanks to my daily supervisor Jeroen Roberts for all of the fruitful discussions we had, and for guiding me through the fascinating world of interacting particle systems. I also would like to thank my second supervisor and complex systems profile coordinator Deb Panja for approving the Machine Learning Theory course as part of the minor. The theoretical background proved to be very useful in understanding and developing the data-driven methods.

Finally, I would like to thank all of the students who were present in the master room throughout the year. The discussions we had were really fun, and I think I even have learned some things about string theory.

Abstract

Partial differential equations (PDEs) are powerful tools to describe large-scale dynamics of microscopic systems. However, deriving a PDE from first principles becomes significantly more involved when systems increase in complexity. In this thesis research, we will consider a data-driven approach to determine (new) PDEs for interacting particle models on a lattice, using the sparse regression based algorithm develop by Brunton *et al* [1]. We will focus on the average behaviour of the Eden process (this describes growth processes seen in bacteria) and the contact process (CP), which also incorporates death. When the Eden process is combined with the symmetric simple exclusion process (SSEP), *i.e.*, diffusive spreading and hard-core exclusion, the motion obeys the Fisher equation, which is a PDE that models the spread of growing populations. The CP is more complex as it possesses a phase transition with criticality. As of yet, the emergent dynamics have not been captured in the form of a coarse-grained PDE, which makes it an interesting case to study from both a biological and mathematical perspective. At the critical point of the CP, we find that the emergent behaviour can be captured by the Fisher equation with an interface broadening term, which explains the observed length scales at the boundary of the population. Our results highlight the strength of data-driven methods for discovering PDEs in complex particle systems. Applying these techniques to experimental results for bacterial colonies presents new opportunities to study their collective behaviour, which will be the focus of future research.

Title page image: The left picture is a space-time plot of discrete particle data obtained from the contact process at the critical point. With our data-driven techniques we found a PDE that describes the average behaviour of this particle system. On the right we have plotted the contributions of this PDE as a function of space and time. With these methods we are thus able to learn macroscopic dynamics by examining microscopic actions.

Contents

1 Introduction

Partial differential equations (PDEs) can provide a continuum description of the microscopic dynamics of particle systems. This is useful, because solving a PDE allows us to predict future states of a particle system as a whole, which in general is less expensive than computing all of the interactions and movements of the individual particles. A well known example of a continuum description of particle dynamics can be found in Brownian motion. This describes how suspended (colloidal) particles move randomly due to interactions with the surrounding medium [2]. This gives rise to a random walk, which in the continuum limit can be modelled by the diffusion equation [3]. This is for example extremely useful in simulating colloid dynamics, as it considerably speeds up the computational speed compared to molecular dynamics [4].

In this thesis, we will focus on PDEs of particle systems that are defined on a lattice. Of these systems there are many examples where it is possible to derive a PDE from first principles, such as the advection-diffusion equation from biased random walk [5], Burgers equation from the asymmetric simple exclusion process [6], and the Fisher equation from a combination of the Eden model and the symmetric exclusion process [7]. These systems are characterised and modelled via a simple (stochastic) rule set, such as moving to the left or right with a chosen probability in the case of a random walker. With these rules, it is then possible to write down a Master equation, which can then be converted to a PDE by taking a continuum limit [8] or with a path integral representation [9]. This is a straightforward calculation in the case of a random walk, but it becomes more difficult if we introduce rules that lead to complex, nonequilibrium and/or emergent dynamics, such as in active matter [10]. In these systems, combinations of microscopic rules can give rise to new emergent behaviour, which makes it hard to deduce a PDE from first principles [11]. That is, combining rules can effectively lead to new rules, which makes it hard to deduce a PDE from first principles.

The above consideration gives rise to the question how we can find PDEs of complex systems without relying on a Master equation or other first principles. One alternative and pragmatic approach would be to infer the macroscopic dynamics from data that follows from the particle rules. This idea is not new. In fact, in past and recent years there has been a lot of development of data-driven techniques to find PDEs from data, specifically in the field of hydrodynamics. See Ref. [12] for an in-depth review of past and recent developments of data-driven methods in fluid mechanics. The advantage of a data-driven method is that we do not have to think how the microscopic interactions give rise to complex and emergent behaviour, because we can directly measure the large-scale effect in terms of a PDE contribution.

One example of an utilisation of a data-driven technique on a particle system can be seen in Ref. [13]. Here, they successfully found a hydrodynamic description of active matter dynamics. To achieve this, they apply filtering techniques to coarse-grain discrete particle movements, which they then feed to a PDE searching algorithm developed by Brunton *et al.* [1]. This algorithm employs sparse regression, where sparseness is enforced by giving PDEs with many terms a large error. This is a promising technique, because it enables the capture of complex particle behaviour in terms of an interpretable PDE. In addition, as a simple test, Brunton *et al.* found the diffusion equation from Brownian motion [14].

From the final example we thus see that it is possible to find PDEs in particle systems. In this thesis, we want to develop a precise methodology for applying this to interacting lattice particle systems. To do this, we will examine two well-studied stochastic particle systems of increasing complexity. These systems are simple enough to generate trustworthy data, so this allows us to focus on finding the limits and pitfalls of the data-driven methods. To maximise the efficiency and reliability of the methods, we will look into finding the optimal filtering techniques, how we can approximate derivatives from noisy data, and how we can check whether the coefficients of PDEs are correct by using various initial conditions and multiple randomly generated data sets.

The first model that we will consider is a combination of the symmetric simple exclusion process (SSEP) and the Eden process, which are both lattice particle systems. We will focus on the 1D case. The SSEP is similar to the random walk, but it also includes hard-core exclusion [15], *i.e.*, particles cannot occupy the same site. The Eden process is a simple stochastic growth model, where particles can give birth to neighbouring lattice sites [16]. This model has been used to explain pattern formation seen in bacteria [17, 18]. Hence, combined with the SSEP, the system can be interpreted as a model for moving and growing bacteria. In literature, it is well known that in the continuum limit the Eden-SSEP density evolves according to the Fisher equation [7], which is a PDE that models the spread of growing populations [19]. However, there has not been a (numerical) study of the finite-size limit towards the continuum. We performed these calculations with the data-driven techniques and we found that the continuum limit of the dynamics is indeed given by the Fisher equation.

The second model is an extension of the Eden model, as it also incorporates death in addition to growth. This model is called the contact process (CP) and it has seen extensive study from mathematicians, because it undergoes a second order phase transition due to competition between birth and death [20]. A lot of study has gone into determining the critical exponents at the critical point [21] and the survival probabilities [22], but to the best of our knowledge, no coarse-grained PDE has been proposed that describes the deterministic dynamics of the average density. A search of the literature only revealed a stochastic Fisher equation both in the long range CP [23], and in the directed percolation model, which is a similar model that is within the same university class as the CP [24]. We focused on the critical point to examine the capability of the algorithm to detect emergent behaviour from complex systems. We found that the resulting PDE is indeed an extension of the Fisher equation, where the extra term explains the interface broadening we observed. This match is not exact, so there is still room for future investigations, such as testing the link between the stochastic PDE and the average dynamics.

In addition to the above physical systems, we also found some weaknesses of the sparse regression algorithm and we improved upon it during our study. Specifically, in the appendix from Ref. [14] a problem was reported regarding the biased random walk. Under certain choices of learning parameters, the output of the algorithm only contained diffusion and no advection. We overcame this issue by (numerically) estimating where the bias towards sparse PDEs should be halted in favour of the accuracy. In addition, we proposed a new error evaluation, which makes a more interpretable and reliable tradeoff between sparsity and accuracy of the PDE. Both strategies are proposed independently by the authors of the algorithm, so they could serve as new building blocks for further improvements of the methods.

With the data-driven methods we could successfully find (new) effective PDEs in interacting particle systems of increasing complexity. We focused on systems with growth processes (with and without diffusion), but these methods could be applied to other interacting particle systems as well, such as the asymmetric simple exclusion process. In addition, our research on the limits of finding PDEs with a finite number of particles, and finite lattice spacings could prove useful for experimentalists, who also need to deal with limited resources. Given the focus on the Fisher equation, a logical step would thus to apply our methods on experimental data obtained from growing bacterial colonies [25, 26].

Finally, let us give an outline of the thesis. We will start with describing the basics of the data-driven methods in section 2. In section 3 we will validate and test the techniques on smooth data obtained from the Fisher equation, and noisy data obtained from the non-interacting (biased) random walk on a lattice. In section 4, we will apply our techniques on the aforementioned interacting particle systems. We end this thesis with a discussion and conclusion in sections 5 and 6, respectively.

2 Methods

In this section we provide an outline of the data-driven techniques that we are going to apply to the interacting particle systems. We start by explaining how we can use a learning algorithm to determine PDEs from data. We will also explain the algorithm developed by Brunton *et al.* [14]. We did find that there were a number of issues regarding the reliability of the algorithm, so we will also introduce a number of improvements. After that we will focus on filters that will aid us to combat the noise we are bound to obtain from the particle data. We end with a number of remarks regarding the validity of found PDEs, and how we can make sure that we obtain the most reliable results.

2.1 Introduction to Learning Partial Differential Equations

Let us start by formulating what it means to learn a PDE from data. In general, a PDE is of the form

$$u_t(x,t) = N(u, \xi_j, x, t), \tag{1}$$

where u_t denotes the derivative of the solution data u with respect to time t. In general, we will also use subscript notation to denote spatial derivatives. For example, u_x is the first derivative with respect to position and u_{xxx} is the third. For now, we will assume that our data u is differentiable. We will discuss the exact form of u in the next section.

In general, $N(u, \xi_j, x, t)$ is a function that may depend on the solution u, its derivatives with respect to space and the coefficients ξ_j in front of these terms. For example, it can have the following form:

$$N(u, \xi_i, x, t) = \xi_0 + \xi_1 u + \xi_2 u^2 + \xi_3 u u_x + \xi_4 u_{xx} + \xi_5 u^3 u_{xxxx} + \dots. \tag{2}$$

The goal of learning a PDE is to find $N(u, \xi_i, x, t)$ such that it matches $u_t x, t$ for every x and t. The problem is of course that we do not know beforehand what $N(u, \xi_i, x, t)$ should contain. The first step is thus to insert terms in $N(u, \xi_i, x, t)$ that we would expect to be there. In general we will choose a basis set / ansatz, containing specific combinations of terms. In the next section we will discuss the precise form of this basis.

Now, each term in the basis set is fixed. Hence, the only variables that we can alter are the coefficients ξ_j in front of the terms. Of course, in physics we often assume that PDEs are sparse, in the sense that the relevant physics of the system can be described using only a small number of terms. If this assumption is true, then many coefficients in Eq. (2) will be zero. The goal is thus to find the coefficients of the correct terms, and set the coefficients of the wrong terms to zero.

A well known example of a sparse PDE is the diffusion equation

$$u_t = D u_{xx}, \tag{3}$$

where D is the diffusion constant. If we were to start with an ansatz such as Eq. (2), then the aim of a learning algorithm is to set all coefficients to zero, except for coefficient in front of the diffusion term u_{xx}. We have sketched this problem in Fig. 1 with a Gaussian initial condition and $D = 1$. We will specify in the next section how we obtained this data. On the left-hand side we have plotted u_t for all x and t. On the right-hand side we plot terms from our ansatz. We see that the field of u_t resembles that of $\xi_3 u_{xx}$. By eye we can estimate that $\xi_3 \approx 1$ and all other coefficients are zero. In the following sections we will explain how we can do that using a computer.

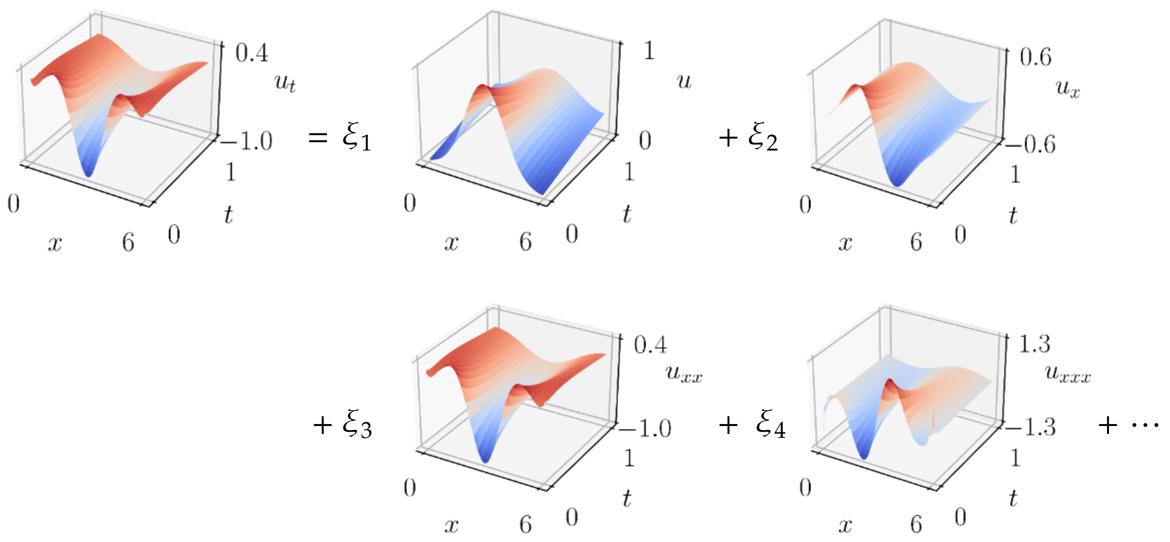


Figure 1: Sketch of how to find a PDE given a data field u. This sketch is similar to Eq.(2). For this example we solved the diffusion equation for $D = 1$ and plotted a number of terms depending on u as a function of x and t. The coefficients are denoted by ξ_j. Note how much the field of u_{xx} resembles that of u_t.

For this, we will use the `STRidge` algorithm developed by Brunton *et al.* [14], which stands for sequential threshold ridge regression. This is a linear regression algorithm with a (ridge) stabiliser that imposes sparsity of the PDE by throwing away small coefficients using thresholds. To understand what this all means, let us first explain what linear regression is and how to couple it to the above problem.

2.2 Linear Regression and Least Squares

The basic idea behind linear regression is to fit straight lines onto data. For example, we could have data of ice cream sales versus the temperature. To see if there is a linear correlation we can try to fit a straight line of the form $h = aT + b$ through our data points. Here T is the temperature and $a, b \in \mathbb{R}$ are the coefficients that need to be found. See Fig. 2 for a sketch. This is an example in 2D, but we can pose a general hypothesis in any dimension d. This function $h : \mathbb{R}^d \to \mathbb{R}$ is given by

$$h_\xi(\vec{z}) = \vec{z} \cdot \vec{\xi}, \tag{4}$$

where $\vec{z}$ belongs to the domain of predictor variables $\chi \subset \mathbb{R}^d$ and $\vec{\xi} \in \mathbb{R}^d$ are the coefficients [27, p. 123]. Here, we assume that $\vec{z}$ contains independent variables. This is an important point, and we will come back to this later.

The task of linear regression is to find the coefficients that minimise the empirical risk, which is defined by

$$L(h_\xi) = \frac{1}{m} \sum_{i=1}^{m} \ell(h_\xi, \vec{z}_i, y_i), \tag{5}$$

where m is the size of the data, $y_i \in \mathbb{R}$ is the label (*e.g.*, ice cream sales) and $\ell(h_\xi, \vec{z}_i, y_i)$ is the loss function, which is a measure how well the straight line fits the data. The form of ℓ which we will use, is the square-loss function and it is simply given by

$$\ell(h_\xi, x_i, y_i) = (y_i - h_\xi(\vec{z}_i))^2. \tag{6}$$

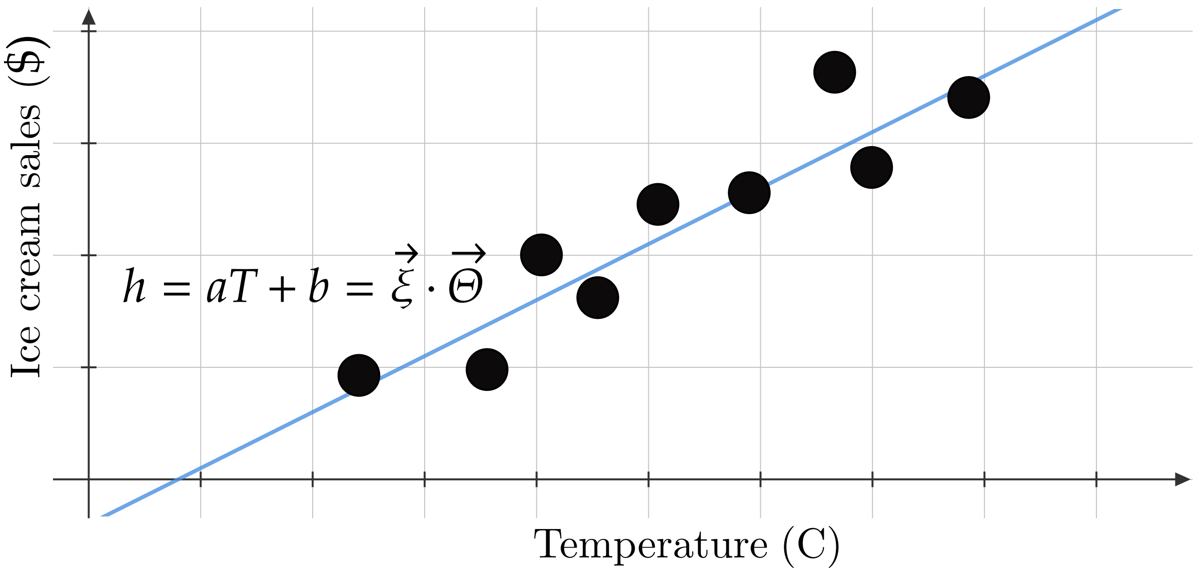


Figure 2: Sketch of a linear regression problem, where scatter plot ice cream sales data as a function of the temperature T. The goal of linear regression is to fit a linear function through the data points. The data is linear as well, hence a logical hypothesis function would be a straight line: $h = aT + b$, where a, b are the coefficients that best fit the given data.

The closer the hypothesis h_ξ is to the real value y_i, the smaller the loss function and thus the empirical risk will be. The empirical risk with this loss function is also called the mean squared error.

Before we continue with the algorithm that actually minimises the mean squared error, let us first see how we can cast equation (1) to a linear regression problem. The idea was to match a combination of functions of u to u_t given some set of coefficients c_i that we wanted to learn. Hence, we can identify u_t as y, and the set of functions of u in the $N(u, c_i, x, t)$ as $\vec{z}$.

For convenience we will redifine $\vec{z}$ as $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$, which we will call our library. Here, $\mathbf{U}$ is a vector of the data set, containing $u(x, t)$ for every x and t:

$$\mathbf{U} = \begin{pmatrix} u(x_0, t_0) & u(x_1, t_0) & \dots & u(x_n, t_0) & u(x_0, t_1) & \dots & u(x_n, t_m) \end{pmatrix}, \tag{7}$$

where $x_i \in \mathbb{R}, t_j \in \mathbb{R}^+$ for $i \in [n]$ and $j \in [m]$. The number of available data points for x and t are n and m, respectively. The library can also have terms like $\mathbf{U}^2$, which has the same form as equation (7), but $u(x_i, t_j)$ is replaced by $u^2(x_i, t_j)$. Furthermore, the library $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$ can contain $\mathbf{Q}$, which is a vector that may carry additional information, *e.g.* functions that do not depend on u, such as $\tan(x)$. If we again assume that the terms in $N(u, c_i, x, t)$ only depend on powers of u and its derivatives, then $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$ will have the form

$$\vec{\Theta}(\mathbf{U}, \mathbf{Q}) = \begin{pmatrix} \mathbf{I} & \mathbf{U} & \mathbf{U}^2 & \mathbf{U}\mathbf{U}_x & \mathbf{U}_{xx} & \mathbf{Q}^2\mathbf{U}^3\mathbf{U}_{xxxx} & \dots \end{pmatrix}, \tag{8}$$

where there will be d terms in total and thus d related coefficients $\xi_j \in \vec{\xi}$. Note that in combination with (7) this object is actually a matrix, where each column $\mathbf{\Theta}_j$ contains all of the data of a particular PDE term.

With this we can write the hypothesis and the mean squared error compactly as

$$\mathbf{h}_\xi(\vec{\Theta}(\mathbf{U}, \mathbf{Q})) = \vec{\Theta}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}, \tag{9}$$

$$L(\mathbf{h}_\xi) = \|\mathbf{U}_t - \mathbf{h}_\xi(\vec{\Theta}(\mathbf{U}, \mathbf{Q}))\|_2^2, \tag{10}$$

1	u	u^2	u^3	u^4	u^5	u_x	uu_x
-0.02	0.76	-5.84	16.71	-19.53	7.56	$-4{\cdot}10^{-14}$	$7{\cdot}10^{-13}$
u^2u_x	u^3u_x	u^4u_x	u^5u_x	u_{xx}	uu_{xx}	u^2u_{xx}	u^3u_{xx}
$-6{\cdot}10^{-12}$	$2{\cdot}10^{-12}$	$-3{\cdot}10^{-11}$	$2{\cdot}10^{-11}$	1.05	1.95	10.89	-16.90

Table 1: The first 16 terms and (rounded) coefficients of the PDE found by least squares given the data set of the diffusion equation.

where $\|\cdot\|_2$ denotes the 2-norm. Note that the sum in this specific 2-norm is over all the data points, *i.e.*, over all values of x and t. We also left out the factor of $\frac{1}{nm}$, because it is irrelevant for the steps to come. So L is now just the squared error.

Let us proceed with how to find the best coefficients $\hat{\xi}$. Formally we need to find

$$\hat{\xi} = \underset{\vec{\xi}}{\arg\min}\, L(\mathbf{h}_\xi) = \underset{\vec{\xi}}{\arg\min} \|\mathbf{U}_t - \vec{\Theta}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}\|_2^2, \tag{11}$$

One algorithm that can find $\hat{\xi}$ is the method of least squares [27, p. 124]. The main idea of this method is to simply take the derivative of equation (10) with respect to one of the coefficients ξ_j and set it to zero to find the minimum:

$$\begin{aligned} 0 = \frac{\partial}{\partial \xi_j} L(\mathbf{h}_\xi) &= \frac{\partial}{\partial \xi_j} \|\mathbf{U}_t - \vec{\Theta}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}\|_2^2 \\ &= \frac{\partial}{\partial \xi_j} \sum_{x,t'} \left(u_t(x,t') - \vec{\Theta}(x,t') \cdot \vec{\xi} \right)^2 \\ &= \sum_{x,t'} \left(u_t(x,t) - \vec{\Theta}(x,t') \cdot \vec{\xi} \right) \Theta_j(x,t'), \end{aligned}$$

where $\vec{\Theta}(x,t) = \vec{\Theta}(u(x,t), q(x,t))$. This needs to be done for every coefficient, so in total there will be d equations that need to be solved and we can write this as

$$\sum_{x,t'} \left(u_t(x,t') - \vec{\Theta}(x,t') \cdot \hat{\xi} \right) \vec{\Theta}(x,t') = \vec{0}. \tag{12}$$

This is thus a linear algebra problem and we can solve for $\hat{\xi}$ by rewriting the above equation as

$$\hat{\xi} = \left(\sum_{x,t'} \vec{\Theta}(x,t') \vec{\Theta}^T(x,t') \right)^{-1} \sum_{x,t'} u_t(x,t') \vec{\Theta}(x,t'). \tag{13}$$

Let us now give an example with the diffusion equation. We will use the same data set that generated the plots in figure 1. To obtain this density, we use a finite difference PDE solver in python, called Fipy [28]. In addition, we need to construct a library $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$. We will now define our basis set. It contains terms of the form $u^p(u_{x^k})^q$, where $p \in \{0,1,2,3,4,5\}$, $k \in \{1,2,3,4,5\}$, and $q \in \{0,1\}$. Note that with terms such as u_{x^2}, we actually mean u_{xx}. The reason why we choose this basis set is because it is relatively simple and contains many terms that are present in basic PDEs, such the advection-diffusion equation, the Fisher equation, Burger's equation *etc.*

For the implementation of least squares we have used the `linalg.lstsq` function from the `NumPy` module.[1] In general, this computation is fast. On an Intel(R) Celeron(R) N4020 CPU

[1]The `linalg` submodule uses BLAS software that may use multiple processor cores for the linear algebra computations. If this is undesirable, make sure to restrict the number of cores being used.

@ 1.10Ghz, it takes less than 0.01 seconds to complete. We also normalise each column $\mathbf{\Theta}_j$ of $\vec{\Theta}(\mathbf{U},\mathbf{Q})$ by dividing it by the respective 2-norm $\|\mathbf{\Theta}_j\|_2$.[2]

The results can be seen in Tab. 1. For the sake of clarity, we rounded the coefficients to the second decimal place, except for the extremely small numbers. Clearly this PDE does not represent the diffusion equation, but the result does minimise the squared error: $L(h_{\vec{\xi}}) \approx 0.0074$. However, from this error we do not get a good intuition whether the result is good or not. To obtain a more reliable and interpretative notion of the error we normalise it by dividing the error by the total squared error, which is simply given by

$$L_{\text{tot}} = \|\mathbf{U}_t - \overline{\mathbf{U}}_t\|_2^2, \tag{14}$$

where $\overline{\mathbf{U}}_t$ denotes the mean of $\mathbf{U}_t$. If the normalised error is close to 1, then the algorithm has not done a better job than a linear fit based on the average of the observed data. For the diffusion equation we find that $L/L_{tot} = 8.5 \cdot 10^{-6}$, so the result is much better than a linear fit of the mean; despite being wrong. This is a clear example of overfitting. The result works really well for this data set, but if we were to compute the error with data from another initial condition, then we expect that the error would be bigger if the same coefficients are used.

Nevertheless, we do see that some coefficients are very small. In principle we could try to set these to zero. If this action does not increase the error, then that would be a good indication that those terms are not relevant. This is one of the key ideas behind the `STRidge` algorithm, which we will come back to later. For now we note that standard linear regression can cause overfitting, and that we somehow need to reduce the number of terms. One remedy for this problem is to use ridge regression.

2.3 Ridge Regression

Remember that we assumed that $\vec{\Theta}(\mathbf{U},\mathbf{Q})$ contains independent variables such that each ξ_j is unbiased. But this assumption is of course wrong: most terms in $\vec{\Theta}(\mathbf{U},\mathbf{Q})$ depend on $\mathbf{U}$ and its derivatives, so we have highly correlated data. This correlation will cause the method of least squared to be more imprecise, *i.e.*, the estimated coefficients will be further away from the true coefficients [29]. In addition, changing the data slightly can drastically change the coefficients, meaning that the algorithm is unstable.

To deal with these problems, we can add a regulator term $\lambda\|\vec{\xi}\|_2^2$ with $\lambda > 0$ to equation (11). The idea is to incorporate the bias of $\vec{\xi}$ explicitly into the minimisation problem, which can help guide us to obtain better estimates compared to the coefficients from the least-square method [29]. This type of regression is called ridge regression and the minimisation problem is given by [27, p. 172]

$$\hat{\xi} = \arg\min_{\vec{\xi}} \left(\|\mathbf{U}_t - \vec{\Theta}(\mathbf{U},\mathbf{Q}) \cdot \vec{\xi}\|_2^2 + \lambda\|\vec{\xi}\|_2^2 \right). \tag{15}$$

Similarly to least squares, we can solve this by taking the gradient of the above equation and setting it to zero. From this it follows that $\hat{\xi}$ is given by

$$\hat{\xi} = \left(\lambda I + \sum_{x,t} \vec{\Theta}(x,t)\vec{\Theta}^T(x,t) \right)^{-1} \sum_{x,t} u_t(x,t)\vec{\Theta}(x,t), \tag{16}$$

[2]We found that this makes results from least squares more consistent between Python version 2 and 3. For the example we use Python 3.

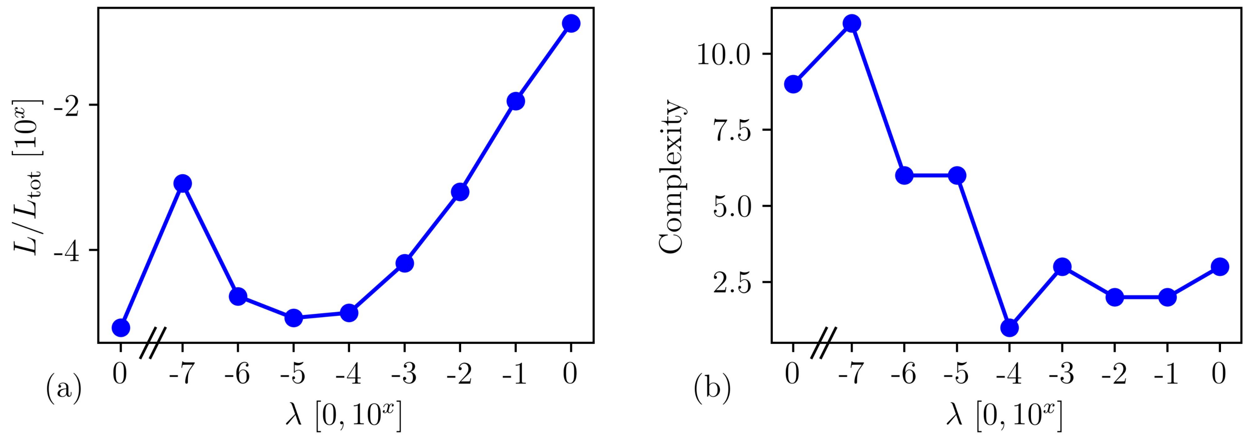


Figure 3: **(a)** We plot the normalised squared error L/L_{tot} of PDEs as a function of the ridge regression parameter λ. This includes $\lambda = 0$ on the left of the slanted lines, *i.e.*, ordinary linear regression. We see that there is a global minimum for $\lambda = 0$ and a local minimum around $\lambda = 10^{-5}$. **(b)** Here we plot (as a function of λ) the complexity, *i.e.*, the number of terms in the PDE, after a cutoff of 0.1 has been applied to the found coefficients. We see that there is a small PDE at $\lambda = 10^{-4}$, which corresponds to the diffusion equation.

where I is a $d \times d$ identity matrix.

The presence of λ has several consequences. First of all, it causes the result to be more stable [27, p. 172]. This means that slight changes of the input $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$ do not cause big changes in the resulting coefficients $\hat{\xi}$. The effect is that the algorithm is less prone to overfitting. The downside however is that with increasing λ, the squared error increases. Thus we need to make a trade-off: we cannot choose λ too small, otherwise we again have overfitting. However, we also do not want λ to be too big, due to the result becoming less reliable. In general there should be an optimum value of λ, *i.e.*, a λ that yields the smallest error, but in practice it is useful to consider a number of values for λ, specifically in the range $0 < \lambda < 1$ [30].

In Fig. 3a we have plotted the normalised error of found PDEs from the diffusion equation data set for different values of λ. We have also included the result for $\lambda = 0$, *i.e.*, standard linear regression. From this picture we see that for $\lambda > 0$ there is a minimum sitting around $\lambda \in \{10^{-5}, 10^{-4}\}$, after which the error keeps increasing. Note that standard linear regression has the smallest error. To obtain an intuition whether the results overfit or not, we have plotted in Fig. 3b the complexity of the resulting PDEs after applying a cutoff of 0.1 on the coefficients. With complexity we mean the number of terms in the PDE after the cutoff. From this we see that the complexity decreases with increasing λ, with a minimum at $\lambda = 10^{-4}$. So for this value we have a relatively low normalised error and we are also not overfitting.

If we examine the resulting PDE at $\lambda = 10^{-4}$ after the cutoff has been applied, we find the diffusion equation: $u_t \approx 0.97 u_{xx}$. This is good news, but we do have to be careful: a low coefficient does not necessarily mean that the contribution is small. If for example the diffusion constant was set to $D = 0.01$, then we would have thrown out the diffusion term, based on the cutoff of 0.1. To find out if it is permissible to remove terms from a PDE, we can compute the squared error with this new PDE, and check if the error has improved or not. If it does, then that is promising. But if it does not, then that is a signal not to remove it. Keeping this thought in mind, we will now examine the `STRidge` algorithm.

2.4 Sparse Regression with STRidge and TrainSTRidge

In the previous sections we have discussed all of the ingredients needed to understand the `STRidge` (Sequential Threshold Ridge regression) algorithm. This is also called a sparse regression algorithm, because it promotes sparsity, *i.e.*, the resulting PDEs tend to become simple/sparse. The input of `STRidge` is a choice of the ridge parameter λ, a threshold/tolerance ξ_{tol} and a training data set.[3] The output is a set of coefficients, where some coefficients are set to zero due to the threshold. Before the algorithm is called, there are a few preliminary steps we need to perform on the given data. These steps are part of the `TrainStridge` algorithm, which we will first explain. After that we will clarify how `STRidge` obtains the coefficients.

The first step is to randomly split the data of $\vec{\Theta}(\mathbf{U}, \mathbf{Q})$ and $\mathbf{U}_t$ with a given ratio into a training and testing set. The standard split ratio is 80% training and 20% testing, which we will adopt as well.[4] The training set is used to estimate the coefficients with `STRidge`. With the results of the training set, we calculate the following error on the testing set:

$$L_{\text{sparse}} = \|\mathbf{U}_{t,\text{test}} - \vec{\Theta}_{\text{test}}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}_{\text{train}}\|_2 + \ell_0 \|\vec{\xi}_{\text{train}}\|_0, \tag{17}$$

where $\|\cdot\|_0$ denotes the 0-norm, so $\|\vec{\xi}_{\text{train}}\|_0$ is simply the number of non-zero terms in $\vec{\xi}_{\text{train}}$. The first term in equation (17) evaluates the accuracy of the trained coefficients $\vec{\xi}_{\text{train}}$, whilst the second term penalises solutions with a high number of non-zero coefficients. Hence, this penalty promotes sparse solutions. The authors of the algorithm from Ref. [14] have chosen to use $\ell_0 = 10^{-3}\kappa(\vec{\Theta})$, with κ the condition number, based on empirical use of the algorithm, and to deal with ill posed problems. There is no other motivation given why the condition number should be used in the penalty. In fact, we found that the condition number has a major flaw, which we will address in the next section. Despite this weak point, we were still able to successfully obtain PDEs with this ℓ_0, so the reader may assume that we use the condition number penalty throughout this thesis, unless stated otherwise.

After splitting the data, we obtain a first estimate of the coefficients using ordinary least squares with $\lambda = 0$. Applying the coefficients to equation (17) provides us with a baseline error, which we can use for comparison with errors calculated with new coefficients.

The next step is to start a loop for a given number of steps, where we begin with an initial tolerance d_{tol}. In the loop we iteratively compute the coefficients with `STRidge`. After each call, the error from Eq. (17) is computed, and is tested against the set of coefficients that has the lowest error L_{best}, called $\vec{\xi}_{\text{best}}$. Initially, $\vec{\xi}_{\text{best}}$ is the set of coefficients from the baseline error. If the new coefficients $\vec{\xi}_{\text{train}}$ satisfy $L_{\text{sparse}} \leq L_{\text{best}}$, then the best error and coefficients are updated accordingly. In addition, the tolerance is increased:

$$\xi_{\text{tol}} = \xi_{\text{tol}} + d_{\text{tol}}.$$

In general, a higher tolerance will yield a more sparse PDE, such that $\|\vec{\xi}_{\text{train}}\|_0$ becomes lower, which decreases L_{sparse} as well. However, if $L_{\text{sparse}} > L_{\text{best}}$, then the constructors of the algorithm assume that highly contributing terms to the PDE are set to zero, because the tolerance is too

[3]In the original code there is a mistake, where they use the entire data set for error evaluation instead of the training set. This issue is fixed as of March 23, 2022.

[4]Be aware that the code to randomly split the data is aided with a python `for-loop`, which becomes inefficient for large data set ($> 10^6$ data points). We advice to change this method with, *e.g.*, sets, to avoid wasting time.

big. In that case, they decrease the tolerance in the following manner:

$$\begin{aligned}
&(1)\quad \xi_{\text{tol}} = \max(0, \xi_{\text{tol}} - 2d_{\text{tol}}),\\
&(2)\quad d_{\text{tol}} = \frac{2d_{\text{tol}}}{N_{\text{iter}} - \text{iter}},\\
&(3)\quad \xi_{\text{tol}} = \xi_{\text{tol}} + d_{\text{tol}},
\end{aligned}$$

where iter is the latest step of the loop. In short, they simply take two steps back, decrease the step size and try again. The effect of this is that ξ_{tol} converges below the coefficient that the algorithm refuses to cut away. After the N_{iter} steps are done, the output of `TrainStridge` is simply $\vec{\xi}_{\text{best}}$.

We will now explain the specifics of the `STRidge` algorithm implementation. The first step of this algorithm is to normalise each column of $\vec{\Theta}_{\text{test}}(\mathbf{U}, \mathbf{Q})$ to unit variance as before. In this case there is another reason for doing this, which we will clarify shortly. After the normalisation, ridge regression with the chosen λ is applied to determine the first set of coefficients. We then apply a hard threshold: each coefficient that satisfies $\xi_j < \xi_{\text{tol}}$ will be set to zero.

Now, remember the problem we outlined in the previous section: a small coefficient does not necessarily mean that the contribution to the PDE is small. By applying a normalisation on each column of $\vec{\Theta}_{\text{test}}(\mathbf{U}, \mathbf{Q})$, we basically treat every term on equal footing; the calculated coefficients ξ_j scale according to $||\mathbf{\Theta}_{j,\text{test}}||_2$. This means that we actually apply a threshold on the total contribution to the PDE and not on the coefficient itself. This allows us to find PDEs that have a combination of small and large coefficients in front of the actual terms.

After the coefficients are set to zero, we again compute the coefficients with ridge regression, but now with only the remaining non-zero terms from the previous round. This iterative procedure is repeated for a given number of steps. Finally, after the recursive call is complete and the sparsity pattern is found, ordinary least squares is performed to obtain the final estimates of the coefficients. The output of `STRidge` is then simply the final coefficients multiplied by their respective normalisations.

We found that the algorithm is relatively fast, because it mainly uses least squares. For example, if we use 100 `TrainSTRidge` iterations, with random walker data of size 500×600, `TrainsTRidge` finishes on average in 70 seconds with an Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60Ghz.

For more information about the details of both algorithms see the appendix of Ref. [14]. In the next part we will discuss a number of weak points we found in the algorithm, and how we changed and adapted the methods to overcome these issues.

2.5 TrainSTRidge Adaptions

In this section we will propose and outline alternative methods that improve the reliability of the `TrainSTRidge` algorithm regarding its capacity to find a balance between sparse and accurate PDEs. Before we move on to the adaptions, let us first state what the problem is. To validate the sparse regression algorithm on lattice particle systems, we analysed the 1D non interacting biased random walk, which models particle movement with a bias towards a certain direction (see section 3.3 for more information). It is well known in literature that the continuum limit of this PDE is the advection-diffusion equation, which is a linear combination of diffusion (u_{xx}) and advection (u_x). In our studies of this system, we found that the output PDE of `TrainSTRidge` depended on the size of the basis set $\vec{\Theta}$. With a small basis set, we found the expected advection-diffusion equation. However, when we added more terms in the basis set, the output PDE only contained the advection term. We found that this is due to the error penalty scaling $\ell_0 = 0.001\kappa(\vec{\Theta})$.

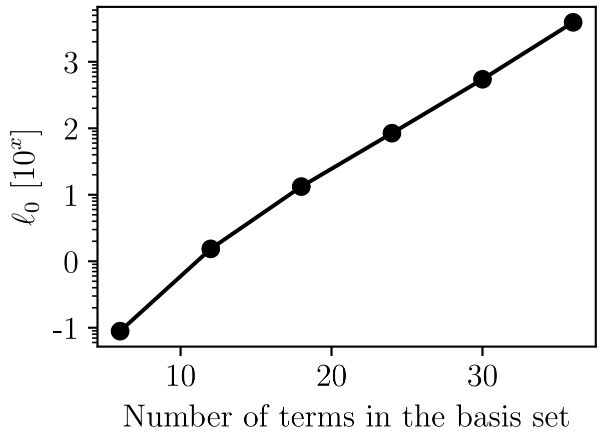


Figure 4: We plot the error penalty scaling $\ell_0 = 0.001\kappa(\vec{\Theta})$ as a function of the number of terms in the basis set $\vec{\Theta}$. This includes terms such as $u^p(u_{x^q})^k$, with $q \in \{1, 2, 3, 4, 5\}$, $k \in \{0, 1\}$, and we increases p from 0 to 5 with steps of 1. To compute $\vec{\Theta}$ we have used a filtered density field obtained from a biased random walk particle system. We see that ℓ_0 grows exponentially in the number of terms we add to the basis set. Hence, the ability to promote sparsity of a PDE with `TrainSTRidge` heavily depends on the size of the basis set.

In Fig. 4 we plot ℓ_0 as a function of the size of the basis set. We see that ℓ_0 increases exponentially if we add more terms to $\vec{\Theta}$. Hence, if we have a large basis set, the error penalty term in Eq. (17) becomes much larger than for a smaller basis set, which increases the bias of the algorithm towards sparse PDEs. As a side note, we suspect that this problem is similar to the reported issues regarding the biased random walk of a Brownian particle in the appendix of Ref. [14] from Brunton *et al.* Here, they found that applying `TrainSTRidge` with the normalisation of $\vec{\Theta}$, resulted in the diffusion equation instead of the advection-diffusion equation, *i.e.*, the bias towards sparse PDEs was too high. Turning the normalisation off did result in the correct PDE, as it circumvents the bias in the error. However, we think that the normalisation is an important aspect of the algorithm as it ensures detection of relevant contributions with (relatively) small coefficients. As such, we wish to combat the bias in the error due to ℓ_0, while still using a normalisation. To achieve this, we are actually going to make use of the physical information obtained by normalising PDE contributions.

2.5.1 Physics Based Threshold for TrainSTRidge Tolerance Increment

In Fig. 5a we are bar plotting the relative PDE contributions $\|\xi_j\Theta_j\|_2/\|\vec{\xi}_{\text{base}}\cdot\vec{\Theta}\|_2$ obtained after computing the coefficients $\vec{\xi}_{\text{base}}$ of the standard basis set with least squares and $\lambda = 10^{-3}$. Here, the data is generated from a biased random walk on a lattice. We see that each contribution is between 0 and 1, implying there is no large overfitting with large contributions ($\gg 1$) that cancel each other out. This is due to ridge regression with $\lambda = 10^{-3}$. If we would have used a lower λ, we would obtain much larger contributions.

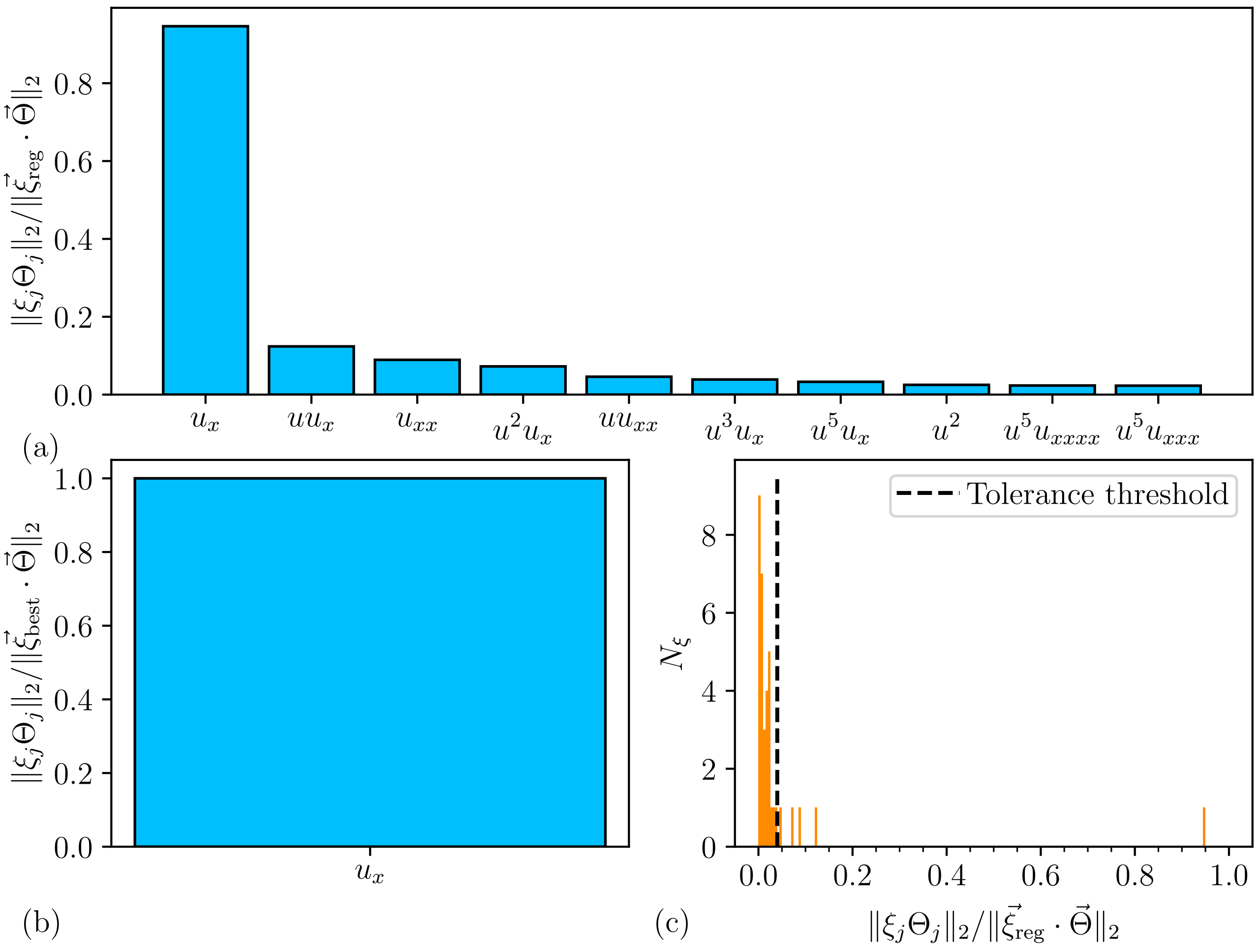


Figure 5: **(a)** Bar plot of relative PDE contributions $\|\xi_j\Theta_j\|_2/\|\vec{\xi}_{\text{reg}}\cdot\vec{\Theta}\|_2$ of terms $\vec{\xi}_{\text{reg}}$ computed with standard ridge regression for $\lambda = 10^{-3}$ on biased random walker data. This system is a non-interacting particle systems, which has both diffusive (u_{xx}) and advective (u_x) properties, see section 3.3 for more information. We plot the ten biggest contributions, sorted from high to low. The x-axis shows the corresponding term of the bar. The highest term corresponds to advection and is much larger than all the other contributions. **(b)** Similar plot to (a), but now for resulting terms from the `TrainSTRidge` algorithm with learn parameter $d_{\text{tol}} = 1$, 100 `TrainSTRidge` iterations and $\lambda = 10^{-3}$. We see that only the advection term is found. Hence, we want to make sure that `TrainSTRidge` does not cut away the diffusion term by introducing a tolerance threshold. **(c)** We make a histogram of the PDE contributions found with ridge regression. We use 201 bins in the range $[0, 1]$, where N_ξ is the count per bin. The tolerance threshold is computed by finding the first gap in the histogram, *i.e.*, the first bin of height zero.

In addition, we see that u_x has the highest contribution, while u_{xx} has a much lower contribution. From this we can deduce that the dynamics of the biased random walk is dominated by advection. We thus see that we can already learn a lot about the system by ridge regression alone. This also gives us intuition why the only surviving term of `TrainSTRidge` with a large basis set is the advection equation. This can be seen in (b), where we bar plot the contribution of the output terms of `TrainSTRidge`.

Despite that u_{xx} has a low contribution, it is still the third highest, *i.e.*, there are many more terms that have a lower contribution. These terms are so small that they cannot be distinguished from each other. In Fig. 5c we quantify this observation by making a histogram of the contributions, *i.e.*, we count between 0 and 1 the number of contribution that are within a certain interval. For this we used 201 bins. We see that the majority of terms have a relatively low contribution, which results in a large solid peak. The terms that belong to this 'noise' peak are indistinguishable from each other, and we can assume that they do not contain important information about the dynamics. As such, if we were to remove any of these terms from the basis set, we would still be able to fit the data, provided that the underlying physics is important enough to give distinguishable contributions. That is, we do not want to blindly throw away terms that are located to the right of this peak, as we would potentially lose important physical aspects of the dynamics. Hence, we should be alarmed if `STRidge` sets a coefficient to zero of a term that has a higher contribution than this peak.

To combat the bias towards sparse PDEs, we thus introduce a tolerance threshold: if we are at a `TrainSTRidge` tolerance for which a (normalised) contribution in `STRidge` is cut away that is above the noise peak, we permanently set ℓ_0 to zero, compute the new error of the current and best coefficients with equation (17), and continue to find the optimal tolerance and PDE according to $L_{\text{sparse}} \leq L_{\text{best}}$. This ensures that no important physical contributions are cut away, since only the accuracy of the sparse PDE will determine the optimal threshold. We compute this tolerance threshold before the start of the `TrainSTRidge` algorithm, and it is simply set to the contribution of the first histogram bin from left to right with zero counts.

Using this method, we managed to find the advection-diffusion equation, regardless of the size of the basis set. If the reader is interested in an example result of this method, he/she can jump to Fig. 22 in section 3.3.

2.5.2 Normalising the TrainSTRidge Error Evaluation

Using the above method, we circumvent the issue regarding the penalty term scaling in the error from equation (17). However, there is no guarantee that this method is alway applicable, since it assumes that we can regularise a large basis set with ridge regression, which may not always be the case. Hence, we will propose a replacement of the `TrainSTRidge` error in Eq. (17). It is based on the observation that we need to have a fair competition between accuracy and sparsity without bias. As such, the accuracy and penalty must be on equal footing, *i.e.*, it is necessary that they are roughly of the same order of magnitude. The simplest way to perform this, is by setting $\ell_0 = 1$ and normalising the accuracy:

$$L_{\text{train}} = \|\mathbf{U}_{t,\text{test}} - \vec{\Theta}_{\text{test}}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}_{\text{train}}\|_2^2, \tag{18}$$

$$L_{\text{base}} = \|\mathbf{U}_{t,\text{test}} - \vec{\Theta}_{\text{test}}(\mathbf{U}, \mathbf{Q}) \cdot \vec{\xi}_{\text{base}}\|_2^2, \tag{19}$$

$$L_{\text{sparse}} = \frac{L_{\text{train}}}{L_{\text{base}}} + \|\vec{\xi}_{\text{train}}\|_0, \tag{20}$$

where $\vec{\xi}_{\text{base}}$ is the set of coefficients computed on the enitre basis set. That is, we normalise the accuracy term with the baseline error.

In general, the basis set will (usually) overfit the data, and that decreasing the number of terms will give a slightly worse overfit. In that case, $L_{\text{train}}/L_{\text{base}}$ will become slightly bigger than 1, while $||\vec{\xi}_{\text{train}}||_0$ will decrease by 1 or more. This results in $L_{\text{sparse}} < L_{\text{best}}$. Only if a relevant contribution of a sparse PDE is cut away, we will have that $L_{\text{train}}/L_{\text{base}} \gg 1$, such that $L_{\text{sparse}} > L_{\text{best}}$. In addition to this error evaluation, we will always increase ξ_{tol} without changing d_{tol}, even if $L_{\text{sparse}} > L_{\text{best}}$. This guarantees that we do not get stuck in a local minima. Of course, we only update L_{best} if $L_{\text{sparse}} < L_{\text{best}}$.

We have tested this error evaluation on each of the available data sets that were provided by Brunton *et al.* on the PDE-FIND Github page of Ref. [14], and similar to the original `TrainSTRidge` algorithm, we found the corresponding PDE of each data set. We will primarily use this error evaluation for the interacting particle systems in section 4.

2.6 Filtering Noise with 2D Fourier Transformations

In the previous sections we have examined the machine learning aspects of the data-driven methods. In this part we are going to consider how to handle the noise we will inherit from the particle data that we will generate. See Fig. 6a for an example of noisy particle data. In general, it is hard to approximate a derivative of noisy data with for example finite differences (FD). The reason for this is that with FD, we calculate the slope by simply considering the change of the function (around a fixed point) over the lattice spacing. If the data is noisy, then there is an offset at each point. Hence, the slope around each point will be mostly based on this noise offset, which means that the derivative will be spiky. We thus need to smoothen the data before we take any derivatives.

We shall do this by using a filter based on a 2D (discrete) Fourier transformation (DFT) of the data $u(x,t)$, which we perform with a fast Fourier transformation (FFT) [31]. By going into Fourier space, we can separate the noise signal from the underlying signal by looking at the amplitudes. Provided there is relatively little noise, the noise frequencies will have a lower amplitude than the dominant signal. Hence, we can simply set the amplitudes of the noise frequencies to zero via a threshold.

In Fig. 6a we plot an example of a noisy signal obtained by ensemble averaging lattice particle occupations of the Eden-SSEP model (see section 4.1). With the FFT we can compute the 2D Fourier transformation of this signal. In (b-c) we scatter plot a of the amplitudes of the obtained frequencies as a function of the temporal and spatial frequencies, where we have projected the spatial and temporal frequencies on top of each other, respectively. We see that there is a clear separation between the underlying signal and the noise frequencies. Hence, we can estimate the height of the amplitude cutoff A_{cut} that we want to apply to set the frequencies below this threshold to zero. In Fig. 6d we zoom into the noisy density plot from (a), where we now also plot the filtered signal. We see that the filter successfully removes the noise signal. However, at $t = 0$ there is a slight discrepancy between the filtered signal and the original signal. This is a side-effect of the filtering process, due to the fact that we also remove information that is hidden in the layers of noise. Increasing A_{cut} would enhance this effect. Hence, we need to be careful to not include biased filtered data into the outlined regression algorithms. This is simply done by cutting away the temporal boundary regions.

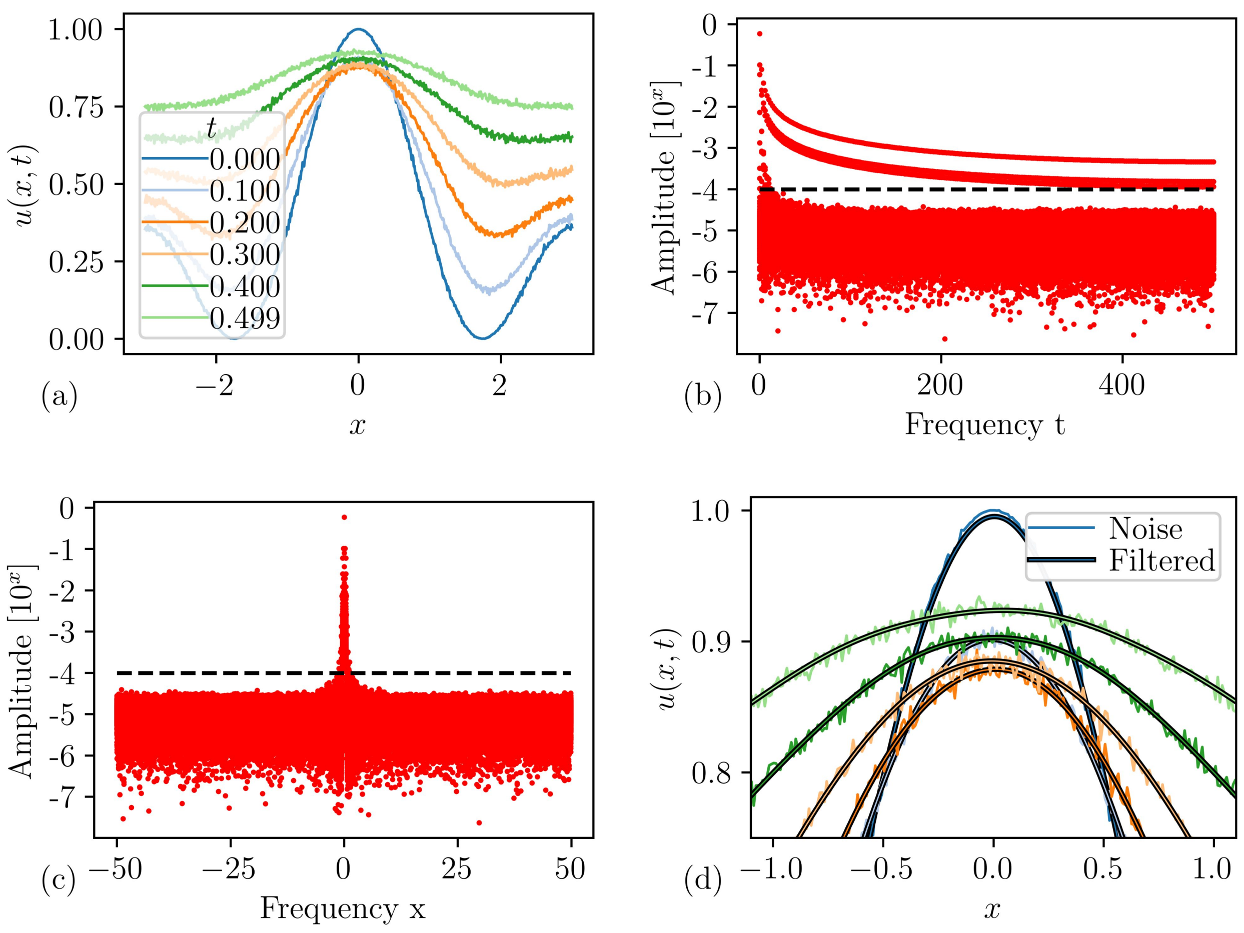


Figure 6: **(a)** A noisy signal of the Eden-SSEP particle system that we need to filter. Here we plot the density of the particles $u(x,t)$ as a function of space x for different instances in time t. The relevant spatial and temporal dimensions are left out for the sake of clarity. More information about this system can be found in section 4.1. **(b)** We show a side-view scatter plot of the 2D DFT of the noisy signal from **(a)**, where we plot the amplitude on the y-axis as a function of the temporal frequencies for each spatial frequency. **(c)** is similar to **(b)**, but with the spatial frequency on the x-axis. The black dashed line represents the amplitude cutoff A_{cut}, which we have set to 10^{-4}. We set every amplitude below this threshold to zero. The inverse of the thresholded Fourier transform can be seen in **(d)**, where we compare the noisy signal to the filtered signal. The axis and colour schemes are the same as in **(a)**, and we have also zoomed in to see the differences more clearly. The filtered signal is smooth and matches the general form of the noisy signal very well, except at $t = 0$, where there is a miss-match at the extremum. Hence, for `TrainStridge` we will avoid temporal regions where the filter introduces a bias.

2.7 Computing Derivatives with the Savitzky-Golay Filter

Once the worst noise has been filtered out, we want to compute the derivatives that we want to insert into our basis set. In the paper from Brunton *et al.* [14], they considered polynomial differentiation (PD) to be the most reliable method to estimate derivatives from noisy data [32]. The idea behind PD is that at each datapoint p, a polynomial of degree d is fitted around this point for $p-n/2 \leq j \leq p+n/2$, where n is the width of the fit. The derivative of that polynomial at p is then taken as the approximate derivative of the numerical data. The downside of this method is that the result may highly depend on the choices of d and n. Furthermore, the approximations at the boundary can be inaccurate, so they remove n points at the boundaries. Hence, a lot of relevant information can be lost if a high width is required to obtain accurate estimates.

We are going to use periodic boundary conditions for the interacting particle systems. So in this case there is no need to toss away points at the boundary, because we can use the periodicity to fit a polynomial through the boundary. Hence, we will use a polynomial fitting tool that can implement periodic boundary conditions. For this we will use the Savitzky-Golay (SG) filter that is available on Scipy version 1.8.1, which has a periodic boundary condition option. This filter uses simple least square fits with polynomials to smoothen and differentiate data [33]. Similar to PD, the input requires a degree d and a width n.[5] For future examples we will denote the width in terms of the spatial size $x_\mathrm{w} = n\Delta x$, with Δx the lattice spacing.

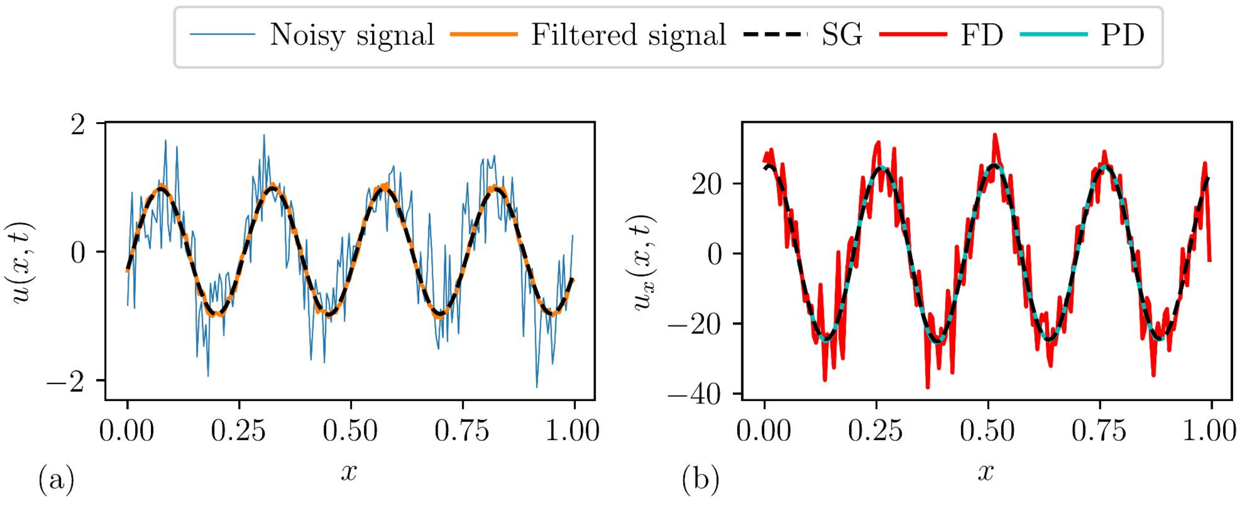


Figure 7: **(a)** We plot a travelling sine wave $u(x,t)$ with added white noise of $\sigma = 0.5$ as a function of space x. We use periodic boundary conditions, we filter the signal with the 2D amplitude filter, and then apply a SG-filter to estimate the curve with a polynomial. **(b)** We compute the first derivative $u_x(x,t)$ using multiple techniques and we plot as a function of space. Finite differences (FD) yields a noisy derivative, despite the filter. The polynomial differentiation (PD) scheme from Ref. [14] is able to get a good estimate of the derivative, at the cost of having to throw away data points at the boundary. The Savitzky-Golay (SG) filter produces the same output, but the option to use periodic boundary conditions also gives accurate estimates at the boundary. Hence, the SG filter is the most useful differentiation scheme of the three tested methods.

To show how the SG-filter works, we apply it on a travelling sine wave with gaussian noise and periodic boundary conditions. In Fig. 7a we plot the sine function, the filtered signal (from the

[5]Be aware that in general (including the Scipy version), the width n should be an odd integer number.

2D FFT filter), and the SG-fit onto the filtered signal as a function of space. Note that the SG-filter smoothens the filtered signal even further, while also accurately fitting the boundaries. In (b) we plot the first spatial derivative of the filtered signal using different differentiation methods, namely SG, PD and FD. We see that FD yields spiky derivatives, despite the 2D FFT filter. PD and SG have similar outputs, but SG also has an accurate estimate at the boundaries, while PD tosses the data away. Hence, for the tested differentiation schemes, the SG-filter is the most promising tool.

Now, if the SG-filter can also smoothen the data, why would we go through the trouble of filtering the data with the 2D FFT filter? There are various arguments to implement the FFT filter. First of all, the 2D FFT filter also smoothens the temporal direction in combination with the spatial direction. This will make sure that the filtered signal is a smoothened field, instead of individual slices that could be discontinuously connected to each other, if we were to use 1D filters only. In addition, we found that higher order SG derivatives are more accurate after application of the 2D FFT filter.

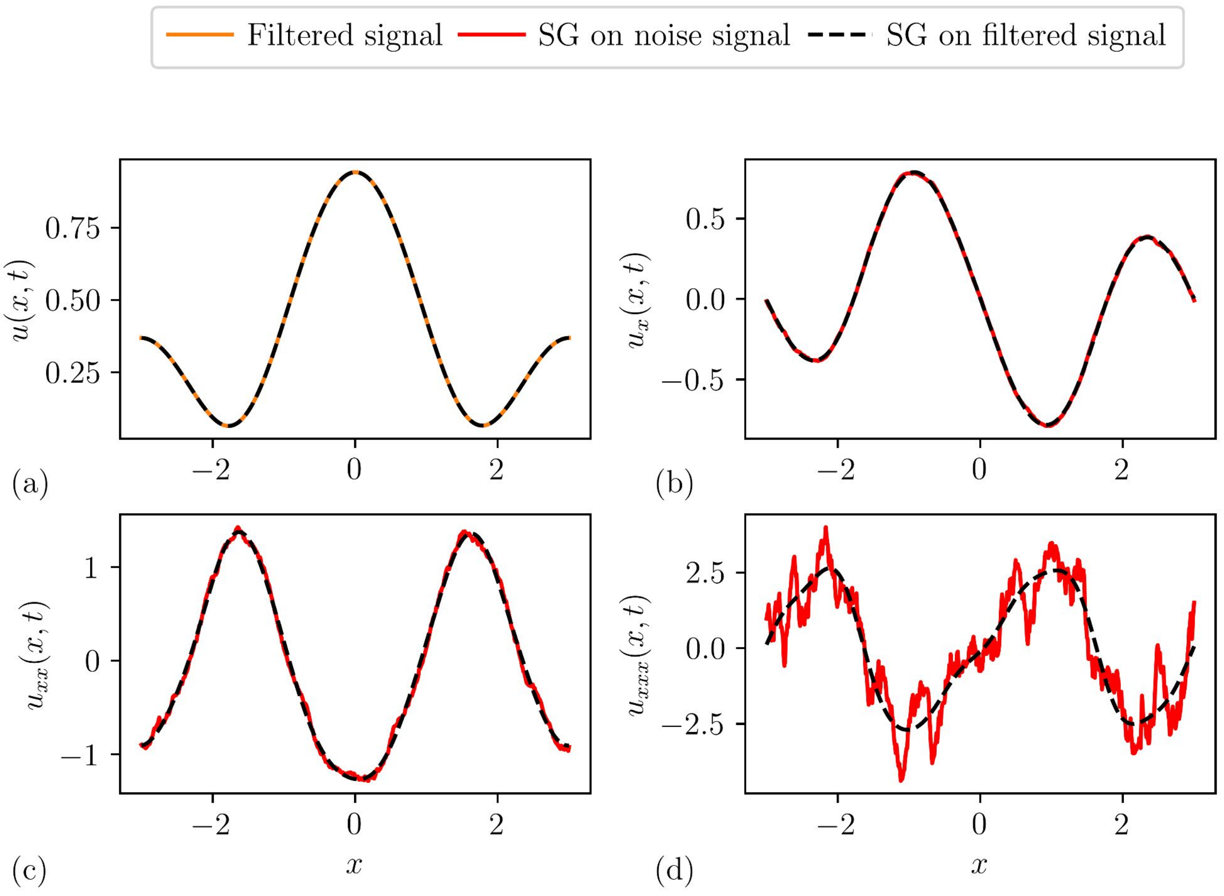


Figure 8: **(a)** We plot the 2D FFT filtered signal of the density from Fig. 6a as a function of space x, and with $A_{\text{cut}} = 10^{-4}$. On the filtered signal we fit a polynomial using the SG-filter with width $x_{\text{w}} = 1.01$. We see that the fitted polynomial matches the filtered signal very well. **(b-c-d)** Estimate of the first, second and third derivative of the density as a function of space x, using the SG filter on the noise signal (red), and on the filtered signal (black). The noise becomes more amplified as for the SG-fit on the noise signal as the derivatives become higher. The SG derivatives on the filtered signal stay smooth. We thus see that the combination of filtering and SG fitting is crucial to obtain smooth higher order derivatives.

In Fig. 8a we plot the 2D FFT filtered signal and the SG-fit onto the filtered signal as a function of space for the same density as in Fig. 6a. In (b-c-d) we plot the first, second and third derivative of the 2D FFT filtered, and the noisy signal, both using the SG-filter with the same parametric input. We see no apparent difference in the first derivative. For the second derivative we see that the SG fit on the noise signal becomes more unwieldy than the fit using the 2D FFT filter. For the third derivative the SG fit on the noise signal is spiky and noisy, while the other fit stays relatively smooth. Hence, we see that combining the 2D FFT filter with the SG-filter is essential to compute higher order derivatives for the basis set.

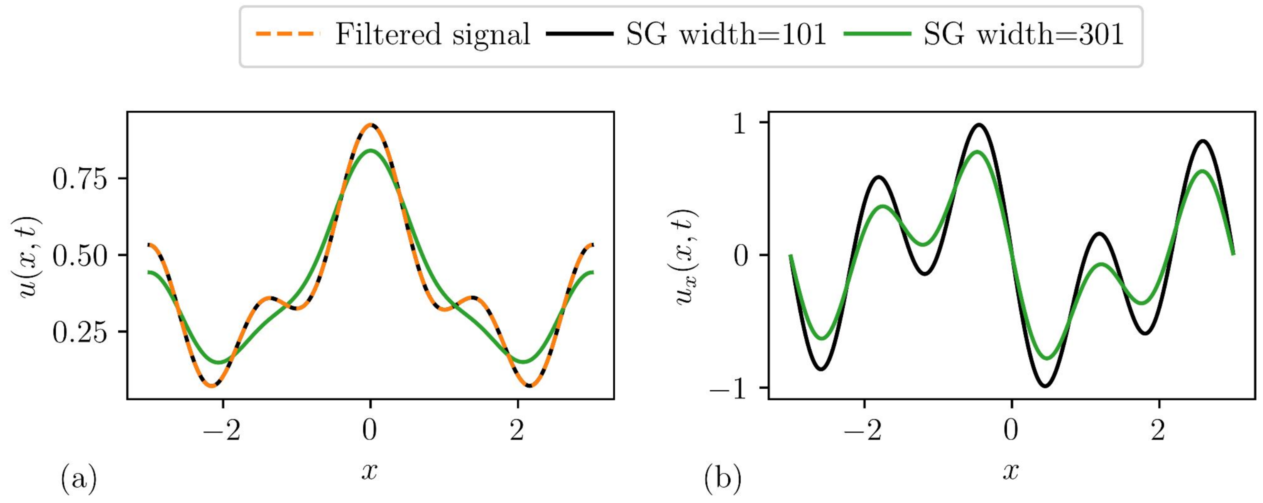


Figure 9: **(a)** We plot the 2D FFT filtered density $u(x,t)$ obtained from the Eden-SSEP system as a function of space x at a certain time step. On top of the signal we fit a SG polynomial with width $x_\mathrm{w} = 1.01$ (black) and $x_\mathrm{w} = 3.01$ (green). The black line matches the filtered signal, whilst the green line does not. **(b)** We plot the first derivative $u_x(x,t)$ as a function of space x, using the SG-filter with the same widths as above. We see that the extremes of the large width are smaller than those of the small width. Hence, choosing a width that is larger than the typical length scale of $u(x,t)$ leads to estimates that do not fit high frequency modes, which yields inaccurate derivatives.

Finally, we would like to address one final point regarding the SG-filter. Similar to PD, there is a dependency on the accuracy of the fit based on the input parameters. A small width could lead to noisy derivatives, but if the width is too large, the polynomial will not be able to capture all of the local gradients of the signal. In Fig. 9a we plot a 2D FFT filtered signal of a density with many gradients. We apply a SG-filter on this signal with $x_\mathrm{w} = 1.01$ and $x_\mathrm{w} = 3.01$, both with degree 5. The small width fit sits on top of the filtered signal, while the large width cannot capture local gradients. In (b) we show the consequence of this offset by plotting the first derivative of the signal. We see that the large width fit has less steep estimates of the gradient than the small width. Hence, if we are going to use multiple initial conditions with varying gradients, we should be careful to select a SG-width that does not bias the derivative of each of those signals.

2.8 Finding Consistent and Reliable Partial Differential Equations

In the previous part we have considered all the tools needed to find PDEs from a single (noisy) data set $u(x,t)$. In this section we will outline a few important aspects regarding the reliability of found PDEs.

The first important step of a data-driven method is of course to collect representative data $u(x,t)$. With representative we mean that $u(x,t)$ should be within the physical regime of interest, *i.e.*, a space and time domain with enough temporal and spatial gradient changes within the system. This will make it easier to distinguish relevant contributions with least squares and find the PDE.

In addition, it is useful to consider multiple initial conditions, because we are then able to cross check results. If a PDE has coefficients that are not consistent across multiple initial conditions, *e.g.*, there are sign swaps or differences that are much larger than the computed standard deviations, then that would be a sign that the PDE is incorrect and/or not general. Usually, we will consider around 4 initial conditions with various gradients. More is always possible, but we found that a small number is already enough to check the consistency of a PDE.

This brings us to the next point: to compare coefficients and other properties of the PDEs, we need to introduce some form of statistics. This is important, because it allows us to compare our results from the interacting particle systems to existing literature. For this we will make use of the stochastic nature of the particle rules: a different random-number-generator (rng) seed will result in different fluctuations around the signal. Hence, we can generate $u(x,t)$ with multiple rng seeds, apply least squares and/or `TrainSTRidge`, and obtain multiple coefficients and squared errors that we can average. This process is not possible for smooth data obtained from solving a PDE. In that case, we will apply least squares on found sparse PDEs for a different number of time- and space-slices of the sparse library $\Theta(x,t)$. For example, if we initially have data for $x \in \{x_0, x_1, x_2, x_3, ...\}$ then the next step we will use $\Theta(x,t)$ with $x \in \{x_0, x_2, x_4, ...\}$ and the one after that we will use $x \in \{x_0, x_3, ...\}$ *etc.* For each slice we compute the coefficients and in the end we will calculate the average $\bar{\xi}$ and standard deviation σ over all the slices.

The above described measures allow us to ascertain whether a PDE is correct/valid and general. In the next section we are going to combine these considerations with the `TrainSTRidge` algorithm and show if we can reliably find PDEs on smooth data, and data from non-interacting particle systems.

3 Validation of TrainSTRidge and Testing Data-Driven Techniques

The aim of this section is to validate the data-driven methods we have outlined in the first part of this thesis. To this end, we are going to apply the techniques on relatively simple systems. This allows us to ascertain whether we can also employ them for more complex systems. We will start with smooth data generated by the Fisher equation. After that we will consider non-interacting particle systems, namely the random walk and the biased random walk (on a lattice). This should give us valuable insight into whether we can also tackle interacting particle systems.

3.1 The Fisher Equation obtained from Smooth Data

The first example we will consider is the Fisher equation. We know that the continuum limit of the Eden-SSEP particle system dynamics gives the Fisher equation. As we will see below, this PDE is a simple extension of the diffusion equation. Hence, it is a natural choice to first test the data-driven methods on.

The Fisher/Fisher-KPP equation was independently proposed by R. Fisher [19] and Kolmogorov *et al.* [34] in 1937. Fisher aimed to model the spread of advantageous genes of a population that is uniformly distributed in a linear habitat. He supposed there should be a local spread, *i.e.* diffusion, and a wave $u_t = -vu_x$ of increasing gene frequency u as new generations are born. For this end, he proposed the following equation

$$u_t = Du_{xx} + ru(1-u), \tag{21}$$

where u can be interpreted as a density of a certain population. The first term on the right-hand side is ordinary diffusion, with D the diffusion constant. The factor ru in the right-hand side can be considered as a growth term and $1-u$ slows down this growth as $u \to 1$.

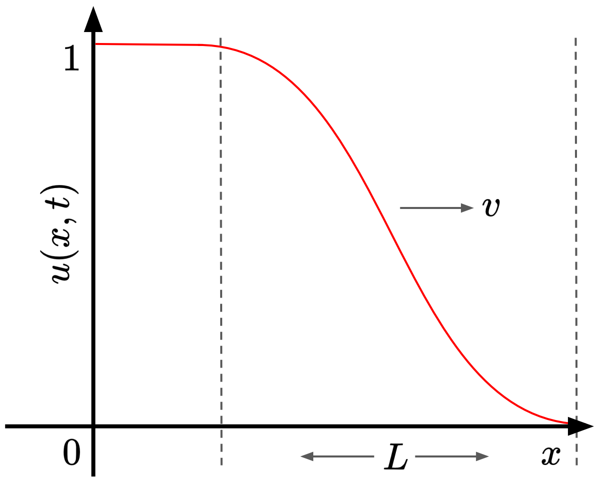


Figure 10: We sketch a travelling wave solution of the Fisher equation. On the y-axis we have the population density $u(x,t)$, which is between 0 (empty) and 1 (full). We plot this as a function of space x. The wave has a leading order width $L = 2\sqrt{D/r}$ and is travelling with a speed $v = 2\sqrt{Dr}$. The travelling wave motion represents a population growing in density at the boundary of its domain. This is due to its growth rate r, and its diffusive properties with diffusion constant D.

Let us now explain the behaviour of the Fisher equation solutions. First of all, there are two steady states: one at $u = 0$ (unstable) and the other at $u = 1$ (stable) [35]. This means that

D	0.01	1.0	5.0
r	100	1.0	0.2
x/L	$[0, 50]$	$[0, 0.5]$	$[0, 0.1]$
$\Delta x/L$	$2 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	10^{-4}
t/τ	$[0, 10]$	$[0, 0.01]$	$[0, 2 \cdot 10^{-5}]$
$\Delta t/\tau$	10^{-3}	10^{-7}	$2 \cdot 10^{-10}$
$\Delta t_{\text{reg}}/\tau$	10^{-2}	10^{-5}	$2 \cdot 10^{-8}$
Initial conditions	$0.1 \exp(100(x-0.5)^2)$ $0.1 \cos^2(2\pi x)$ $\exp(50x^2)$ 0.1	$\exp(100(x-0.5)^2)$ $\cos^2(2\pi x)$ $0.5 \cos^2(4\pi x)$ 0.1	$\exp(100(x-0.5)^2)$ $\cos^2(2\pi x)$ $0.5 \cos^2(4\pi x)$ 0.1

Table 2: Initial conditions and parameters for obtaining solutions of the Fisher equation. We make the parameters for space and time dimensionless by dividing by L and τ, respectively.

$u \to 1$ as $t \to \infty$. The factor $r > 0$ in front of these terms is the growth rate, which has an inverse time dimension T^{-1}. As such, the relevant time scale of this system is $\tau = r^{-1}$. The larger the value of r, the faster u goes to 1.

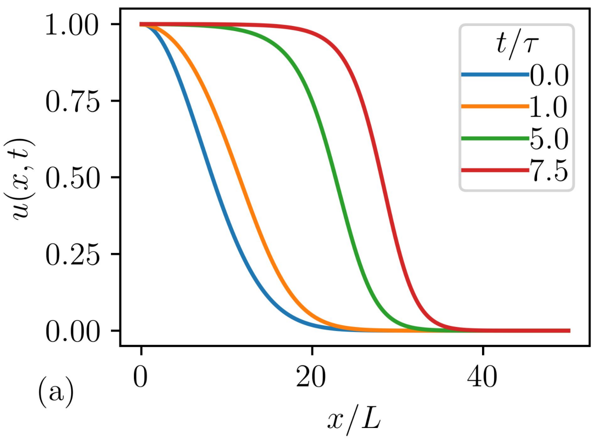
Furthermore, the Fisher equation has a travelling wave solution, where the length scale $L = 2\sqrt{D/r}$ can be interpreted as the width of the wavefront and $v = L/\tau = 2\sqrt{Dr}$ as the leading order speed, see Fig. 10 for a sketch. The travelling wave is one of the main features of the Fisher equation and has been found in several areas of physics, biology and chemistry. For example, the wave solution of the Fisher-KPP matches the frontier of branching Brownian motion [36]. In addition, the validity of the Fisher equation has been tested experimentally to the growth of bacterial colonies [37]. From both examples we see the relationship between growth and diffusion, and travelling waves of population densities.

There are also other instances of the Fisher Equation used in physics. This includes flame propagation after an exothermic reaction [38], and also the spread of a neutron population in a nuclear reactor [39]. See Ref. [35] for a complete overview of the physics of wavefronts in a more general setting.

To solve the PDE, we will use `Fipy`. We will consider three regimes: dominant growth, balanced, and dominant diffusion. For this we will consider the following parameters: $D \in \{0.01, 1.0, 5.0\}$ with $v = 2.0$, so $r \in \{100, 1.0, 0.2\}$. With these different regimes we want to test if we can still find the correct PDE, despite the large differences in the order of magnitude of the different coefficients. We expect that this is not a problem, considering the normalisation in `TrainSTRidge`.

For each of the different parameter sets we have 4 different initial conditions, consisting of cosines, gaussians and straight lines, see Tab. 2 for the exact functional forms. In the table we also show the parameters used for generating the data with `Fipy`. In general the solutions were stable, provided we used small time step sizes. In addition, we found that a higher diffusion requires smaller time step size. This makes sense, because the diffusion equation ordinarily needs to be solved with $\Delta t \leq \Delta x^2/(2D)$.[6] In Fig. 11 we show two examples of stable solutions of the PDE. Both solutions are within the physical regime of interest, *i.e.*, we probe distinguishable growth and diffusive behaviour that allows us to capture the Fisher equation.

[6]To compute the derivatives for the basis set, we will use a higher temporal resolution than we used for obtaining the solutions, such that we use 1000 temporal data points. This does not pose a problem regarding the stability of the derivatives, as the spatial and temporal derivatives are computed independently from each other.



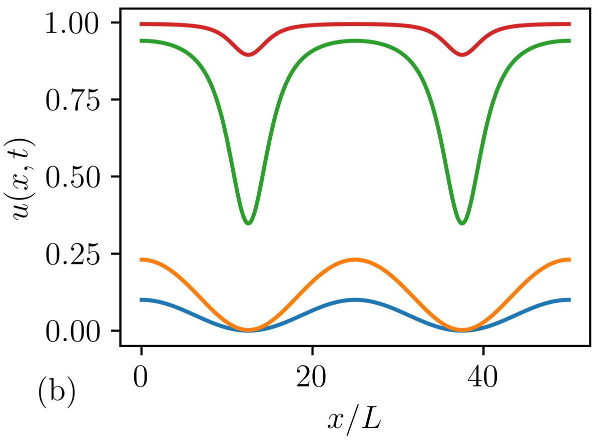
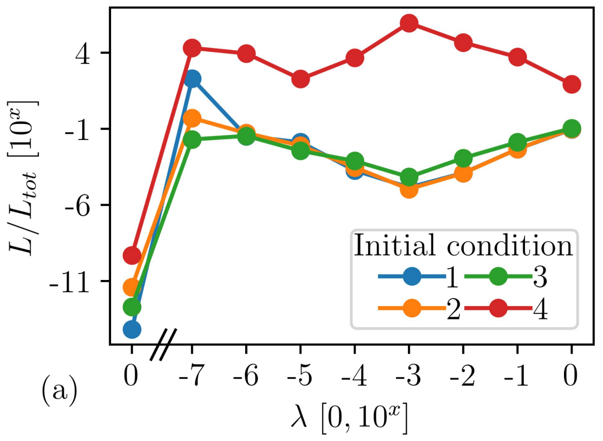


Figure 11: We plot the population density $u(x,t)$ obeying the Fisher equation with $D = 0.01$ and $r = 100$ as a function of dimensionless space x/L for different instances in time. Here, $L = 2\sqrt{D/r}$ and $\tau = r^{-1}$ are the space and time scales of the system. **(a)** Travelling wave solution of the Fisher equation. Diffusion and growth will give rise to a travelling wave at constant speed, which can be interpreted as population migration. **(b)** Cosine initial condition converges to the steady state of $u(x,t) = 1$, where the population has reached carrying capacity.



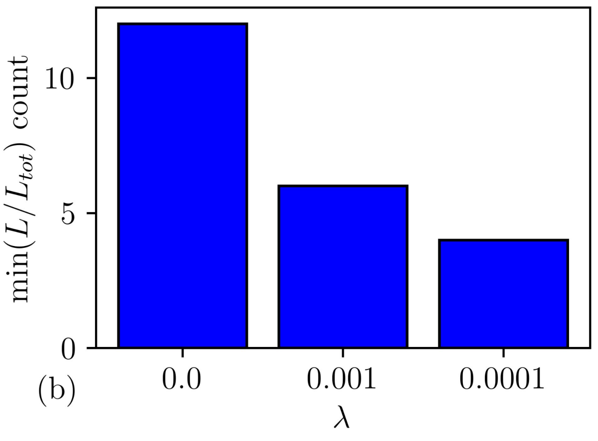
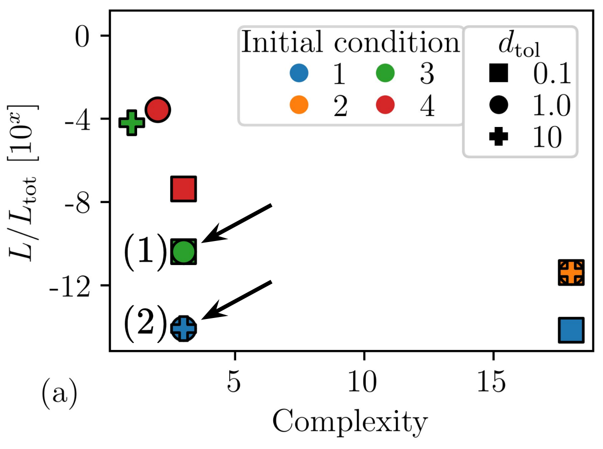
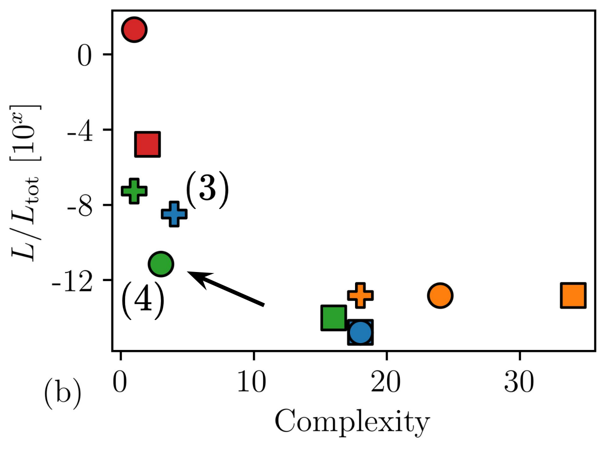
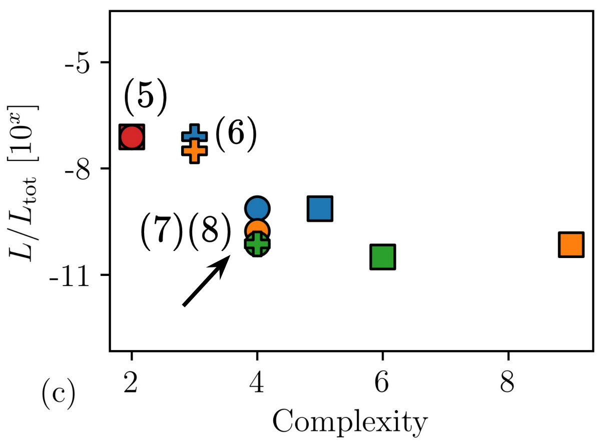


Figure 12: **(a)** We plot the normalised squared error L/L_{tot} of found PDEs with the basis set as a function of the ridge parameter λ, which includes $\lambda = 0$ on the left as denoted by the dashed lines. We do this for each initial condition with $D = 1$. We see that $\lambda = 0$ is the minimum for each case. **(b)** We bar plot the count of which λ has the lowest, second-lowest and third-lowest error among all parameter sets and initial conditions. The x-axis shows the corresponding λ. We see that $\lambda = 0$ has been counted 12 out of 12 times. Hence, $\lambda = 0$ is the optimal ridge parameter.

To obtain a first impression of the data, we will first use ridge regression on the standard basis set to find the optimal ridge parameter λ. In Fig. 12a we plot for each initial condition with $D = 1$, the normalised squared error L/L_{tot} as a function of λ. We see that $\lambda = 0$ has the lowest error for each initial condition. In addition, this error is much smaller than 1, suggesting that we can find an excellent PDE fit onto the data. In (b) we show a bar plot of the number of counts across all initial conditions and parametric sets, which λ has the first, second and third lowest error. We see that $\lambda = 0$ is counted the most (12 out of 12), hence we will use that ridge parameter for further investigation with `TrainSTRidge`.







I	Right-hand side PDE
🟩 1	1.00u-1.00u^2+1.00u_{xx}
🟦 2	1.00u-1.00u^2+1.00u_{xx}
🟦 3	351.42u^3-702.15u^4+5.00u_{xx}-4.44u^3u_{xx}
🟩 4	0.21u-0.21u^2+5.00u_{xx}
🟥 5	100.00u-100.00u^2
🟧 6	99.99u-99.99u^2+0.01u_{xx}
🟧 7	99.95u-99.85u^2-0.10u^3+0.01u_{xx}
🟦 8	99.95u-99.85u^2-0.10u^3+0.01u_{xx}

Figure 13: We scatter the error versus the complexity for the PDEs we found with `TrainSTRidge` for $\lambda = 0$. This is done for different values of the input thresholds d_{tol} to find the optimal level of complexity. (a) Pareto front of the data set with $D = 1$. Here we plot the normalised squared error L/L_{tot} of the found PDEs on the y-axis, versus the complexity on the x-axis. This is done for different initial conditions as depicted in different colours. The different shapes correspond to PDEs resulting from `TrainSTRidge` with different d_{tol}. The arrows denote the PDEs that are in the elbow of the pareto front curve, *i.e.*, the PDEs that have the best trade-off between accuracy and complexity. The PDEs with an attached number are highlighted in the table, where the cell colours in the table match the initial condition colours. (b) and (c) show the same type of plot for the data sets corresponding to $D = 5$ and $D = 0.01$, respectively. In the table we colour the terms blue, if they are part of the Fisher equation. If that is not the case, we colour them red. The coefficients are rounded to the second decimal place. The pareto selected PDEs denoted by the arrows (closely) match the Fisher equation with the correct coefficients.

For `TrainSTRidge` we start with four different thresholds $d_{\text{tol}} \in \{0.01, 0.1, 1, 10\}$, and we will use 25 increments. This allows us to ascertain what the optimal threshold is. For `STRidge` we used 10 sweeps of setting the coefficients to zero. In Fig. 13(a-b-c), we scatter plot for each initial condition the errors versus the complexity of the found PDEs for $D = 1$, $D = 5$, and $D = 0.01$, respectively. The arrows highlight the Pareto front, *i.e.*, the PDEs that have the best tradeoff between accuracy and sparsity. In the accompanying table we show the found PDEs and their coefficients. We see that the Fisher equation has the best tradeoff in each parameter set, except for $D = 0.01$, where we also find the Fisher equation with an additional u^3 term. We also see that the coefficients of the Fisher equation match with the input parameters.

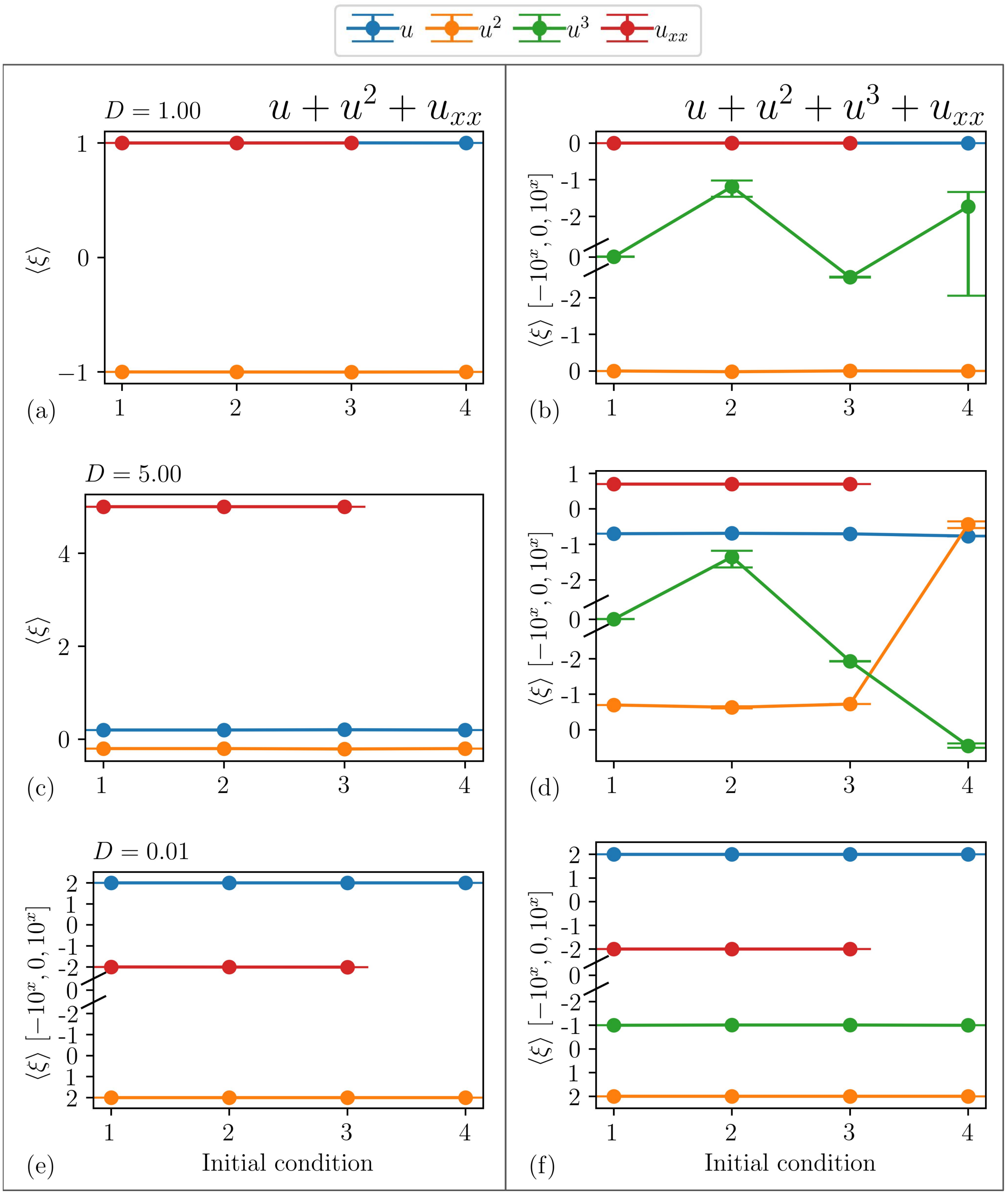
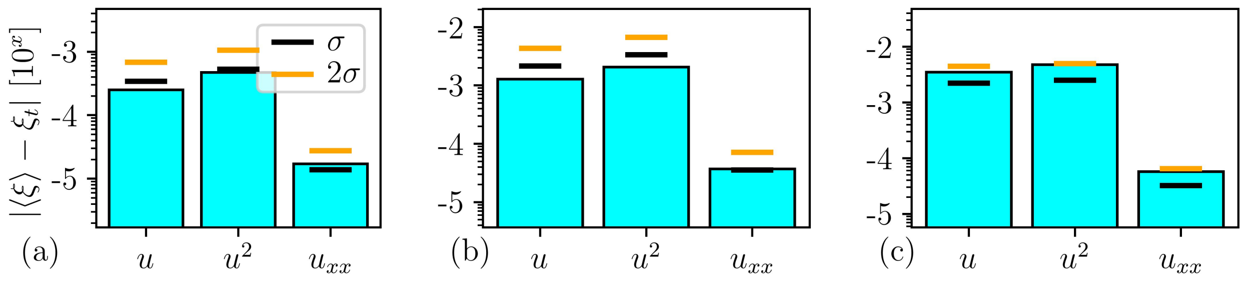


Figure 14: Consistency plot of the coefficients of found PDEs across different initial conditions and parameter sets. For each subfigure, we plot the average coefficient $\langle\xi\rangle$ of terms of a particular PDE as a function of the initial condition. The average is taken over output PDEs resulting from 10 different space and time slices of the library data $\Theta(x,t)$. In the first column, *i.e.*, subfigure (a)-(c)-(e) we show the results for PDE 1: $u_t = \xi_0 u + \xi_1 u^2 + \xi_2 u_{xx}$. For (b)-(d)-(f) we show results for PDE 2: $u_t = \xi_0 u + \xi_1 u^2 + \xi_2 u^3 + \xi_3 u_{xx}$. Rows (a)-(b), (c)-(d) and (e)-(f) correspond to the results for $D = 1$, $D = 5$, and $D = 0.01$, respectively. The orange and/or the green line in (b) and (d) are not consistent across the initial conditions, hence PDE 2 does not match the data correctly.



D	r	u	u^2	u_{xx}
1	1	1.0003 ± 0.0003	-(1.0005 ± 0.0005)	1.00002 ± 0.00001
5	0.2	0.201 ± 0.002	-(0.202 ± 0.003)	5.00004 ± 0.00004
0.01	100	99.997 ± 0.002	-(99.995 ± 0.002)	0.00994 ± 0.00003

Figure 15: **(a-b-c)** For $D = 1$, $D = 5$, and $D = 0.01$, respectively, we bar plot the absolute difference between the true coefficient ξ_t and the measured coefficient $\langle\xi\rangle$ for each term of the Fisher equation. The x-axis denotes the term corresponding to the bar. We also compare the differences with the standard deviations σ and 2σ, which are depicted by the horizontal bars. In the accompanying table we show the values of the true and measured coefficients, where we have rounded to the most significant digit of the standard deviation. Each measured coefficient is within one or two standard deviations of the true value, hence there is an excellent agreement between the input and output values.

This is promising, but we want to make sure that the Fisher equation is indeed the best description of the data, given the chosen basis set. For that we will compare the coefficients across different initial conditions. We will do the same with the Fisher equation with the cubic term added to it. In order to obtain statistics that allow for comparison, we adopt the method outlined in section 2.8, *i.e.*, we will slice the data, and obtain coefficients for each slice by applying least squares on the PDE (not the the basis set). This allows us to obtain the average coefficients $\langle\xi_j\rangle$ and standard deviation σ_j.

In Fig. 14(a-c-e) we plot for the Fisher equation and each parametric set, the average coefficients as a function of the initial condition. We have left out $\xi_{u_{xx}}$ for the fourth initial condition, since it is a straight line without any gradients, *i.e.*, $u_{xx} = 0$. We see that all of the coefficients are straight lines, *i.e.*, they are constant. In addition, they seem to be consistent with the input values. We will examine the consistency down below. First we also have a look at the other PDE we found. In Fig. 14(b-d-f) we show the results for the Fisher equation with u^3. We see that all of the Fisher terms are consistent in each initial condition. In addition, for the cubic term is consistent in (f), *i.e.* for $D = 0.01$, for which we found this PDE. However, it is not consistent in the other parameter sets. In (b-e) we see that $\langle\xi_{u^3}\rangle$ swaps sign across different initial conditions. This is a clear indication that this term is unphysical. Based on this result, we can state that the Fisher equation is indeed the best PDE that describes the dynamics of data generated by the Fisher equation.

To see how well least squares can fit the coefficients, we compare the found values with the actual coefficients ξ_t. In Fig. 15 we plot $|\overline{\xi} - \xi_t|$ for each term in the Fisher equation and compare it to the standard deviation σ. For $\overline{\xi}$ and σ we vary over the coefficients from the slicing of the data, and all initial conditions. We see that for each parameter set and Fisher term, we manage to compute the coefficients up to the third decimal place and better. In addition, each result is within σ or 2σ of the true coefficient. Hence, we can find the correct coefficients with excellent accuracy, and within an acceptable error margin.

In conclusion, with our data-driven methods we can successfully select the best fit of smooth data obtained from solving the Fisher equation. In addition, we find that `TrainSTRidge` has no trouble in finding linear and non-linear terms, which should be useful when we are going to consider interacting particle systems with growth processes. In the next section we are going to apply our data-driven methods to non-interacting particle systems.

3.2 The Diffusion Equation obtained from Random Walkers

In the previous section we have tested the data-driven methods on smooth and clean data obtained by solving the Fisher equation. In this section, we will test this algorithm on data generated by non-interacting random walkers on a lattice. This will prepare us to tackle more complex particle systems with interactions.

In literature it is well known that the continuum limit of the random walk dynamics leads to the diffusion equation [5]. In addition, in the limit $N \to \infty$, where N is the number of particles, the relation will be exact. However, we cannot simulate an infinite number of particles, only a finite number. Hence, we are going to investigate how many particles we need to find the diffusion equation and its coefficients. That is, we want to find out how large N must be in order to get a good approximation of the limit $N \to \infty$. This is an important question, as it will give us insight into whether we can obtain accurate predictions for particle systems in general, and interacting particle systems in particular.

In order to answer this question, we need to undertake some preliminary steps. We will start with a short explanation of what a random walker is, and how we can obtain data from a random walk that we can use to find PDEs. After that we give a demonstration of what the effect of the number of particles is on the level of noise, and how effective the 2D FFT filter from section 2.6 is to remove that noise. We then demonstrate how we can utilise the stochastic nature of the random walk to generate many data sets that we can use for consistency checks, as described in section 2.8. From these results we will then be able to conclude that with a high enough, but finite number of particles, we can obtain a very good approximation of the diffusion equation in the high N limit.

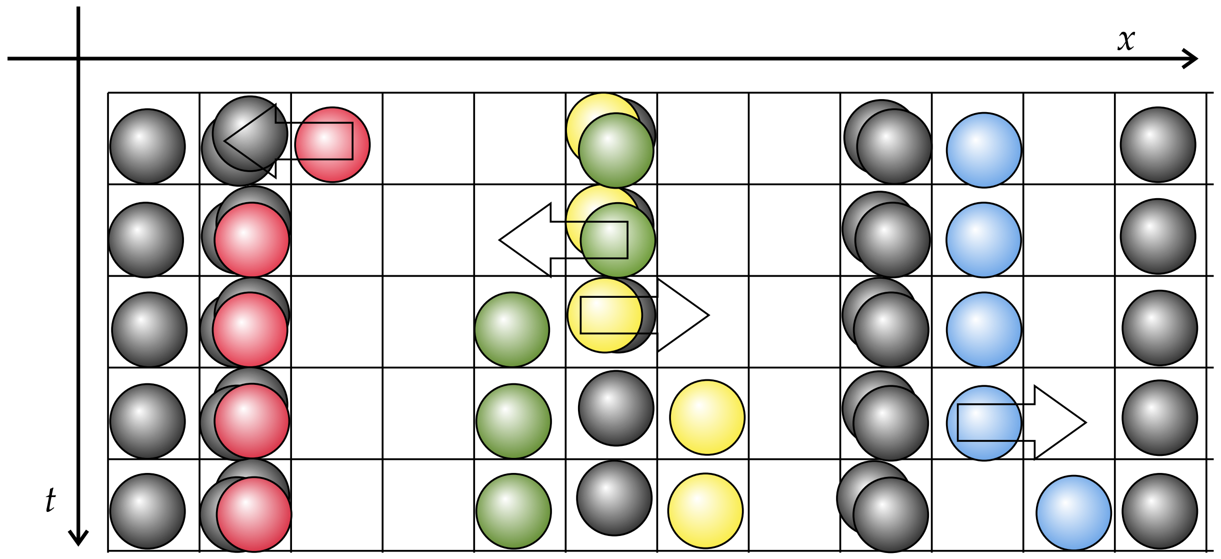


Figure 16: Sketch of a time evolution of non-interacting random walkers. Each row represents a new time step where a single particle is chosen uniformly at random to undertake an action. This can either be a movement to the left with probability p or to the right with probability $1-p$. For a symmetric random walk, $p = 0.5$, *i.e.*, there is an equal chance to move to the left as to the right.

Let us begin by saying what a non-interacting (biased) random walk particle system precisely is. As the name suggests, it is a model for particles that move randomly on a lattice, see Fig. 16 for a sketch. This lattice consists of M sites and each site is seperated with a lattice spacing Δx. We will be looking at the 1D case, so $L = M\Delta x$ is the length of the box. We will also use periodic boundary conditions, *i.e.*, the rightmost site is connected to the leftmost site. In addition, there are N particles on the entire lattice. The particles are non-interacting, hence they can overlap, *i.e.*, multiple particles can occupy a single site. Now, given a distribution of particles, we simulate the random walk by selecting a particle uniformly at random, and then let it move to the right or left with probability p or $1-p$, respectively. This step has a time unit $\Delta x^2/(2DN)$, where D is the diffusion rate, and is repeated until a given time T. For each time step and lattice site, we compute the particle density $u(x,t)$, which is the ensemble average of the occupation number, *i.e.*, the number of particles at a site divided by N. The dynamics of the density can then be described via the advection-diffusion equation

$$u_t = Du_{xx} + vu_x, \tag{22}$$

where $v = 2D(2p-1)/\Delta x$ is the speed of the body of walkers. In this section we will use $p = 0.5$, so $v = 0$. If the reader is interested in how to derive this equation from the model, see Ref. [40].

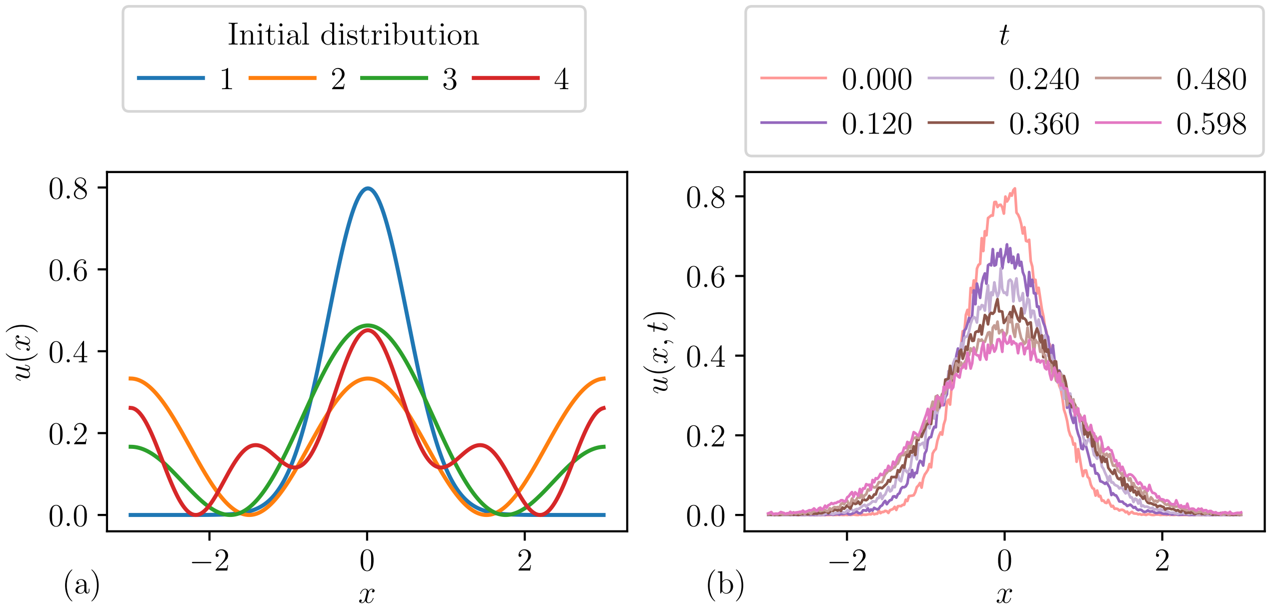


Figure 17: **(a)** Initial distributions $u(x)$ as a function of the lattice space x, which are used to draw particles for an initial condition. The functions to create the distributions are given by (1) $\exp(-x^2/2\sigma^2)$ with $\sigma = 0.5$, (2) $\cos(2n\pi x/L)$ with $n = 2$ and L the size of the grid, (3) $\cos(2n\pi x/L) + \cos(n\pi x/L)$ with $n = 2$ and (4) $\cos(2n\pi x/L) + \cos(n\pi x/L) + \cos(n\pi x/2L)$ with $n = 4$. Each function is shifted and normalised such that the minimum is at $u = 0$ and the integral under the curve is 1. **(b)** Time evolution example of the average random walker density $u(x,t)$ as a function of space x. We used the gaussian initial condition and the number of particles is $N = 10^5$. The gaussian peak shrinks and broadens, just like one would expect from diffusion.

Now that we know the dynamics, we want to generate data that we can test. Since we are dealing with a finite number of particles, the density u cannot be smooth, hence we cannot simply start with a smooth functional form $f(x)$ as an initial condition. Instead, $f(x)$ will be our initial distribution from which we randomly draw N particles. In Fig. 17a we show the distributions $u(x)$ that we are using to generate our initial conditions, which we label from 1 to 4. We create $u(x)$ by simply shifting $f(x)$ by $-\min(f(x))$, such that the minimum (which can be either positive or negative) gets shifted to zero. This prevents the density from being negative. After that we rescale the shifted function such that $\int f_{\text{shift}}(x)dx = 1$. To obtain an initial particle density $u(x,0)$, we compute the cumulative distribution function (CDF) of the initial distribution. Then, for each particle, we sample a position from the CDF, where we then place that particle. If this procedure is done, we divide the number of particles at each site by N to obtain $u(x,0)$.

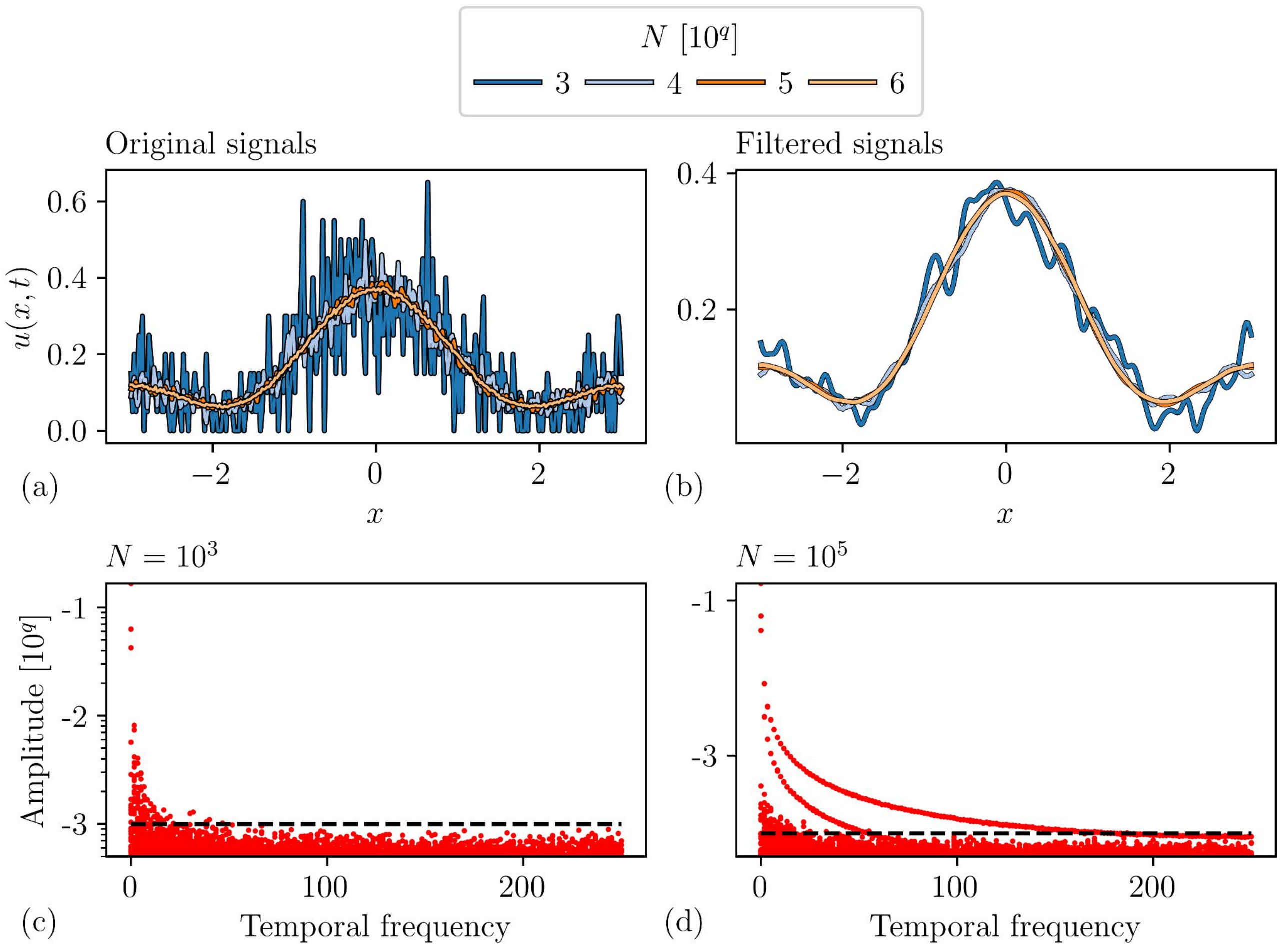


Figure 18: **(a)** We plot a single time instance of a generated random walker particle density $u(x,t)$ as a function of space x for various number of particles N. We see that for $N = 10^3$, the number of particles is too low to have a good density approximation, *i.e.*, the signal is very noisy. For higher N we get better approximations and substantially less noise. **(b)** We have filtered the signals from (a) using the 2D FFT amplitude cutoff filter with suitably chosen thresholds. The filtered signal of $N = 10^3$ is still noisy, but the high frequency fluctuations are gone. For $N = 10^5$ and higher we get very smooth signals. Hence, for high N we expect that `TrainSTRidge` will have no trouble finding the diffusion equation. For $N = 10^3$ this will be more challenging, but not impossible considering that the general form of the signal is still visible. **(c-d)** We fourier transform the $N = 10^3$ and $N = 10^4$ signal from (a) and scatter the amplitudes as a function of the temporal frequencies. Note that this is a 2D plot, where the spatial frequency axis goes into the screen. The black dashed line, sitting just above the noise level, is the amplitude cutoff we have chosen for (b). We see that for $N = 10^3$, there is no clear distinction between the noise and the signal. So we need to remove information about the signal, which we expect will lead to inaccuracies in determining the diffusion constant. For $N = 10^5$, the amplitudes of the noise are lower, revealing a distinguisable signal. So for this N we expect accurate predictions of the diffusion coefficient.

Once we have the initial condition, we can start generating data for the symmetric random walk. For the purposes of obtaining representative results with `TrainSTRidge`, we will generate 50 different data sets for all initial conditions, each with a different rng seed. We will do this for four different numbers of particles: $N \in \{10^3, 10^4, 10^5, 10^6\}$. All other input parameters will be the same, such that we will have fair comparisons. These are $D = 0.5$, $p = 0.5$, $\Delta x = 0.02$, $L = 6$, $\Delta t = 0.002$ and $T = 0.6$.[7]

In Fig. 17b we show an example of the time evolution of a generated density $u(x, t)$, starting from a gaussian initial condition with $N = 10^5$. We see that the top of the gaussian shrinks, whilst the tail becomes broader. This is a sign of diffusion, which suggests that we indeed can obtain the diffusion equation within the chosen physical regime. We also see that the density is noisy, hence it will be important to filter the signal to obtain smooth derivatives.

In Fig. 18 we demonstrate the efficiency of the Fourier filter from section 2.6 on the random walker data for different values of N. In (a) we plot the density as a function of space for a single time. We see that $N = 10^3$ is very noisy and that increasing Ndrastically decreases the fluctuations around the signal. Hence, higher N gives smoother densities after applying the filter, as is shown in (b).

For the filtering proces we choose an amplitude cutoff that is right above the noise level. In (c-d) we show an example of this by plotting the amplitude of the fourier signal versus its temporal frequencies for $N = 10^3$ and $N = 10^5$. For $N = 10^3$, we must choose a cutoff of $A_{\text{cut}} = 10^{-3}$ to get rid of most of the noise. However, since a big part the signal is buried under the noise, we will also remove some information of the underlying signal as well. For $N = 10^5$ the noise level is lower, revealing a clean signal. Thus, with $A_{\text{cut}} = 10^{-4}$ we obtain a smooth density upon inversing the fourier transformation after applying the cutoff. For $N = 10^4$ and $N = 10^6$ we have chosen $A_{\text{cut}} = 3 \cdot 10^{-4}$ and $A_{\text{cut}} = 3 \cdot 10^{-5}$, respectively. With this filter we expect that data for $N = 10^5$ and $N = 10^6$ will give accurate predictions of the diffusion equation. $N = 10^4$ seems to yield a decent approximation of the density, but we think that the small frequencies will increase the deviations from the diffusion constant. Finally, for $N = 10^3$ we expect very high fluctuations in the diffusion constant. We do think that in this case, `TrainSTRidge` can still find a match with the diffusion equation, because the general form is visible by eye, and because we will apply the SG-filter to obtain the derivatives.

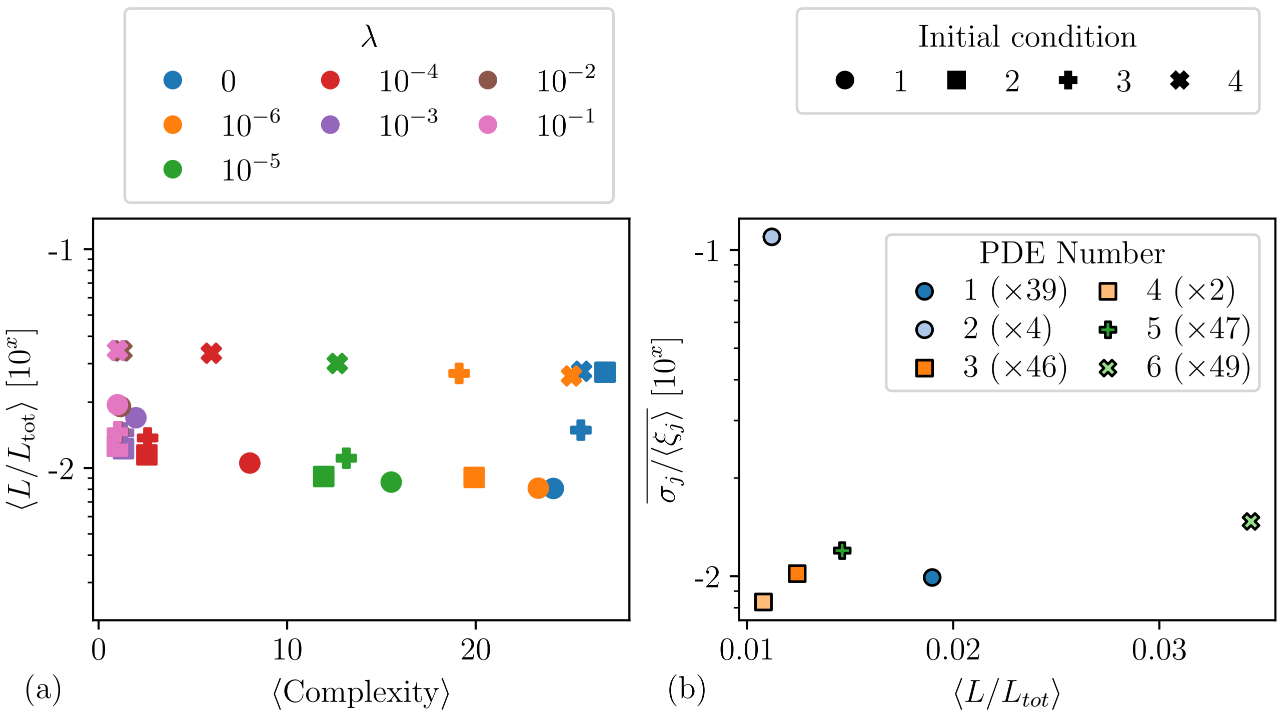
For the SG-filter we will choose different polynomial widths per initial condition, because some of them have steeper slopes. From initial condition 1 to 4, these are $x_{\text{w}} \in \{1.22, 2.02, 2.02, 1.41\}$ in units of space. For each of them we use a polynomial degree of 5. For the temporal direction we use the entire time range with degree 3, because diffusion only gives rise to slowly decaying or growing densities per spatial point, for which we do not need high order polynomials.[8] [9] We will also remove the first and final temporal data by $t_{\text{cut}} = 0.02$ to get rid of undesirable boundary effects of the Fourier filter and SG-filter.

With the derivative settings in place, we can construct the library $\vec{\Theta}$ for each data set. We will choose the standard basis set, consisting of combinations of polynomial powers and derivative degrees up to five. For `TrainSTRidge` we use 100 iterations with initial tolerance $d_{\text{tol}} = 1$. For `STRidge` itself we will have 36 iterations. We will now give an example of the `TrainSTRidge` output for $N = 10^5$. The results for the other N are similar.

[7]Note that this time step is the time difference between taken samples, not the intrinsic time step of the simulation.

[8]Lowering the degree even further and decreasing the width is also an option. However, we found that this gives highly fluctuating time derivatives for low N, which we want to avoid.

[9]This argument is not applicable to the spatial direction, because at each time step, we have steep gradients due to the initial conditions. To approximate these gradients, we need higher order polynomials



I	Right-hand side PDE	Count
1	$\xi_1 u_{xx}$	39
2	$\xi_1 u^3+\xi_2 u_{xx}+\xi_3 u u_{xx}+\xi_4 u u_{xxxx}$	4
3	$\xi_1 u_{xx}$	46
4	$\xi_1 1+\xi_2 u^2+\xi_3 u_{xx}+\xi_4 u u_{xxxx}$	2
5	$\xi_1 u_{xx}$	47
6	$\xi_1 u_{xx}$	49

(c)

I	$\langle\xi_{u_{xx}}\rangle$
1	0.489±0.005
3	0.509±0.005
5	0.501±0.006
6	0.491±0.007

(d)

Figure 19: We applied `TrainSTRidge` to the 50 generated random walker data sets for all initial conditions, each with a different rng seed. We average the results from the different seeds, which we express via $\langle\cdot\rangle$. In this figure we highlight our findings for $N = 10^5$. **(a)** We scatter the average normalised squared error $\langle L/L_{\text{tot}}\rangle$ versus the average complexity of the PDEs we found for different ridge parameters λ. We see that the error is relatively constant for each λ, whilst the complexity is minimised by $\lambda \geq 10^{-3}$. Hence, we will investigate the PDEs found with $\lambda = 10^{-3}$. **(b)** We show for all initial conditions a pareto of the significant PDEs, *i.e.*, PDEs that have been counted more than once in the rng seed set. In this context, the average $\langle\cdot\rangle$ is taken over the PDEs of the same form. The PDEs and their count can be seen in table **(c)**. For each of these PDEs, we scatter the mean error on the coefficients $\overline{\sigma_j/\langle\xi_j\rangle}$ as a function $\langle L/L_{\text{tot}}\rangle$, where σ_j is the standard deviation around the average coefficient $\langle\xi_j\rangle$. We see that the diffusion equation has been found the most often in each initial condition. The other two PDEs have a higher complexity, a lower count and are not significantly present in the other initial conditions. In addition, PDE 2 has a much higher coefficient error than PDE 1. It is interesting to see that PDE 4 has the best pareto tradeoff, but this is likely caused by the low count. From all of this we can conclude that the random walker data is indeed best described by diffusion. **(d)** We show the average coefficients that are measured for each initial condition. We see a good agreement with the input coefficient $D = 0.5$, with only a difference that is slightly larger than the standard deviation for some of the initial conditions.

First, we want to find the optimal ridge parameter λ, *i.e.*, the parameter that has the best tradeoff between a low complexity and low squared error. The general procedure of finding this λ is now slightly different compared to the clean example we examined in the previous section. In that case we had only a single instance of the data $u(x,t)$ that we examined. Now, we will apply `TrainSTRidge` to 50 different data sets (per initial condition and N). Hence, we want to make use of the statistics we obtain from all of the resulting PDEs, instead of looking at each data set individually. We will do this by computing specific quantities for each found PDE, *e.g.*, the normalised squared error L/L_{tot}, and averaging them over the entire rng data set. We will denote this average by $\langle\cdot\rangle$.

In Fig. 19a we scatter plot for different values of λ, the average normalised squared error $\langle L/L_{tot}\rangle$ of the found PDEs versus their average complexity. This we do for each initial condition. Unlike the previous cases, $\lambda = 0$ is not successful in minimising the complexity. We see that the complexity decreases for increasing λ. After $\lambda = 10^{-3}$, the complexity does not change significantly anymore, as it is sitting slightly above 1. In addition, we see that per initial condition, the squared error only increases a little bit for increasing λ. Hence, the data for $\lambda \geq 10^{-3}$ seems to hold the most promising PDEs. From this we can also conclude that regularisation of the regression process becomes more important if we have noisy data, compared to clean data.

We will now look into the significant PDEs that were found for $\lambda = 10^{-3}$. With significant we mean a particular form of the PDE that has been counted more than once in the entire rng seed data set (per initial condition). Hence, we obtain different numerical values for each coefficient ξ_j of the significant PDE. Over these values we take the average to obtain $\langle\xi_j\rangle$ and the standard deviation σ_j. From these we can compute the fraction $\sigma_j/\langle\xi_j\rangle$, which can be interpreted as the error on a coefficient. Since a PDE can have multiple terms, we will take the mean over the coefficients to obtain $\overline{\sigma_j/\langle\xi_j\rangle}$. This quantity is low if the coefficients from different rng seed sets are consistent with each other, so we can use this error in a pareto tradeoff plot.[10]

In Fig. 19b we scatter plot per initial condition, the mean error of the coefficients $\overline{\sigma_j/\langle\xi_j\rangle}$ versus $\langle L/L_{\text{tot}}\rangle$. In table (c) we show the forms and rng seed counts of the pertinent PDEs in (b). We see that for initial condition 1, PDE 1, which is the diffusion equation, has a coefficient error ten times smaller than PDE 2, while the squared error is only twice as big. Hence, in that initial condition, the diffusion equation is the clear winner. In addition, we find the diffusion equation in the other initial conditions, all with an overwhelming rng seed count. We also see that for initial condition 2, there is a candidate with the lowest squared error and lowest coefficient error. However, its scatter point sits not far away from the diffusion equation point and it also includes the diffusion term. Furthermore, it has only been counted twice, which does not make it statistically relevant. For the other initial conditions, only the diffusion equation was significant. Hence, we see that `TrainSTRidge` has no trouble finding the diffusion equation from random walker data.

We will now turn our attention to the computed diffusion constants. In table (d) we show the average diffusion constant $\langle\xi_{u_{xx}}\rangle$ and standard deviation for each initial condition. As expected, the diffusion coefficient sits around $D = 0.5$, with values that are only a few standard deviations away from each other. The differences are not within one standard deviation, suggesting there might be some other errors in our procedure that are not captured by the random deviations due to N, *e.g.*, due to the approximations with the SG-filter. Nevertheless, the results affirm that our expectation that $N = 10^5$ yields good approximations to the diffusion equation.

[10]Note that this error only makes sense if the PDE is significant, since PDEs with only a single count will have $\sigma_j = 0$. This would imply that these PDEs have the lowest error, which is of course nonsense.

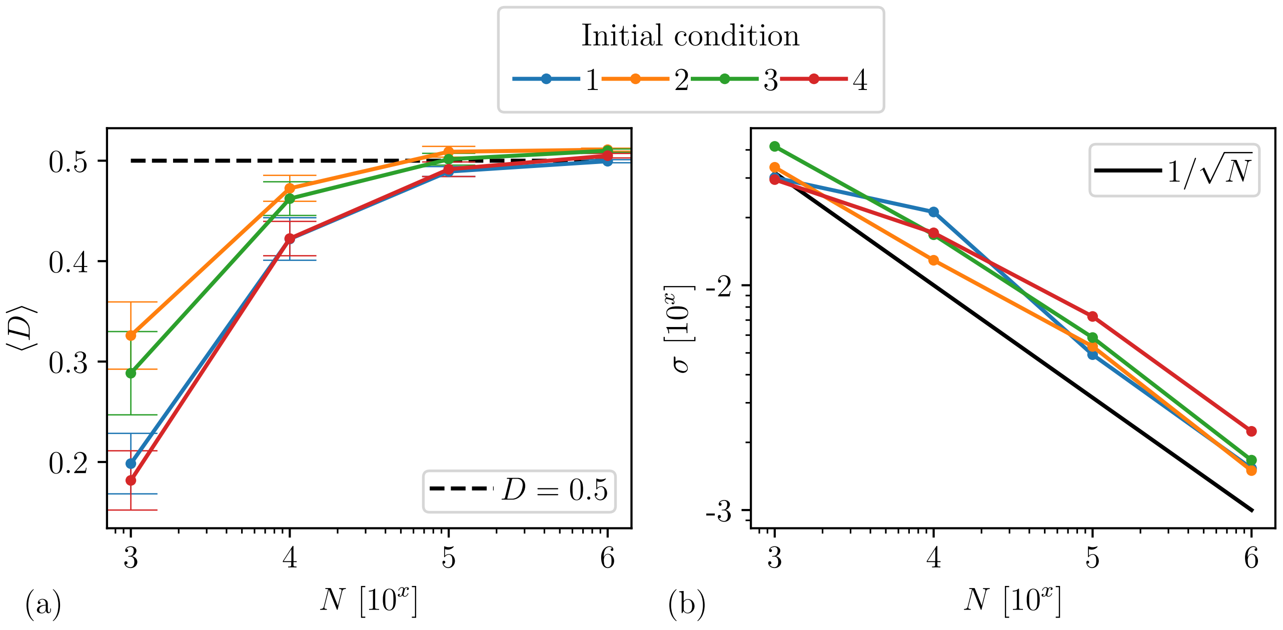


Figure 20: **(a)** We plot the average diffusion constant $\langle D \rangle$ and its standard deviation σ found with `TrainSTRidge` for $\lambda = 10^{-1}$ as a function of N. The different lines correspond to results from different initial conditions. For $N = 10^3$, the measured diffusion constant is considerably lower than the input diffusion of $D = 0.5$. Nevertheless, it is still impressive that `TrainSTRidge` managed to find the heat equation at all for this N. For $N = 10^4$, the coefficients are much closer to the input and for $N = 10^5$ and $N = 10^6$ we get a very good match. Note that in some cases, the coefficients are not one standard deviation away from the input, especially for lower N. To estimate how well the standard deviations capture the variations due to the fluctuations of the number of particles, we plot in **(b)** the standard deviation as a function of N and compare it to the gaussian deviation $1/\sqrt{N}$. We see that the variations per initial conditions are of the same order of magnitude as $1/\sqrt{N}$, and sometimes even larger. Hence, for $N = 10^3$, there is certainly a bias of the average coefficients, meaning that the density approximation is not very accurate, as expected from the high level of noise we saw in figure 18. For $N = 10^5$ and $N = 10^6$, the density approximation is excellent, meaning that a high enough, but finite number of particles is sufficient to approximate the limit $N \to \infty$.

As we said before, the pareto results are similar for each N, with the only difference being a shift in the value of $\langle L/L_{\text{tot}}\rangle$ and the optimal ridge parameter λ. We found that with $\lambda = 10^{-1}$, each N, including $N = 10^3$, pointed to the diffusion equation as the optimal PDE for the entire data set. In Fig. 20a we plot per initial condition the average diffusion constant as a function of N, including the standard deviations. For $N = 10^5$ and $N = 10^6$ the coefficients sit nicely around the expected value of $D = 0.5$, implying that for $N \geq 10^5$ we can obtain very good approximation for the limit $N \to \infty$.

For $N = 10^3$ and $N = 10^4$ we see that the coefficients are below $D = 0.5$. This means that effectively, the filtered signal did not decay as fast as it would for a higher number of particles. This is supported by our observations that for later times, the height of the filtered peaks stopped decreasing. One explanation for this is that the apparent spread of the density due to diffusion is obscured by the noise. That is, the signals are too noisy to differentiate decay from high frequency fluctuations. This is not surprising given the fact that a large part of the signal had to be cut away, because the fourier amplitudes were not distinguishable from noise (see Fig. 18c. Another explanation could be an introduced bias of the coefficients due the SG-filter. But since we used the same SG parameters for every N, we think that changing the polynomial width or degrees will not affect the average diffusion constant for $N = 10^3$ by a lot.

One matter that we can easily check with the given data is the accuracy of the standard deviation σ. That is, we want to compare σ with the expectation from statistical fluctuations to check for systematic errors in our method or the underlying data. We shall do this by comparing σ to the gaussian deviation $1/\sqrt{N}$. In Fig. 20b we plot σ as a function of N in a log-log plot. We see that for $N = 10^3$, the deviations are comparable to $1/\sqrt{N}$. For $N \geq 10^4$, σ is even higher than the gaussian deviation. This is not surprising given the fact that there are many other variables that can affect the deviation, *e.g.*, the SG-filter widths that may perform differently for different rng seed densities. In any case, this does give us confidence that the variations with the rng seeds gives us faithfull results from a statistical point of view. Hence, increasing the number of data sets on which we can test diffusion, will not alter the existing results by much.

To summarise our results for the symmetric random walk, let us start by saying that for each N, there is a detectable diffusive process that we can easily examine using our data-driven methods if we utilise the random nature of the particle system. In addition, we find that a high enough, but finite number of particles is enough to get accurate approximations of a continuum description. Hence, for the interacting particle systems we will make sure that we will use a high number of particles.

3.3 The Advection-Diffusion Equation obtained from Biased Random Walkers

In the previous subsection we found that with $N = 10^5$, we are able to find the diffusion equation. We will now test if we can also obtain the advection-diffusion equation if we consider biased random walkers. In our studies of this system, we found that `TrainSTRidge` is unable to keep the diffusion term: it throws away all of the terms, except for the advection term u_x. We have addressed this problem in section 2.5, where we note that the use of the condition number in the error leads to an amplified bias towards sparse PDEs. In that section we also propose an alternative method, which circumvents the problem by simply setting the loss function to zero after a certain tolerance threshold. With our new method we are able to find both advection and diffusion.

We will use the same parameters as before, except for the probability to move to the right: $D = 0.5$, $\Delta x = 0.02$, $L = 6$, $\Delta t = 0.002$, $T = 0.6$ and $p = 0.6$. From this we can also compute the expected theoretical value of the speed: $v = 2D(2p - 1)/\Delta x = 10$.

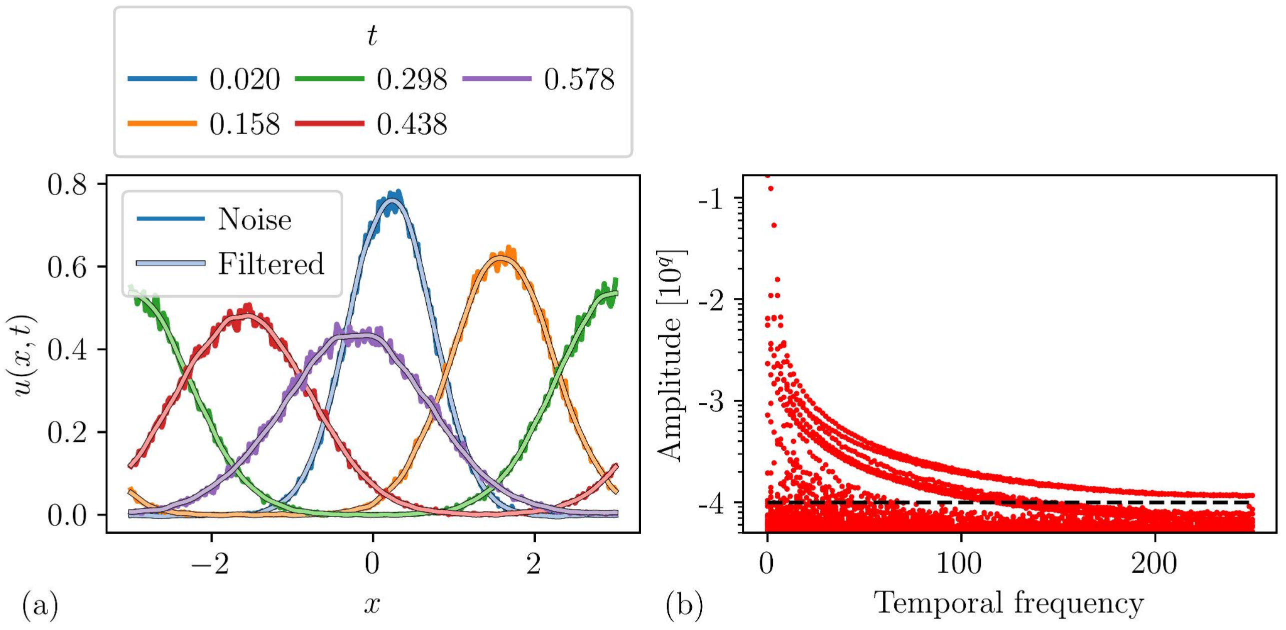


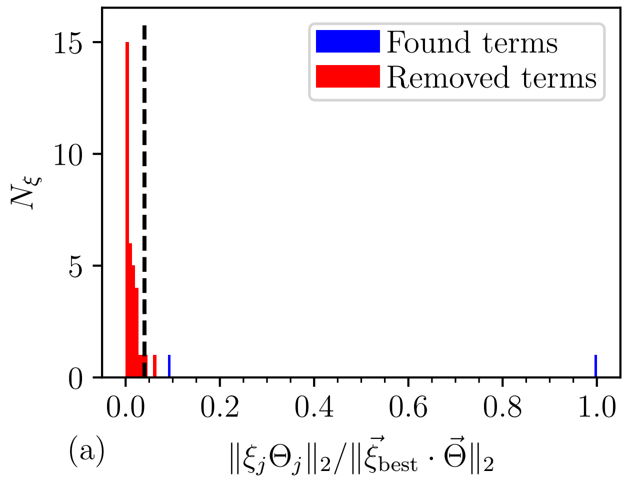
Figure 21: **(a)** Snapshots of a generated biased random walker density $u(x,t)$, with $N = 10^5$, $D = 0.5$, $p = 0.6$ and $\Delta x = 0.02$. The dark colored lines show the original noisy signal and the light colored lines are the filtered signals. We see that the entire density translates to the right over time due to the bias. In addition, the peak widens as it shrinks. Hence, there is a clear advection and diffusion process going on. Furthermore, the filtered signal is smooth, making it well suitable for `TrainSTRidge`. **(b)** We plot the amplitudes of the fourier transformation of the noisy signal in (a) as a function of the temporal frequency. The top decaying red lines correspond to the underlying signal, while the points below the cutoff are mostly noisy frequencies. Thus the filter successfully keeps the signal and removes the noise.

Our initial functions to create the initial distributions will be $f(x) = \exp(-x^2/2\sigma^2)$ with (1) $\sigma = 0.5$ and (2) $\sigma = 0.2$. The other function will be $f(x) = \cos(\omega x) + \cos(\omega x/2) + \sin(2\omega x)$ with (3) $\omega = 2\pi/L$ and (4) $\omega = 4\pi/L$.[11] In total we will generate 100 densities with a different rng seed (per initial condition). In Fig. 21a we show an example of a biased random walk

[11]We have different initial conditions compared to the $p = 0.5$ case, because we wanted to make sure that the first spatial derivative differs from the third derivative. This is not the case for $\cos(2\pi x/L)$.

density as a function of space x for different times t. Just as in Fig. 17b, the gaussian initial condition shrinks and broadens. What is different, is of course the advection of the population to the right. We also see that below $T = 0.6$, the density has almost returned to its initial position. Since the box has length $L = 6$, we see that the speed of the travelling population is indeed around the expected value of $v = 10$. We have also filtered the signal. Similar to the random walker case, we use a amplitude cutoff $A_{\text{cut}} = 10^{-4}$ for the filter. In Fig. 21b we show the fourier signal of the density. We again see a clear signal rising above the noise level, hence we also expect that the derivatives will be smooth and accurate representations of all the gradients.

For the SG-filter, we will use the following spatial and temporal polynomial widths from initial condition 1 to 4: $x_{\text{w}} \in [1.5, 0.62, 2.02, 1.22]$ and $t_{\text{w}} \in [0.15, 0.082, 0.17, 0.102]$.[12] For all cases we will use a polynomial degree of 5. We found these widths by visually checking how smooth the first derivatives were after applying the Fourier and SG filter.



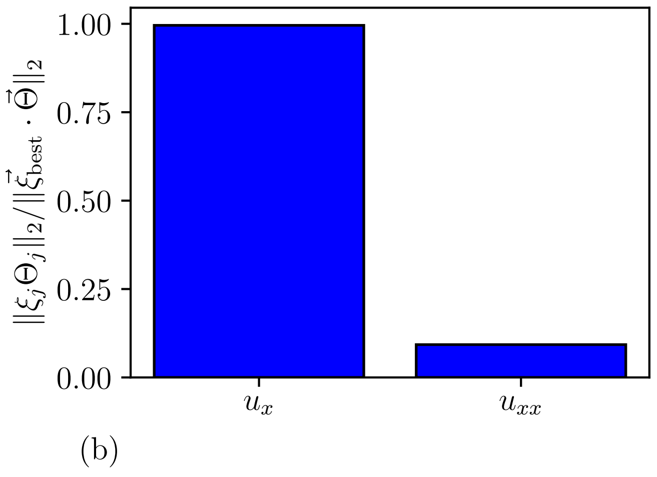


Figure 22: We applied `TrainSTRidge` with the tolerance threshold method on the biased random walker data set with $N = 10^5$, $D = 0.5$, $p = 0.6$ and $\Delta x = 0.02$ (see section 2.5.1 for more details). **(a)** We plot a PDE contribution histogram of the removed (red) and found (blue) coefficients ξ_j, where $\|\xi_j\Theta_j\|_2/\|\vec{\xi}_{\text{best}}\cdot\vec{\Theta}\|_2$ is the relative contribution that we count on the y-axis as denoted by N_ξ. For this we used 201 bins in the range $[0, 1]$. The dashed line is the tolerance threshold we have computed. We see that two terms are not cut away, which correspond to advection and diffusion. This can be seen in **(b)**, where we make a bar plot of the surviving terms with the relative contribution on the y-axis and the corresponding terms on the x-axis. Hence, with our new method we can reliably find the optimal tolerance and thus the optimal PDE.

To compute the tolerance threshold, we used 201 bins between 0 and 1 for the regression contribution histogram (see figure 5 for an example). In Fig. 22 we show the results of the `TrainSTRidge` algorithm using the tolerance threshold method. In (a) we plot a contribution histogram of removed and kept terms, where we count the number of terms that have a particular relative contribution $\|\xi_j\Theta_j\|_2/\|\vec{\xi}_{\text{best}}\cdot\vec{\Theta}\|_2$. The dashed line corresponds to the computed tolerance threshold. We see that the bulk of the removed terms are below this threshold, while the found terms are above it. Interestingly enough, there is also a single removed instance above this threshold. This means that there was no accuracy penalty to remove that particular term, even though it had a relatively high contribution in the overfit. But more importantly,

[12]The reason why we change the temporal width per initial condition is because at a fixed spatial point, the density goes up and down as a function of time due to the travelling population.

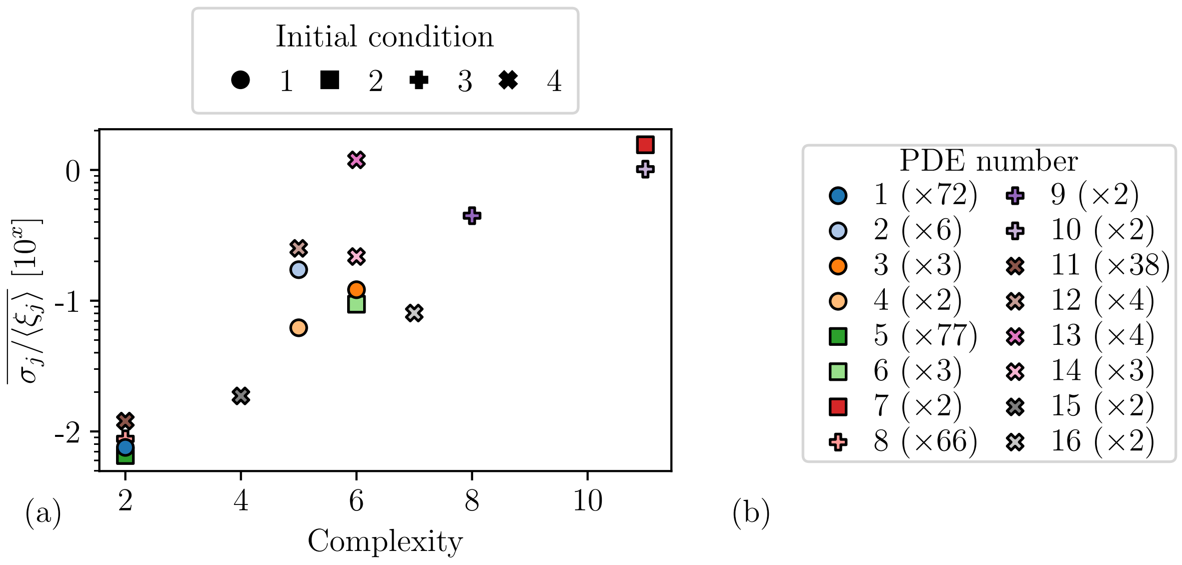
note that we have two leftover terms. In (b) we show what these surviving contributions are. As expected, they correspond to advection and diffusion.

Using this new method, we can apply `TrainSTRidge` to the entirety of the biased random walker data. Similar to the random walker, we use the basis set library. We again perform 36 `STRidge` and 100 `TrainSTRidge` iterations with initial tolerance $d_{\text{tol}} = 1$.

We first find the optimal ridge parameter in a similar vein to figure 19. We found this to be $\lambda = 10^{-3}$. In Fig. 23a we scatter plot the coefficient error $\overline{\sigma_j/\langle\xi_j\rangle}$ versus the complexity of the significant PDEs. In (b) we show the corresponding PDEs, including the number of times that particular PDE was counted in the entire rng data set (per initial condition). We find that there is a very clear pareto front in the bottom left corner, with complexity 2 and a coefficient error around 10^{-2}. This is below any other PDE in the scatter plot. Furthermore, they each correspond to a different initial condition and have been counted the majority of times. As expected, these PDEs are the advection-diffusion equation, which means that our new method works very well for the biased random walk.

In table 23c we show the corresponding coefficients of the advection and diffusion term. We see that for each initial condition, the speed sits perfectly around the theoretical value of $v = 10$ and is within the standard deviation. The same is true for the diffusion constant $D = 0.5$. Hence, with $N \geq 10^5$, we can also make accurate approximations of the advection-diffusion equation.

In addition, it is very interesting to see that the predictions for the diffusion rate are more consistent for the biased random walk system than for the symmetric walkers case. We think that the reason for this is because the temporal signals of the biased random walkers are easier to fit through with a polynomial compared to symmetric walkers. Thus, polynomial fitting tools to approximate the derivatives shine in the presence of many gradients in the spatial *and* temporal direction. For future research, it would thus be interesting to see if we can get more consistent measurements of the diffusion constant for lower N by using the biased random walkers instead of the random walkers. In this way, we can really make sure whether the bias of the diffusion coefficients in the random walker system is due to the noise of the signal, or because the SG-filter is a suboptimal choice for a diffusion only process.



(c)

I	Init.	$\langle\xi_{u_x}\rangle$	$\langle\xi_{u_{xx}}\rangle$
1	1	-9.998±0.009	0.500±0.007
5	2	-9.99±0.01	0.499±0.006
8	3	-10.00±0.01	0.497±0.008
11	4	-10.00±0.02	0.51±0.01

Figure 23: Biased random walker results for $N = 10^5$, $p = 0.6$ and $\Delta x = 0.02$ of the `TrainSTRidge` algorithm with the tolerance threshold method for the error evaluation. We generated 100 data sets with different rng seeds for each of the four initial conditions. Similar to the random walker case, we found an optimal of ridge parameter $\lambda = 10^{-3}$, which we used to investigate the found PDEs. **(a)** We scatter the error of the coefficients $\overline{\sigma_j/\langle\xi_j\rangle}$ versus the complexity. The average $\langle\cdot\rangle$ is taken over the PDEs with a unique form and σ_j is the standard deviation of the average coefficient $\langle\xi_j\rangle$. We see a very clear pareto front of four PDEs in the bottom left corner at complexity 2, each from a different initial conditon. When we look at the legend in **(b)**, we see that they have been found in many rng seed instances, while maintaining a very low coefficient error compared to the PDEs with a low count. Hence, it is no surprise that they correspond to the advection-diffusion equation, which can be seen in table **(c)**. Thus, our new method for finding the optimal threshold in `TrainSTRidge` gives us more reliable results than the standard procecure. In the table we also see that each of the coefficients are statistically consistent with each other and expected coefficients $D = 0.5$ and $v = 10$.

4 Learning Partial Differential Equations in Interacting Particle Systems

In this section we are going to apply our data-driven methods on interacting particle systems. We will start with the Eden-SSEP model, for which we will obtain the expected Fisher equation. Then we will examine travelling waves of the pure Eden model. We will compare the Eden results with the CP above criticality, and finally we will examine the CP at the critical point, for which we will obtain an effective PDE.

4.1 The Fisher Equation obtained from the Eden Model combined with the Symmetric Simple Exclusion Process

In the previous sections we have examined with our data-driven methods both the Fisher equation and the noninteracting random walk on a lattice. In this section we will add our first layer of complexity to the random walk, namely exclusion and particle growth. The random walk with exclusion is called the symmetric simple exclusion process (SSEP). A sketch can be seen in Fig. 24a. The movement of the particles are exactly the same as described in the previous section: particles move to the left or right with diffusion rate D. The big difference is that in this case, particles cannot overlap and a diffusive move to an already occupied site will be denied. We will combine the SSEP with the Eden model, which is a simple stochastic growth model, see Fig. 24b for an overview of the rules. In this model, particles can give birth to neighbouring sites with a rate r. But if a site is already occupied, no new particles can grow on it. See Ref. [41] for an overview of both models.

Similar to the Fisher equation the Eden-SSEP model can be considered as a model for moving and growing populations, such as bacteria. As we said in the introduction, it is well known that the continuum limit of the model yields the Fisher equation [7]. Intuitively, this is due to the fact that in the continuum limit, the random walk steps turn into diffusion, such that the occupation number of neighbouring sites become uncorrelated. This way, we can assume a mean field approximation, which we can use to derive the terms $ru(1-u)$.

Let us now consider the Fisher equation from the perspective of an interacting particle system. The equation is given by:

$$u_t = Du_{xx} + ru - ru^2. \tag{23}$$

The first term is the diffusion term due to the random walk. The second term is the linear growth due to particle duplication. The last term is the (nonlinear) logistic term, which stems from the fact that exclusion leads to less growth in occupied regions compared to low density regimes. In addition, the logistic growth term only emerges in the continuum limit and is not satisfied microscopically. Hence, this system is a good initial test case for the capability of our data driven methods to find PDEs in complex systems. Since we have already found the Fisher equation from smooth data and tested the capability of our methods to tackle noisy data with the random walker, we expect that we will be successful in finding the Fisher equation.

Now, to derive this PDE, we need to assume that $\Delta x \to 0$. With our data-driven approach, we are essentially doing the same thing: we choose a small lattice spacing to approximate the gradients as derivatives. However, we cannot take an arbitrarily small lattice spacing, as that would give rise to very long simulations. We can only approximate the continuum limit by taking a small, but finite Δx. Hence, in this section we will also look into how small the lattice spacing must become in order to obtain a valid effective continuum description of the dynamics of the particle system.

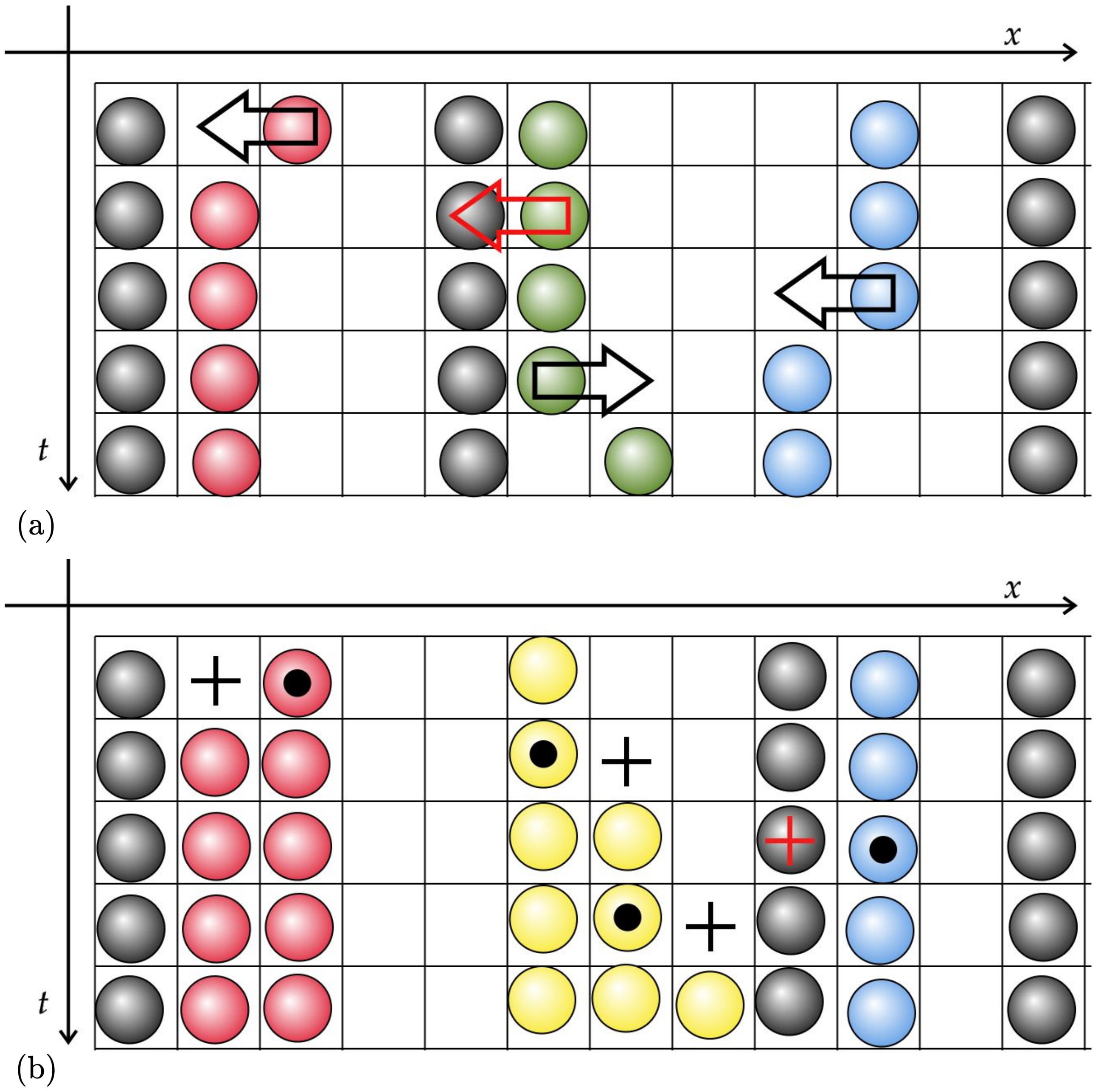


Figure 24: **(a)** Sketch of the Symmetric Simple Exclusion Process (SSEP), which models particle movement on a lattice, where x and t indicate the spatial and temporal direction, respectively. At each time step, particles move to the right and left based on the diffusion rate (black arrows). If the path is blocked by another particle, the particle stays put (red arrow). **(b)** Sketch of the Eden process rule set, where new particles appear based on a growth rate. At each time step, a particle is selected (black circle) to grow another particle in a neighbouring cell (black plus). If a particle already occupies a site, no new particles can grow on it (red plus). In this section we will combine the SSEP and Eden process, which can be interpreted as a model of a bacterial colony. Because this system contains diffusion and growth, we expect that for a small enough lattice spacing, the movement of the average population density is described by the Fisher equation.

Before we move on to the results, we must first reconsider how to obtain data that we can use for our data-driven methods. In the noninteracting random walk section, we simply stacked many particles on top of each other and then divided the occupation number by the total number of particles to obtain an average density. In this case, we cannot do this, because exclusion prevents sites to be occupied by more than one particle. This means that we cannot obtain a smooth density from a single simulation, because it would only contain zeros (empty sites) and ones (occupied sites). In addition, the number of particles is not conserved, because there is growth. Hence, we will use a slightly different approach compared to the random walker section. First we will draw an initial configuration from an occupation probability function: at each site there will be a given probability that it will become occupied or not. With this initial condition we then run the simulation from which we can obtain temporal data. We will repeat this process N times in a single script that makes use of one RNG seed. This allows us to ensemble average all lattice occupations to obtain a density field $u(x,t)$.

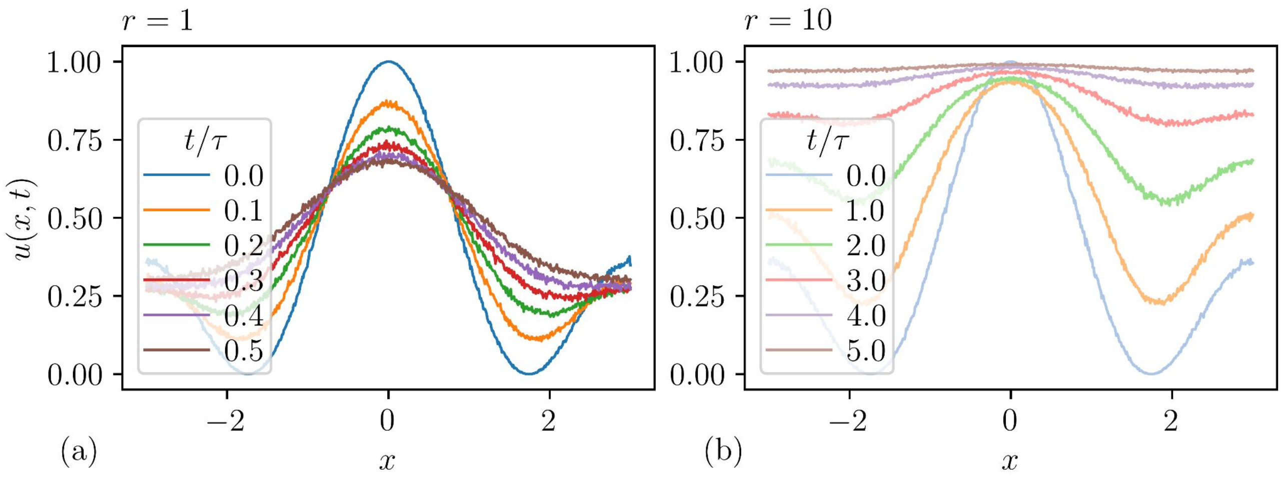


Figure 25: **(a)** We have generated data $u(x,t)$ of the SSEP+Eden system for a balanced growth to diffusion ratio, with input $D = 1$ and $r = 1$. We plot $u(x,t)$ as a function of space x with a lattice spacing of $\Delta x = 0.01$. As time increases, the densities spread out, indicating diffusion. In addition, the centre does not decrease as much as one would expect for a pure diffusive process, hence there is a non-negligible but visible growth. We thus expect a good agreement with the Fisher equation. **(b)** Similar plot, but now for a dominant growth of $r = 10$. The density goes to 1 as time increases, meaning that the lattice gets full. This would thus be an interesting case to investigate the difference of the input and output coefficient of the growth and diffusion rate.

In Fig. 25a we plot for $\Delta x = 0.01$, $N = 5000$, $r = 1$ and $D = 1$ the density $u(x,t)$ as a function of space x at different time steps. We see that the peak in the middle spreads out, indicating diffusion, but the growth process is hard to distinguish by eye. This process is better visible in Fig. 25**b**, where we show a similar plot for $r = 10$. Here, the density grows to the steady state of $u(x) = 1$, which we have also seen for the Fisher equation in section 3.1.

We will now analyse what the effect of Δx is on the growth process and how this compares to a solution of the Fisher equation. In Fig. 26**a** we compare the average density for $r = 10$ and $D = 1$ at a fixed time for different lattice spacings. In addition, we plot a solution u_f of the Fisher equation with the same coefficients and initial condition. We choose $\Delta x = 0.01$, but other lattice spacings yield similar curves. We see that the Eden-SSEP density with $\Delta x = 0.01$ lies slightly below the Fisher solution, but there is no exact match. Nevertheless, the behaviour is similar, so this supports our expectation that the `TrainSTRidge` algorithm has no problem finding the Fisher equation. We also see that increasing the lattice spacing makes the gap between the density and the solution bigger. Hence, the effective growth rate is decreased when the lattice spacing becomes bigger.

A useful way to quantify growth is by computing the total number of particles as a function of time. We can compare this for varying lattice spacings to quantify the error introduced by

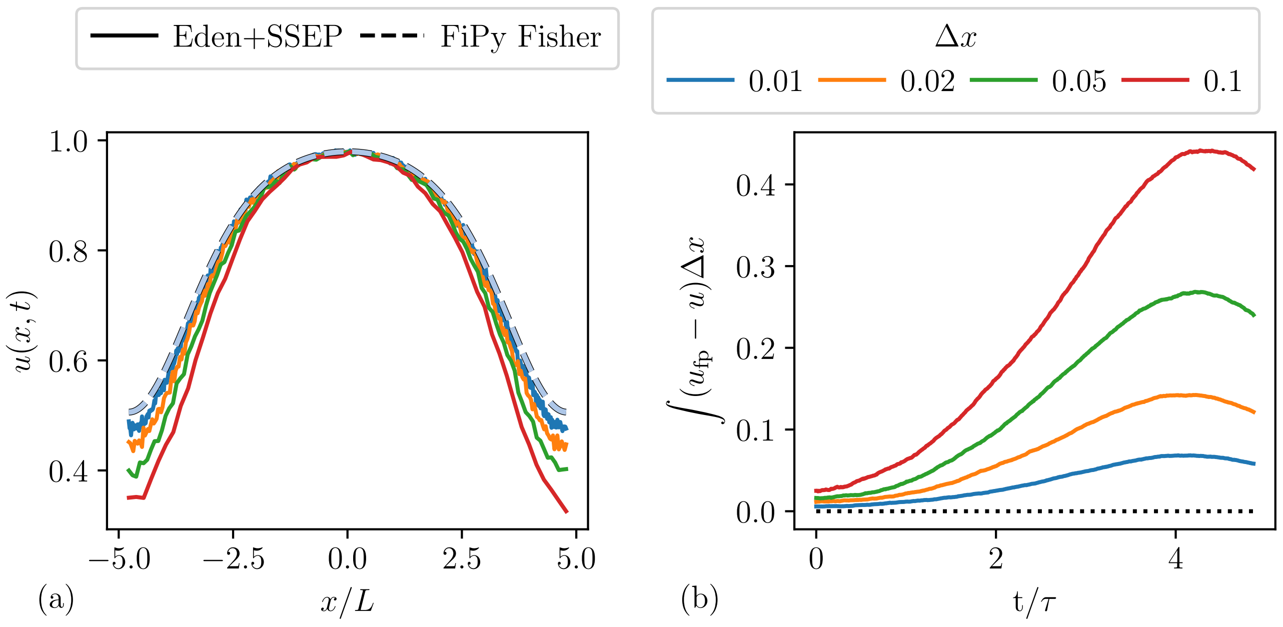


Figure 26: **(a)** We compare the density $u(x,t)$ of the SSEP+Eden system as a function of space x/L for different values of the lattice spacing Δx at time $t/\tau = 4.0$. The initial condition is a gaussian and the input growth and diffusion rate are $r = 10$ and $D = 1$, respectively. We also plot a single instance of the density for a solution of the Fisher equation, which we generated with Fipy for $\Delta x = 0.01$ and other input values similar to the SSEP+Eden system. We see that as Δx decreases, the SSEP+Eden density comes closer and closer to the Fisher solution. **(b)** We quantify the distinction seen in (a) by plotting the difference of the space integrals $\int(u_{\mathrm{fp}} - u)\Delta x$ as a function of time t/τ using Simpson's rule. Here, u is the density of the SSEP+Eden system and u_{fp} is the solution of the Fisher equation generated by FiPy with the same lattice spacing as u. Over time, the difference between the Fisher solution increases, and decreases again as the solutions reach the steady state of $u(x,t) = 1$. The difference are bigger for a larger Δx. This tells us that for high enough input growth rates, the finite size of the lattice spacing influences the effective birth rate of the particle system. Decreasing the lattice spacing diminishes this effect.

the discrete lattice. Hence, we will compute $\int(u_f - u)dx$ for each time step.[13] In Fig. 26b we plot the integral differences as a function of time. We see that over time, the differences first increase and then decrease as the densities reach a steady state. In addition, we observe that the difference is larger for bigger Δx. This confirms our observation that the effective growth rate depends on the lattice spacing.

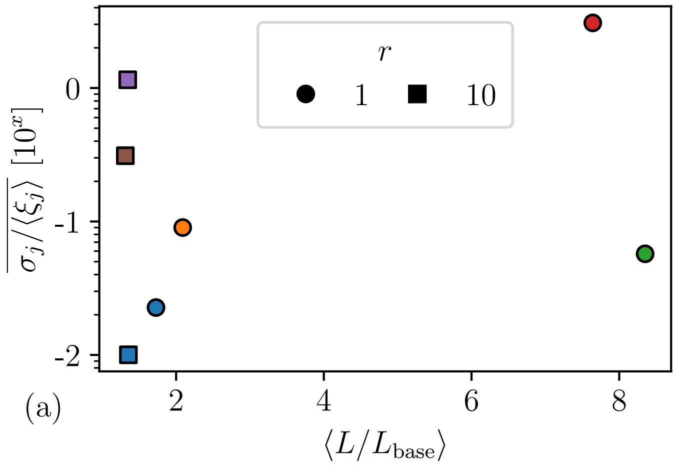
We also did a similar comparison with the $r = 1$ Fisher solution and $r = 1$ Eden-SSEP data for $\Delta x = 0.01$, but we did not observe any significant difference. This suggests that the Fisher equation assumptions also depend on the growth rate. That is, for a fixed diffusion rate, a higher growth rate requires a smaller lattice spacing to assume independence of the microscopic actions of the particles. We can understand this a bit better by considering the limits $r \to 0$ and $r \to \infty$. If $r \to 0$, then we only have diffusion, which works for every Δx after rescaling. So adding a little bit of growth will not affect the correlations of the lattice occupations. But if we let $r \to \infty$, then the process is dominated by growth. In this case, the effective growth rate will depend on Δx (provided we do not scale it).

Let us now specify the settings for the simulations and the `TrainSTRidge` algorithm. We will

[13]We use Simpson's rule to estimate the integrals, which works well for both noisy and smooth data.

focus on $r = 1$ and $r = 10$ with both $D = 1$. We used four initial conditions with $L = 6$, $T = 0.5$, $\Delta x = 0.01$ and $\Delta t = 10^{-3}$: (1) $\exp(-x^2/\sigma^2)$ with $\sigma = 0.9$, (2) $\cos(\omega x)$ with $\omega = 2\pi n/L$ and $n = 2$, (3) $\cos(\omega x) + \cos(\omega x/2)$ with $n = 2$ and (4) $\cos(\omega x) + \cos(\omega x/2) + \cos(\omega x/4)$ with $n = 4$. The latter two initial conditions are shifted so that the minimum probability is zero. We also divide the shifted function by the maximum so that the highest probability is 1. To obtain the densities we simulated 5000 runs and we repeated this for 100 different rng seeds. We Fourier filter the density with an amplitude cutoff $A_{\text{cut}} = 10^{-4}$. The spatial SG-width for each initial condition is $x_{\text{w}} = 1.01$, except for the fourth one, which is $x_{\text{w}} = 0.75$. In addition, for each initial condition we use $t_{\text{w}} = 0.101$ and $p_x = p_t = 5$. We will cut away the first and last temporal regime of the data by $t_{\text{cut}} = 0.05$.

The initial library $\vec{\Theta}$ will be the standard ansatz that we also used for the previous sections. For `TrainSTRidge`, we used the new error evaluation we outlined in section (-). In addition, we used 200 `TrainSTRidge` iterations, starting from $d_{\text{tol}} = 1$, and 36 `STRidge` iterations.



(a)

I	Right-hand side PDE
🟦 1	$\xi_1 u + \xi_2 u^2 + \xi_3 u_{xx}$
🟧 2	$\xi_1 u + \xi_2 u^3 + \xi_3 u_{xx}$
🟩 3	$\xi_1 u + \xi_2 u_{xx}$
🟥 4	$\xi_1 u + \xi_2 u_{xx} + \xi_3 u u_x + \xi_4 u u_{xxxx}$
🟪 5	$\xi_1 u + \xi_2 u^2 + \xi_3 u^3 + \xi_4 u_{xx}$
🟫 6	$\xi_1 u + \xi_2 u^2 + \xi_3 u^3 + \xi_4 u^5 + \xi_5 u_{xx}$

(b)

D	r	$\langle\xi_u\rangle$	$\langle\xi_{u^2}\rangle$	$\langle\xi_{u_{xx}}\rangle$
1	1	1.01±0.02	-1.02±0.04	0.999±0.006
1	10	9.85±0.06	-9.85±0.06	0.99±0.02

(c)

Figure 27: We have generated data $u(x, t)$ for the SSEP+Eden system for different seeds and initial conditions. Following the validation scheme using `TrainSTRidge` with the error evaluation adaption, we found multiple promising PDEs describing the dynamics of $u(x, t)$ for $r = 1$ and $r = 10$, which can be seen in table **(b)**. For each of these PDEs, we applied regression on each data set to estimate multiple average quantities, which we denote by $\langle\cdot\rangle$. **(a)** We scatter the mean error of the coefficients $\overline{\sigma/\langle\xi\rangle}$ versus the average squared error normalised by the baseline error $\langle L/L_{\text{base}}\rangle$. For both $r = 1$ and $r = 10$, the Fisher equation has the lowest $\overline{\sigma_j/\langle\xi_j\rangle}$, whilst maintaining a relatively low squared error. We can thus conclude that the SSEP+Eden system is best described by the Fisher equation. **(c)** We show the average coefficients $\langle\xi_j\rangle$ of the Fisher equation. The coefficients of $r = 1$ has an excellent agreement with the input values. For $r = 10$, the effective output growth is slightly below the input, implying that the birth has been slowed down. This is to be expected, since we only have a finite lattice spacing Δx. Furthermore, we see that $\langle\xi_u\rangle = -\langle\xi_{u^2}\rangle$, meaning that the PDE of the same form as the the Fisher equation.

With the above settings we applied the same general procedure we performed in the random walker section. In Fig. 27 we show the final results of the promising PDEs that we selected with pareto analysis. To obtain these results, we applied least squares on the selected PDE forms for each rng set and initial condition, which allowed us to calculate the averages $\langle\cdot\rangle$.

In (a) we scatter plot the error of the coefficients $\overline{\sigma_j/\langle\xi_j\rangle}$ versus the relative squared error $\langle L/L_{\text{base}}\rangle$ of each of the found PDEs. Here, L_{base} is the squared error computed by applying least squares with the full library $\vec{\Theta}$ onto the data. We see that there are two blue scatter points in the lower left corner, one for $r = 1$ and the other for $r = 10$. In table (b) we see that they correspond to the Fisher equation form. The other PDEs have a similar form, but have less consistent coefficients and have a higher squared error. Hence, with our data-driven methods we find that the Fisher equation is the most accurate and consistent continuum description of the Eden-SSEP model.

In table (c) we show the average coefficients for the Fisher terms. We see that for input $r = 1$ and $D = 1$, the output coefficients are within one standard deviation from the predicted values. This result supports the theoretical derivation that the average dynamics of the Eden-SSEP model is described by the Fisher equation. For $r = 10$, we find that the output growth rate is slightly lower than the input, which confirms our expectation that the output depends on a combination of the lattice spacing and the growth rate. Nevertheless, we see that $\langle\xi_u\rangle = -\langle\xi_{u^2}\rangle$, implying that the effective PDE is still the Fisher equation. This is promising, because it means that even with limited data (we only used 600 lattice sites), we can obtain accurate continuum descriptions of complex particle systems.

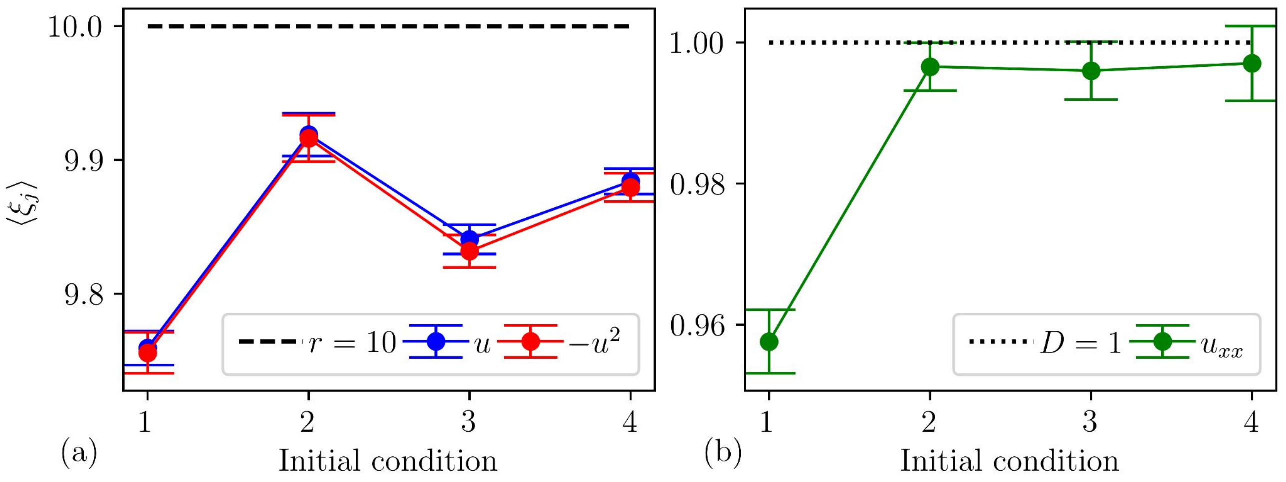


Figure 28: **(a)** We plot the average coefficient $\langle\xi_j\rangle$ for $r = 10$ of the growth terms inside the Fisher equation as a function of the initial condition. Here, $\langle\cdot\rangle$ denotes the average taken over all seeds *per* initial condition. The blue line is the coefficient of u and the red line is *minus* the coefficient of u^2. We see that for each initial condition, the blue and red lines are well within each other's standard deviations, implying that growth and counter-growth are consistent for each initial condition. This means that the Fisher equation is a very good approximation to these systems. We also see that the growth rate itself varies per initial condition. Hence, there is a local dependency that causes the effective growth to be lowered by a small amount. **(b)** We plot the same type of graph as in (a), but now for the measured diffusion rate. For each initial condition, we see that the measured diffusion rate is slightly lower than the input rate. This is likely caused by an interplay of the exclusion process, high birth rate and a finite lattice spacing.

Finally, let us examine to what extent the large-scale particle system dynamics depend on Δx and r by looking at the average coefficients per initial condition. Unsurprisingly, we find that for $r = 1$, the coefficients of the Fisher equation for each initial condition were within each other's standard deviation. This is not the case for $r = 10$, where the output coefficients do depend on the initial condition. In Fig. 28a we plot the average output growth rates as a function of the initial condition. We see that per initial condition, $\langle\xi_u\rangle$ is within one standard deviation from $-\langle \xi_{u^2} \rangle$. But we also see that the differences of the growth rates between initial conditions are larger than the deviations. In Fig. 28b we plot the average diffusion rate as a function of the initial condition. Here we observe similar behaviour as in (a), *i.e.*, the coefficients are lower than the input and have a local dependency. This means that there is a systematic error on the coefficients that is not captured by the variations we took. This implies that in this case, the effective growth and diffusion depends on the local structure of the system. This makes sense, because we found that the lattice spacing is not small enough to assume independency of the actions of the individual particles. That is, varying the initial configurations will give rise to different correlations, which will ultimately affect the long term behaviour.

Because this particle system is complex, it is hard to predict for which input diffusion and growth rate, the lattice spacing is 'small enough' to get the same output coefficients. Ultimately, it is a combination of all the microscopic details. For mild input coefficients, such as $D = 1$ and $r = 1$, it is sufficient to have $\Delta x = 0.01$. But if the growth-diffusion ratio increases, a smaller lattice spacing is required to get the exact same results as the mean field approximation. Nevertheless, it is reassuring that despite the possible local dependencies, we can still obtain a continuum description of the same functional form with (roughly) the same coefficients. This shows that the data-driven methods can successfully find the same type of large-scale dynamics of complex particle systems, while also being able to distinguish physical parameters. This should be especially useful in the contact process, because this system is even more complex than the Eden model with diffusion.

Before we move on to the CP, we first want to test if we can still obtain a continuum description if we only consider growth without diffusion. That is, we want to find out if we can find an effective PDE due to growth processes only. This is useful to know, because the CP also does not have a diffusion step in the model. So if there is a PDE that describes the dynamics of the CP, then it will be an emergent or effective PDE. Testing the capability of the data-driven methods to find effective descriptions for the Eden model is thus a useful way to prepare ourselves for the more difficult CP.

4.2 Travelling Waves of the Eden Model

In the previous section we verified that the Eden model combined with the SSEP gives rise to dynamics that obey the Fisher equation. To obtain the PDE, we had to explicitly add a diffusion process to the Eden model. In this section we will check if we can still obtain an effective PDE if we only have growth and no diffusion. Hence, this section should be seen as a set up for the CP, where there is also no diffusion.

Let us start by reasoning what sort of dynamics can occur if we have only growth processes. Suppose we have a population drawn from a gaussian distribution. In general, the centre of the population will have more particles than in the tail. In addition, there will be a number of empty sites between the particles at the boundary and the particles in the centre. In this system we can identify two processes. The first process is filling up the empty sites inside the population. The second process is the advancement of the populations at the boundaries. In general, the first process is faster than the second one, because there is simply a bigger chance that a particle grows inside the population than at the boundary. Hence, if we want to find an effective PDE we should either focus on the first or the second process; trying to capture both types of dynamics with a single PDE (probably) will not work.

Now, the aim is to find an effective PDE, that is, we want to find a continuum description that is dependent on the (local) spatial gradients. For the first process, this will be difficult, because the process will be dominated by linear growth and exclusion, which are ODE terms. However, for the second process we expect that in the long time regime, the population will spread out at a constant speed, similar to advection. See Fig. 29a for a sketch of the situation. Moreover, this process is symmetric, because a population can grow to the left and right at the same time. Based on these heuristic arguments, we expect the following effective PDE:

$$u_t = v|u_x|, \tag{24}$$

where $v = r\Delta x/2$ is the speed of the population. The factor of a halve is there because particles grow to the left and right with rate $r/2$. The absolute value causes the density to increase on both sides of the population, such that the symmetry of the process is satisfied.

Now, if we simply start from a single particle as an initial condition, then the moving front of the boundary will be discontinuous from which we cannot compute a derivative (with our methods). Hence, to find this PDE we need appropriate initial conditions that will result in smooth travelling waves at the boundary of the population. In addition, we would like to vary the initial slope of the starting configurations, to see whether it has any effect on the average behaviour. For this end, we will use $\tanh(ax)$, because we can easily vary the slope a and we can easily manipulate it to obtain a symmetric initial condition: if we have $x \in [-L/2, L/2]$, then the initial condition is simply

$$u(x) = \begin{cases} \frac{1-\tanh(a(x-x_0))}{2} & \text{if } x \geq 0 \\ \frac{1+\tanh(a(x+x_0))}{2} & \text{if } x < 0 \end{cases}, \tag{25}$$

where x_0 is the first position where $\tanh(ax) > 0.999$.

With this initial condition we can generate data. We will use $a \in \{3, 5, 7, 9\}$ (so four initial conditions), $\Delta x = 0.01$, $L = 8$, $N = 5000$, and $r \in \{0.1, 0.5, 1, 5, 10, 20\}$. The run times T are chosen such that the tail of the travelling wave with $a = 3$ reaches the boundary. The time interval is $\Delta t = T/N_t$, where $N_t = 500$ is the number of temporal data points. We repeat the runs 10 times with different rng seeds.

In Fig. 29b we plot for $a = 3$ and $r = 1$ an example of the average density as a function of space for different times. We see that the boundaries of the populations shift away from the centre,

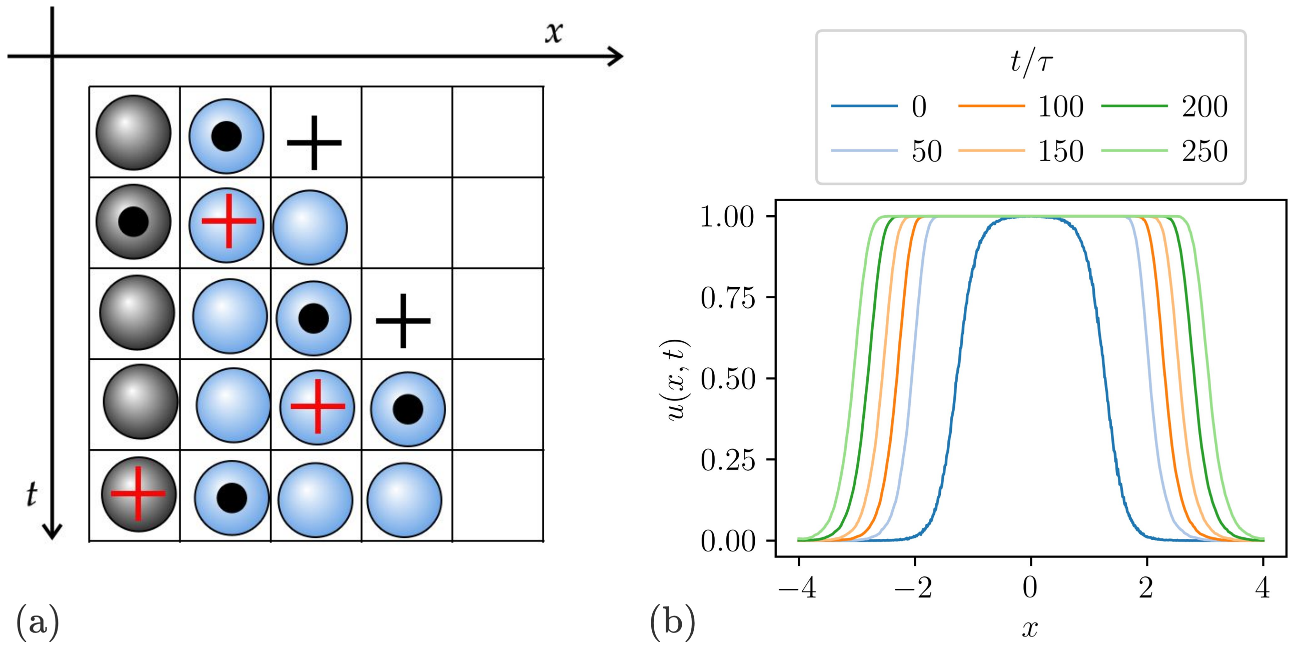


Figure 29: **(a)** Sketch of the Eden process at the boundary of the population. In this situation, growth can only occur to the right for the rightmost particle. **(b)** We plot the average density $u(x,t)$ generated by the Eden process with birth rate $r = 1$ as a function of space x. The initial condition is a centred population with slopes of a tangent hyperbolic function and $\Delta x = 0.01$. We see that in the first time regime, the tail of the initial population has fully grown. After that we get travelling wave behaviour corresponding to the scenario we have sketched in (a). Also note that the gradients are smooth, meaning that we can compute reliable derivatives for the library $\vec{\Theta}$. Finally, we see that the travel wave behaviour is symmetric, *i.e.*, the population expands to the left and to the right. Hence we expect symmetric terms like $|u_x|$ to be important additions to the library.

with continuous spatial gradients. In addition, we see that between $t/\tau = 0$ and $t/\tau = 50$ the boundaries have shifted considerably more than for later times. This difference can be explained by the two growth processes we described earlier. In the short time regime, the lattice between particles fills up, while there is only growth at the boundary in the latter time regime. Hence to measure Eq. 24, we will remove the initial time regime of the data. More concretely, we wait until the rightmost point of the boundary at $t/\tau = 0$ (so the rightmost position x_r where $u(x,0) < 0.01$) has reached the steady state density (so the first time t_r where $u(x_r,t) > 0.99$). So we only use temporal data after t_r. This makes sure that the physics is dominated by the long time regime behaviour of the Eden process.

With this data, we can apply our data-driven methods to analyse the average behaviour. In the library $\vec{\Theta}$ we included the basis set terms with powers and derivatives up to 2. In addition, we will add $|u_x|$, $|u_{xx}|$, $(u_x)^2$, and $(u_{xx})^2$. For the spatial derivatve we estimate the SG-width based on the length scale of the travelling wave front. That is, for every time t_j, x_w is the length between the first positions where $u(x,t_j) > 0.99$ and $u(x,t_j) < 0.01$ for $x > 0$. This improved the accuracy of the derivatives for latter times. With these settings, we found that applying `TrainSTRidge` for a relative low threshold in general yields a sparse PDE of the from $u_t = \xi_0 u - \xi_1 u^2 + \xi_2|u_x|$. Increasing the threshold removes the logistic terms, which results in $u_t = v|u_x|$. In Fig. 30a we compare u_t and $v|u_x|$ as a function of space and different times. We see that $|u_x|$ captures the behaviour of u_t, including the widening of the peak. In Fig. 30b we plot the measured speeds divided by the lattice spacing as a function of r. We find that there is an excellent match with the predicted value of $v = r\Delta x/2$. Hence, with our data-driven methods we can successfully find effective PDEs for growth processes without diffusion, which should be especially useful for the CP.

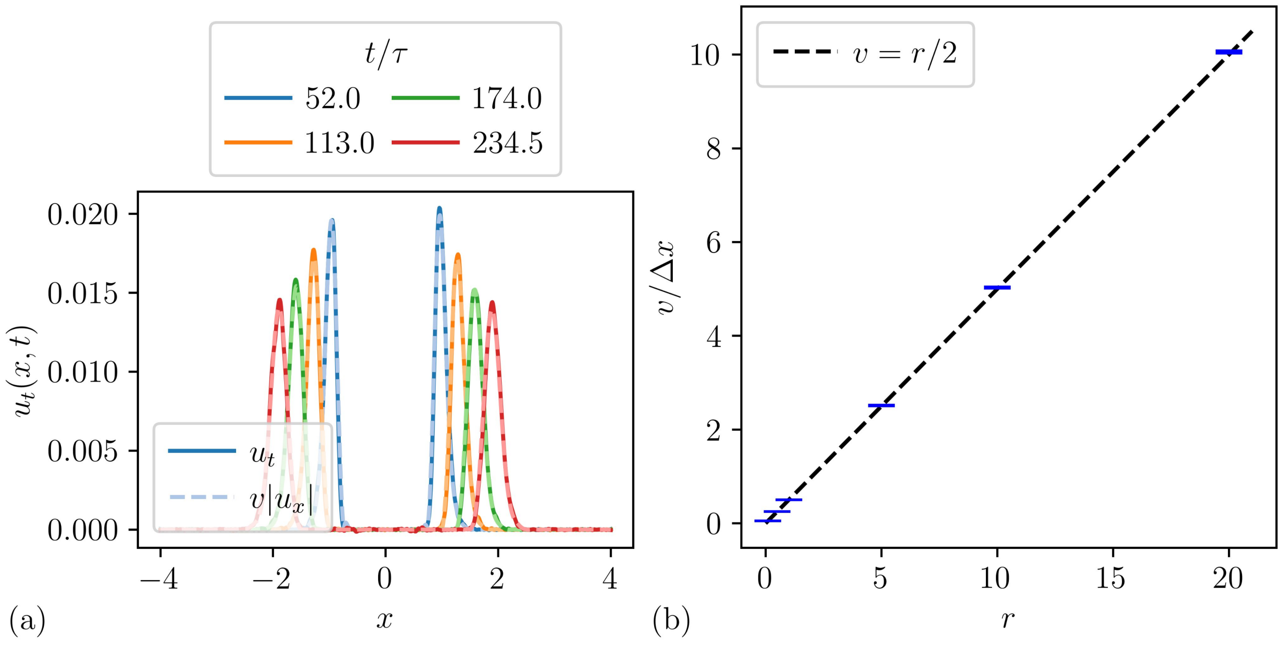


Figure 30: **(a)** We plot the time derivative $u_t(x,t)$ as a function of x for different instances of time t/τ. Here, $u(x,t)$ is a density generated by the Eden process with $r = 1$ and $\Delta x = 0.01$. We also plot the absolute advection term $v|u_x|$, where we measured the speed v using regression. We see that there is a good match between the two curves. **(b)** We plot the measured speed of the travelling waves divided by the lattice spacing $v/\Delta x$ as a function of the growth rate r. The measurement is done for ten different seeds and four initial conditions, which we averaged over. We see an excellent agreement with the theoretical value of $v = r/2$. Both graphs gives us confidence that the PDE $u_t = v|u_x|$ we found with `TrainSTRidge` is indeed an excellent description for the travelling wave behaviour of the Eden process.

4.3 The Contact Process

In the previous sections we have examined the Eden model (with and without the SSEP) to find (effective) PDEs of particle systems inhabiting growth processes. In this section we are going to add our final layer of complexity to the model, namely particle death. The particle model that has both growth and death is called the contact process (CP) and it has a phase transition based on the growth rate, as discussed earlier. In Fig. 31 we sketch the rules of the CP. These are for the most part the same as for the Eden model. To incorporate decay, we simply add the rule that occupied sites become vacant with a death rate δ. We set this rate to $\delta = 1$, which we will also use as a time scale of the model. With this rate, the critical birth rate is $r_c \approx 3.297848$. Every initial configuration below this critical point will go extinct with probability 1. Above criticality, there is a positive probability that an initial configuration will survive. In that case, the behaviour is similar to the Eden model in the sense that the system will keep on growing. At the critical point, the number of particles grow polynomially with a critical exponent, but it also has been proven that each initial configuration goes extinct with probability 1 as $t \to \infty$ [22]. A more in depth overview of the CP can be found in Ref. [42].

We thus see that adding a simple rule to the Eden model has a huge impact on the overall behaviour of the system. In the Eden model the complexity arises from the interaction of particle growth and exclusion. But by adding particle death, there is also an interplay between growth and decay that causes the system to have different macroscopic properties depending on the ratio of the rates. In this section (and in the next one), we will be examining how this complex interplay exactly affects the average behaviour of the CP. That is, we want to find out if we can describe the average dynamics of the CP in the form of a PDE. This allows us to ascertain if there are emergent properties of the CP that we can capture with a PDE, *i.e.*, whether there is a unique interplay between growth and death that leads to certain types of terms in the PDE that could help explain the observed dynamics.

For this we will analyse two regimes of the CP, namely above the critical point and at the critical point. We will first examine the CP above criticality, as it serves as a connective bridge between the semi-complex behaviour of the Eden model and the more complex behaviour of the CP at criticality, of which the latter will be the focus in the final part of this section.

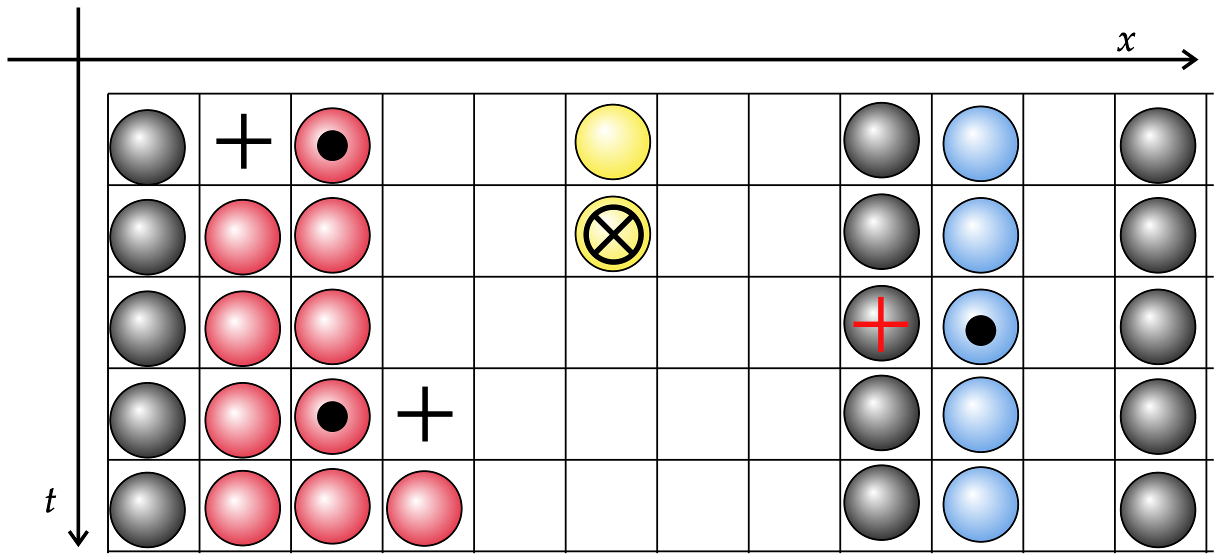


Figure 31: Sketch of the CP, which is a stochastic lattice particle system. Particles can grow on neighbouring sites with rate r (black plus) and they can perish with rate δ (black cross). There is also exclusion, so particles cannot grow on occupied sites (red plus).

4.3.1 Travelling Waves of the Contact Process above Criticality

Above the critical point, the CP is dominated by growth. Similar to the Eden process, there is a nonzero steady state density. However, due to the decay of particles, this density is not 1, but below it. So if the process has reached this density, we expect that at the boundaries, the long term behaviour of the average dynamics will be similar to the Eden model. The reason for this is as follows. Let us consider a situation similar to Fig 29a, where the lattice is filled up, except at the boundaries. In this case, the lattice occupation in the centre of the population will decrease a bit due to decay, but on average, this occupation will be stable. Hence, the only new changes occur at the boundary of the population. Here, the population expands similar to the Eden model, but since the right- and leftmost particles also have a probability to die, this expansion is slowed down. So on average, we expect that the same behaviour occurs, but with a net speed that is lower than the Eden speed. It will thus be interesting to see how the propagation speed depends on the growth rate, especially in the neighbourhood of the critical point.

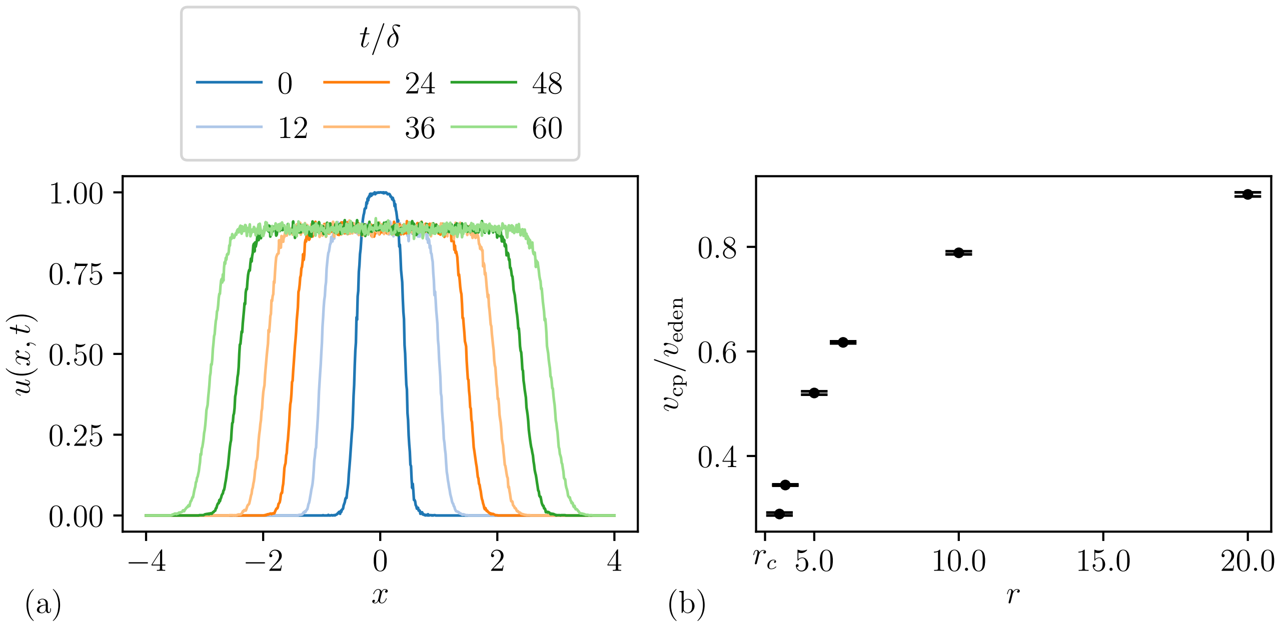


Figure 32: **(a)** We plot the average density $u(x,t)$ as a function of space x for different times. We use $N = 1000$, $\Delta x = 0.01$ and $r = 10$ (so above the critical point). The CP shows similar behaviour as the Eden process: the population spreads out at the boundaries. The main difference is that the steady state density is that the steady state density is below 1, due to the death process. Similar to the Eden process, we found that after the initial population at the boundary has reached its steady state, the dynamics can be described by $u_t = v|u_x|$. **(b)** We plot $v_{\mathrm{cp}}/v_{\mathrm{eden}}$ as a function of the growth rate r. Here, v_{cp} is the average measured speed of the CP populations, which we generated with 10 different rng seeds. We computed this speed via a least squares fit on $u_t = v|u_x|$. $v_{\mathrm{eden}} = r\Delta x/2$ is the expected speed in the case that the death rate is zero. We see that near the critical point r_c, the speed of the population decreases compared to v_{eden}. As the growth rate increases, the speed gets closer v_{eden}. Hence, for $r \to \infty$, we expect that $v_{\mathrm{cp}} \to v_{\mathrm{eden}}$, since the growth will dominate over the death process. As $r \to r_c$, we expect that $v_{\mathrm{cp}}/v_{\mathrm{eden}}$ will decease even further, but it is hard to make predictions about the actual value, since $u_t = v|u_x|$ might not be a good description of the dynamics at that point.

For the data-driven approach, we will generate data using the same initial conditions and settings as in section 4.2. The only difference is the growth rates: $r \in \{r_c + 0.5, 4, 5, 6, 10, 20\}$. In Fig. 32 we plot an example density of the CP above criticality as a function of space for different times. We see that after $t/\delta = 0$, the density at the centre of the population quickly shrinks to a steady state. At the boundary, the density expands to the left and right, which is the same behaviour we saw in the Eden model.

To analyse the long term travelling wave behaviour (and not the initial time regime), we apply the same procedure as in the Eden section, *i.e.*, we wait for the tail of the population to grow to the steady state density before we start applying our data-driven method.[14] We found that this process is slower than the decay of the centre to the steady state, so we do not need to wait for the latter process to occur before we can start the measurement.

Similar to the Eden model, we found that in the measured time domain, the continuum description could be described by $u_t = v|u_x|$. In Fig. 32b we plot the average measured speeds v_{cp} (divided by the theoretical Eden speed $v_{\text{eden}} = r\Delta x/2$) as a function of the growth rate. We see that near the critical point, v_{cp} is considerably smaller than v_{eden}. As r increases, v_{cp} grows relatively closer to v_{eden}. This makes sense, because the death contribution becomes insignificant when $r \to \infty$, such that $v_{\text{cp}} \to v_{\text{eden}}$. Hence, we see that above the critical point, the average behaviour of the CP is comparable with the Eden model.

Now, when $r \to r_c$, it becomes slightly harder to predict what the actual speed will be. However, we do know that the steady state will be zero as $t \to \infty$. Hence, there will never be a travelling wave of the exact same behaviour as above the critical point.

[14]To obtain numerical values for the steady state, we measured (in advance) the average density at the centre of the population for times where the density stopped changing. We did this for each of the growth rates

4.3.2 Effective PDE from the Contact Process at the Critical Point

At the critical point of the CP, it is hard to deduce what the complex interactions between death and growth will bring about for macroscopic behaviour. Let us show why. In Fig. 33 we show two examples of the lattice occupation starting from a configuration drawn from the tangent hyperbolic initial condition with $a = 9$. We simulated the runs for $t/\delta = 500$ and used $\Delta x = 0.01$. In (a) the system goes extinct before the simulation has ended, while in (b), some trees keep on growing (and spreading) until the end of the simulation. We thus see that it is very hard to predict what the outcome will be based on the rules alone; there is a stochastic element to the survivability of the CP system. But if we are going to ensemble average the occupations, then the stochasticity should be averaged out, which will lead to a better tractable problem and gives us important information about macroscopic properties. Of course, due to the many particle deaths, the resulting density will be more noisy than a density obtained from the Eden model. Hence, we will use $N = 10^5$ runs to make sure we obtain (relatively) smooth data that captures many aspects of the CP at the critical point.

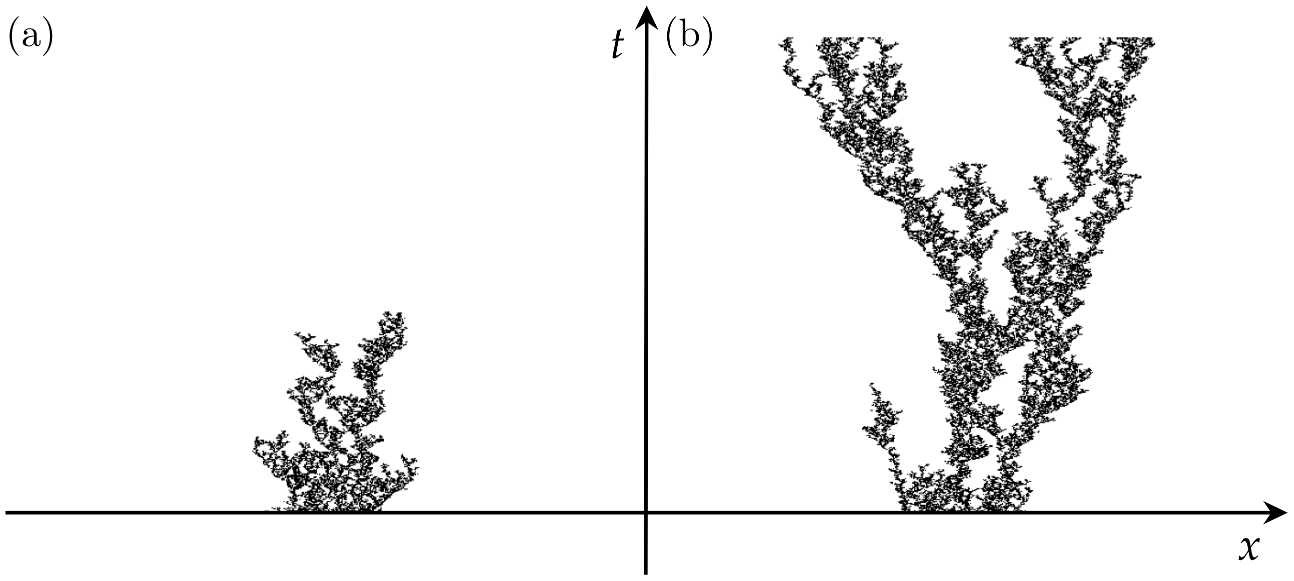


Figure 33: **(a-b)** We plot the 1D lattice occupation of the CP at criticality as a function of space x and time t. We simulate the system for $t/\delta = 500$ with $\Delta x = 0.01$, starting from the initial configuration drawn from Eq. 25 with $a = 9$. Black sites are occupied, while white sites are vacant. In **(a)** we see that the system dies out before the simulation has ended. In **(b)** the system survives at the end of the simulation, with different branches spreading out. This behaviour highlights the complexity and stochastic nature of the model.

Before we move on to the collected data, let us emphasise that the resulting average behaviour will represent the dynamics of all the populations combined instead of a continuous description of a single population. After all, configurations may go extinct and they can have branches with large patches of vacant sites between them. As such, the average dynamics will be an effective continuous density that does not describe individual populations. Hence, the density is perhaps better interpreted as a probability that a particle will occupy a site on a certain position at a certain time. This also means that any PDE that we obtain with our methods should be considered as an effective continuum description instead of a stochastic model that can replicate the dynamics of a single simulation, such as a stochastic PDE. With that out of the way, let us now discuss the data generation.

Similar to the previous two sections, we will consider initial conditions that are distributed around the centre of the lattice, such as a gaussian function or the function from Eq. 25. This allows us to examine the boundary behaviour and compare it to the CP above the critical point.[15] These will be (1) $\exp(x^2/\sigma^2)$ with $\sigma = 0.5$, (2) the tangent hyperbolic from Eq. 25 with $a = 5$ and (3) $a = 9$ for $x \in [-L/2, L/2]$. We will choose $L = 6$, $\Delta x = 0.01$, $T/\delta = 500$, $\Delta t/\delta = 1$ and we repeat the data generation 25 times with different rng seeds.

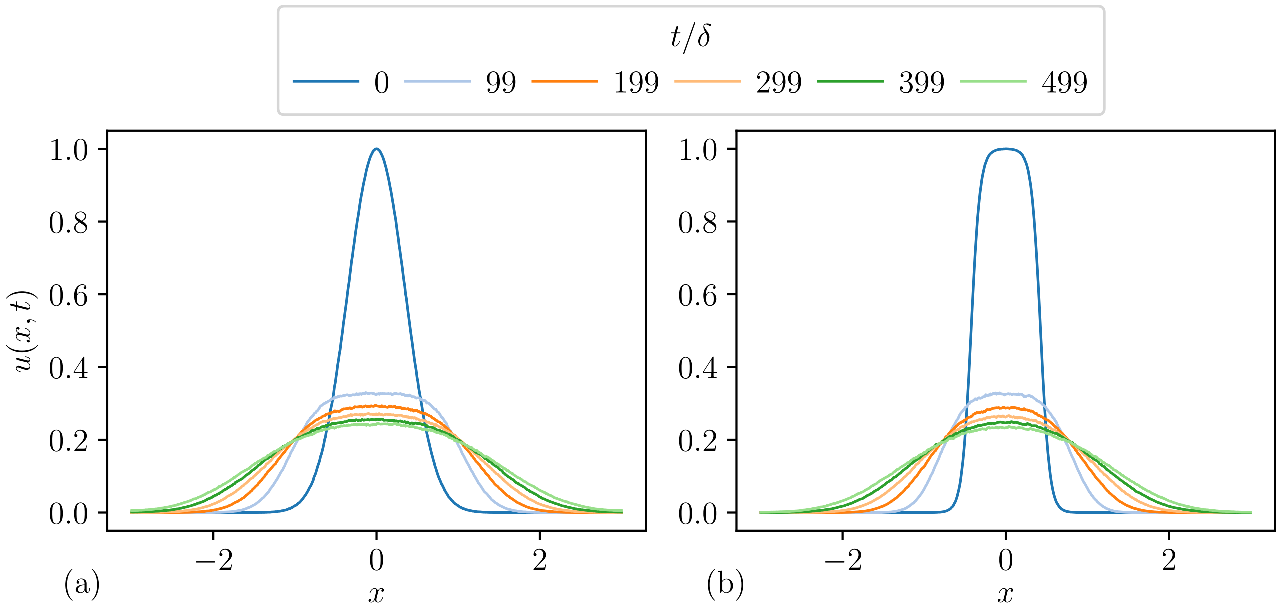


Figure 34: **(a-b)** We plot the average density $u(x,t)$ of the CP with $r = r_c$ as a function of space x for different times. The initial condition is **(a)** a gaussian and **(b)** a manipulated $\tanh(ax)$ (see Eq. 25). We used $N = 10^5$ and $\Delta x = 0.01$. We see that in the first time regime, the density of the peaks shrink considerably and attain a similar shape. After that the densities slowly spread out, with increasing length scales at the boundaries of the population. This indicates that the dynamics are dependent on the gradients, which suggest the possibility of an (effective) PDE that can capture the change in gradients with a spatial derivative term.

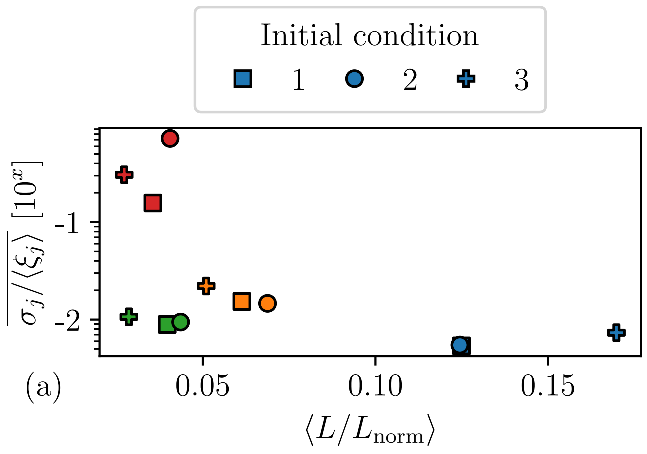
In Fig. 34a we plot an example of the average density as a function of space for different times with the gaussian initial condition. In (b) we plot it for the tangent hyperbolic. We see that in both plots, the peak of the average density shrinks much faster in the first time regime, compared to latter times. This is to be expected, because many particles will die before there the density settles to a more 'equilibrated' state that reflects the death to growth ratio better than the initial configuration. We also found that this decay is too fast to be captured accurately for the chosen time step. Hence, for `TrainSTRidge` we will exclude the first 50 time steps of the data.[16] We also see that the density spreads out at the boundaries as time goes on. To be more specific, the gradients becomes less and less steep, such that the length scale of the interface increases.

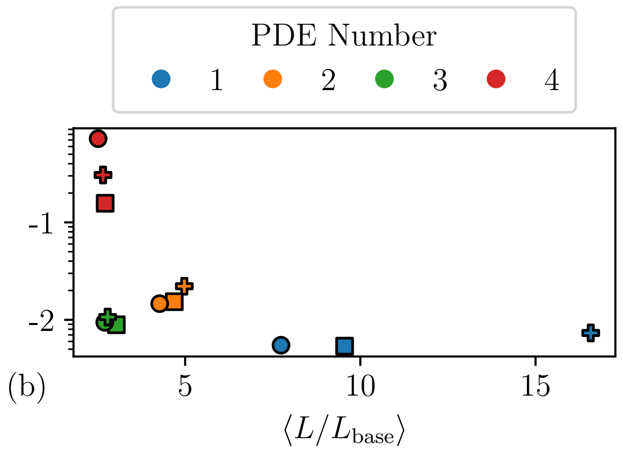
This suggests that it is useful to include interface broadening terms in the ansatz PDE, such as u_x^2 or uu_x^2. Hence, our library $\vec{\Theta}$ will consist of the following terms: $u^q(u_{x^p})^k$, with $q, k \in \{0, 1, 2\}$ and $p \in \{1, 2\}$. We will not consider higher order terms, because terms such as u^3 and u^4 will introduce a lot of biased overfits and we do not expect these terms to be that relevant based on what we have seen from the Eden model and the Fisher equation. Of course, since the system is critical, it could even be that the exponents are fractional. So in principle, we could add

[15] We also looked into functions such as cosines, but we found slightly different average behaviour, which we did not want to focus on.

[16] We examined the first time regime with high temporal resolution as well, but we did not find any sparse PDEs, only overfits.

an expansion of different fractional powers based on the critical exponents of the CP. But this complicates the library quite a bit and we first want to study the system with a relatively simple ansatz which should at least give us directions as to what terms in the PDE are important.[17]





(c)

I	Right-hand side PDE
🟦 1	$\xi_1 u + \xi_2 u^2 + \xi_3 u_{xx}$
🟧 2	$\xi_1 u + \xi_2 u^2 + \xi_3 u_{xx} + \xi_4 \lvert u_x \rvert$
🟩 3	$\xi_1 u + \xi_2 u^2 + \xi_3 u_{xx} + \xi_4 u_x^2$
🟥 4	$\xi_1 u + \xi_2 u^2 + \xi_3 u_{xx} + \xi_4 u_x^2 + \xi_5 u_{xx}^2 + \xi_6 u u_{xx}^2$

(d)

I.C.	$\langle\xi_u\rangle$	$\langle\xi_{u^2}\rangle$	$\langle\xi_{u_{xx}}\rangle$	$\langle\xi_{u_x^2}\rangle$
🟩 1	$(2.70 \pm 0.03)\cdot 10^{-3}$	$(-1.25 \pm 0.01)\cdot 10^{-2}$	$(3.87 \pm 0.02)\cdot 10^{-4}$	$(2.34 \pm 0.03)\cdot 10^{-3}$
🟩 2	$(3.05 \pm 0.03)\cdot 10^{-3}$	$(-1.39 \pm 0.01)\cdot 10^{-2}$	$(3.74 \pm 0.03)\cdot 10^{-4}$	$(2.14 \pm 0.03)\cdot 10^{-3}$
🟩 3	$(2.69 \pm 0.03)\cdot 10^{-3}$	$(-1.26 \pm 0.01)\cdot 10^{-2}$	$(3.81 \pm 0.03)\cdot 10^{-4}$	$(2.22 \pm 0.03)\cdot 10^{-3}$

Figure 35: **(a-b)** We plot the error of the coefficients $\overline{\sigma_j/\langle\xi_j\rangle}$ of found PDEs (see table **(c)**) as a function of **(a)** $\langle L/L_{\text{tot}}\rangle$ and **(b)** $\langle L/L_{\text{base}}\rangle$, where L is the squared error of the PDE, L_{tot} is error normalisation based on the mean of the time derivative and L_{base} is the squared error of the initial ansatz PDE. We see that PDE 3 has the best pareto tradeoff between the errors. This PDE is the Fisher equation with $(u_x)^2$ added to it, which explains the observed length scales. This shows that at the critical point of the CP, there is emergent behaviour that can be captured by an effective PDE. In table **(d)** we compare the average coefficients between the initial conditions. The coefficients are close to each other, but there is a small offset that is larger than the standard deviations. This is not surprising, since this PDE only works after a certain time has passed. Hence, each initial condition might be in a slightly different state depending on the time from which we start measuring.

To compute the derivatives, we applied the 2D Fourier filter with $A_{\text{cut}} = 10^{-5}$. Nevertheless, we found that the temporal data was still noisy. Hence, for the temporal derivative we chose degree 2 for the SG-filter with width $t_{\text{w}} = 51$. For the spatial data we used degree 6 and we estimated the width based on the length scale of the interface, similar to what we did with the Eden model. With this data, we first applied `TrainSTRidge` with the error evaluation method from section 2.5.2. We used 200 `TrainSTRidge` and 36 `STRidge` iterations with $d_{\text{tol}} = 10^{-3}$. The reason why we set d_{tol} to a small number is because it would otherwise cut away all terms

[17]To add to this, we found out that we could overfit the PDE with this full (but relatively small) library. Hence, adding more terms would not make a difference at first glance.

in the first iteration. This is because the normalised contributions are relatively in the choosen regime. With these settings we found a various number of PDEs. We selected the four most common and promising PDEs using the pareto error analysis. These can be seen in Tab. 35c. They consist of the Fisher equation (with different coefficients for u and u^2) and extensions of the Fisher equation, with terms such as $|u_x|$ and $(u_x)^2$. This is peculiar, because the CP does not have an (intrinsic) diffusion step. Hence, the diffusion is an emergent property of the system. But before we draw any further conclusions, let us first examine which of the found PDEs is the best candidate.

We applied least squares on each PDEs for each data set and initial condition so that we can compare the average errors. In Fig. 35a we scatter plot for each PDE (and initial condition) the mean error on the coefficients as a function of the normalised squared error. We see that the Fisher equation has the smallest standard deviations but the largest squared error. To estimate how well the PDE compares to the PDE computed with the entire basis set, we plot in (b) the same data, but now as a function of the squared error divided by the baseline error. We see that this error is around 10 and higher, which is relatively big. Of course, this is to be expected, because we do not observe a travelling wave solution with a constant length scale, which would be the case if the CP data would obey the Fisher equation. All the other PDEs are extensions of the Fisher equation, of which the one with $(u_x)^2$ has the best pareto tradeoff. It has a squared error that is around three times as big as the baseline error. In addition, this term explains the observed interface broadening: it pushes the density at the centre of the interface more upwards than at the edges, causing the boundaries to become more and more elongated in combination with diffusion.

We also see that this PDE competes with the one containing $|u_x|$. This makes sense, because $|u_x|$ is similar to $(u_x)^2$. At first glance, it is perhaps strange that there is a preference for $(u_x)^2$, given that we could describe the CP above criticality with $|u_x|$. However, the argument that the population grows somewhat like the Eden process does not hold at the critical point, so we should not interpret $|u_x|$ as a propagation term with a certain speed. In that regard, we can conclude that at $r = r_c$, we obtain $v_{\text{cp}} = 0$ in Fig. 32, because the preference for $(u_x)^2$ suggests that there is a different process going on than propagation.

Finally, let us examine how consistent and reliable the resulting PDE is. In Tab. 35d we show the average coefficients for each initial condition. We see that each coefficient lies close to each other, but with a difference that is slightly bigger than a few standard deviations. This suggests that there is a systematic error that causes the (average) configurations to behave slightly differently. One reason for this might be the extinction events, which may be dependent on the initial condition.

To test how well this PDE can describe the average dynamics of the CP, we numerically solve the PDE with `Fipy` and compare it to the CP density. To initialise the solution, we use the same lattice and time step. To obtain an initial condition, we filter and smoothen the CP density at the time where we started measuring the PDEs (so at $t/\delta = 50$). In Fig. 36 we plot for the same initial conditions as in Fig. 34 the solution / CP density as a function of space for different times. We see an excellent match between the two curves for both initial conditions. Only at the end we see a small discrepancy: the solution curves around the CP lines, which suggests that the PDE is a very good approximation, but not an exact match. We also solved and compared the PDE at $t/\delta = 0$. We found that the solution reacted much slower than the CP density. Hence, the effectiveness of the PDE does depend on the time regime of the CP. This could also explain why there is a difference in the coefficients per initial conditions. If at $t/\delta = 50$ the configuration is farther away from the initial transient, then this could result in slightly different behaviour and coefficients.

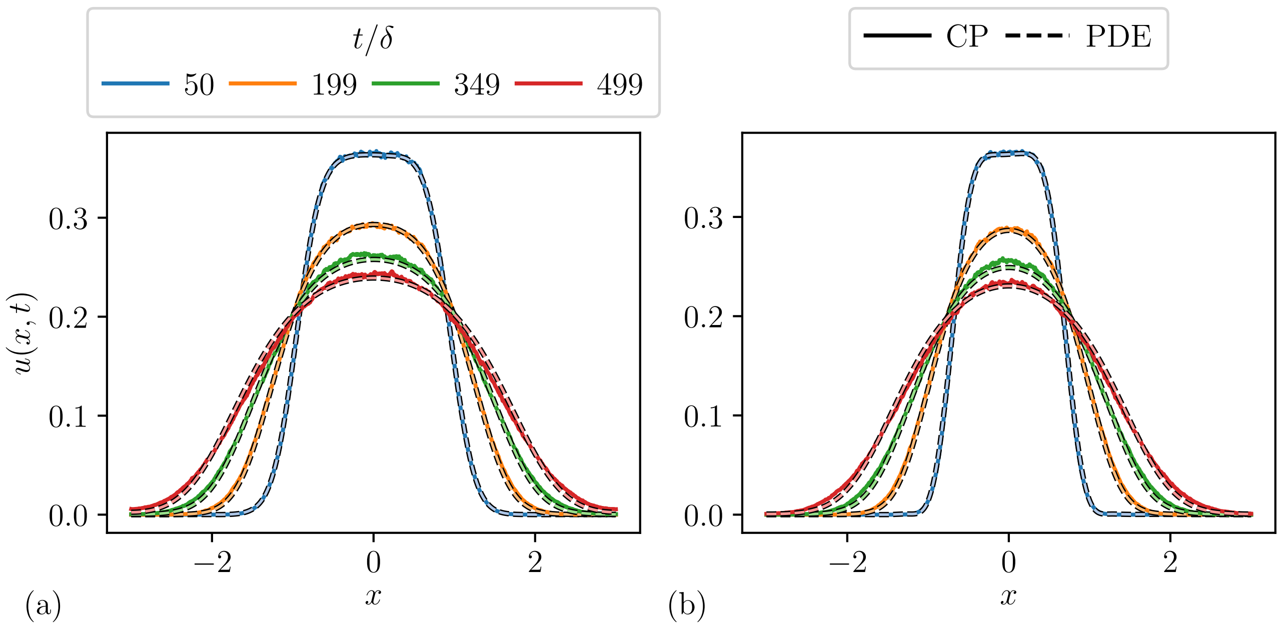


Figure 36: **(a-b)** We plot the average CP density $u(x,t)$ as a function of space x for different times and compare this to the solution of the found PDE (dashed lines). The initial conditions are the same as in Fig. 34, but we start from $t/\delta = 50$, which is also the time from which we started measuring the PDE. The initial condition for solving the PDE is a filtered and smoothed density of the CP. We see that there is a good match, with a slight offset for latter times. Hence, this PDE is a good effective description of the dynamics.

In the next section we are going to discuss these results, including those from the other interacting particle systems.

5 Discussion

In this thesis, we have found various PDEs for interacting particle systems, each one describing how growth (and/or diffusion) can lead to large-scale movement of particles. We will first discuss if these PDEs are to be expected and how they relate to other literature.

The first interacting particle system that we examined is the Eden model (particle growth into vacant neighbouring sites) combined with the symmetric simple exclusion process / SSEP (particle displacement with exclusion). For this system, we numerically verified that for suitable choices of input parameters, the dynamics of the system is described by the Fisher equation. Specifically, we found an excellent match with the literature for growth rate $r = 1$, diffusion rate $D = 1$ and lattice spacing $\Delta x = 0.01$. Changing the growth rate to $r = 10$ resulted in the Fisher equation as well, but the Fisher coefficients for different initial conditions were statistically inconsistent with each other, and the expected value. We can reconcile this result with the literature based on the assumptions required to derive the Fisher equation. A small enough lattice spacing is needed to obtain the PDE, and we indeed found that increasing/decreasing the lattice spacing improved/declined the accuracy of the Fisher equation. We thus see that a finite-size lattice spacing limit towards the continuum can still lead to accurate measurements of the PDE. This information could be useful for experimentalists, who also try to infer PDEs from finite data.

We have also studied the pure Eden model. We found that we could describe the dynamics at the edges of the growing colony as a travelling wave $u_t = v|u_x|$, where the speed is linear in r. Similar to the Eden-SSEP system, we found that there is a direct link with the microscopic and macroscopic parameters of the system. This allows for accurate measurements of important physical parameters, such as the speed of the travelling wave, the growth rate, and the diffusion constant (provided there is diffusion) from data. Of course, the Fisher equation also has a travelling wave solution, where the speed is proportional to $\sqrt{rD}$. For future testing, we could thus look into fitting $u_t = v|u_x|$ on the Eden-SSEP model, because it is not clear if the Fisher equation also holds in the regime where $D \ll 1$ and $r \gg 1$, which are needed to generate travelling wave behaviour. If successful, the data-driven methods would be a promising contender for measuring travelling wave behaviour in experimental systems, such as growing bacterial colonies [25].

The next particle system we considered is the basic contact process (CP), which is an extension of the Eden model: particles also have a probability to perish. The onset of death gives rise to critical behaviour, which means that the CP is one of the simplest particle systems that has a phase transition [42]. The elegant simplicity of this model gave rise to many studies on the critical behaviour, but there are still some open questions: no coarse-grained PDE of the basic CP at criticality has been proposed in the literature, only a stochastic PDE for directed percolation (DP), which is similar and in the same universality class as the CP. This sPDE is the stochastic Fisher equation [24], *i.e.*, the Fisher equation with an added white noise term. As such, it is not unexpected that we also found an extension for the Fisher equation for the coarse-grained density. At first glance, one might expect that the white noise would be averaged out, such that we only obtain the Fisher equation. However, the white noise could lead to nontrivial correlations for future time steps, which would introduce a bias in the average dynamics.

At this moment it is not entirely clear how the stochasticity relates to $(u_x)^2$. In addition, we know that the found PDE does not work on every time scale. Lastly, it has has coefficients that are dependent on the initial conditions. As such, the relation may be coincidental, where $(u_x)^2$ is simply an effective term that helps approximate the length-scale broadening of the interface. We have not looked into implementing stochasticity into the current methods, but one alternative way to test the sPDE is to solve and fit it onto the data by tuning the input coefficients.

It is also not yet entirely clear why $(u_x)^2$ is preferred over $|u_x|$, which we observed for the CP above the critical point. We found that far above the critical point, the CP is similar to the Eden model: the amount of death is negligible to the birth such that in the long time regime, there is a travelling wave obeying $u_t = v|u_x|$. It would thus be interesting to see if this transition into $(u_x)^2$ near the critical point happens smoothly or is something discontinuous. It is a priori not given that the travelling-wave behaviour is the same at the critical point as above it, so arguing $|u_x|$ over $(u_x)^2$ is not much different than choosing $|u_x|^3$ or any other term. As such, it would also be interesting to consider if the data is better described by $|u_x|^k$, with k taking different integer numbers or even critical exponents. This is left for future studies.

This brings us to the final point. Ultimately there should be some connection between the critical exponents and the PDE. The reason for this is that the solution contains information about the average number of particles, which shows power-law scaling with a critical exponent [43]. Hence, to verify this PDE, we could try to estimate this critical exponent from the solution, and compare it to the literature. The connection could also be explicit with critical exponents or fractional derivatives, which might allow us to deal with the inherent scale invariance of the system at the critical point. However, it is presently not clear how to extend the basis set to manage this level of coefficient and exponent freedom, without succumbing to overfitting. We found that the data-driven methods work best with a simple ansatz based on prior knowledge of similar systems, and symmetries of the system itself. In general, adding an increasing number of terms to the basis set does not lead to better sparse descriptions.

6 Conclusion

In summary, we have utilised data-driven techniques to find PDEs. For this we made use of the algorithm developed by Brunton *et al.* [14]. The goal of this study is to develop a strategy for applying this algorithm to interacting lattice particle systems. The first step towards reaching this goal was to validate the algorithm on smooth data obtained from solving the Fisher equation, and noisy data from non-interacting particle systems, such as the random walk. These studies provided us with the insight that we need to use various initial conditions, filters and a statistical analysis on multiple coarse-grained data sets to reliably obtain new PDEs. In general, we found that randomly generating at least 25 data sets on 4 or more initial conditions was enough to get relevant statistics. In addition, coarse-graining the data with 10^4 or more simulations containing around 100 or more particles yields relatively smooth data that can easily be filtered.

We have also proposed and used a number of improvements to the algorithm developed by Brunton *et al.* [14]. The most important change is that we have altered the error evaluation in the algorithm, such that it is more independent of the basis set / starting PDE, which allows us to reliably obtain PDEs that are both sparse and accurate. These additions are general and do not rely on the fact that the underlying system is a 1D lattice model for particles. Furthermore, our filtering techniques to remove the noise and estimate the derivatives are easy to use and have interpretable tuning parameters. Hence, the methods discussed in this thesis could serve as useful tools for future utilisation of the algorithm in experimental and numerical studies alike.

Building upon our experience with non-interacting particle systems and the Fisher equation, we considered an interacting particle system that is a combination of two models: the Eden model (which has particle growth on vacant neighbouring sites), and the symmetrical simple exclusion process (SSEP), which models particle displacement with hard-core exclusion. For suitable choices of the input parameters, we obtained the theoretically predicted Fisher equation $u_t = Du_{xx} + ru(1-u)$ within the standard deviation ($< 2\%$). This numerical result is new and shows the potential for the application of data-driven techniques onto (strongly) interacting particle systems.

Next, we turned to the pure Eden model and we obtained travelling waves that could be described via $u_t = v|u_x|$, but not by the Fisher equation. The right-hand side is a symmetric advection term, *i.e.*, the waves move to the right and to the left. We found that the coefficients were within 1% of the expected values. From this we can conclude that we can obtain effective PDEs for interacting particle systems that do not contain diffusion.

Finally, we examined a version of the Eden model that incorporates death, namely the contact process (CP). Based on the ratio between growth and death, the CP can be in different phases, including a critical phase. We found that far above the critical point, the CP is similar to the Eden model: there are travelling waves, where $v_{\text{cp}} \to v_{\text{eden}}$ as the growth rate becomes bigger. At the critical point of the CP, we found a new effective PDE for the average dynamics that is similar to the Fisher equation: $u_t = Du_{xx} + au - bu^2 + c(u_x)^2$. The emergent diffusion and Fisher terms are to be expected from the stochastic Fisher equation of the CP, which has a white noise term. The appearance of the $(u_x)^2$ term is explained by the observed interface broadening of the average density. However, we did find that there is a dependency on the time regime and initial condition. This implies that there are correlations in the dynamics that are hard to capture with a sparse coarse-grained PDE. This is not unexpected given the stochastic nature of the critical behaviour. Hence, for future studies we could try to fit a solution of the stochastic Fisher equation onto the average density.

From our combined results, we can conclude that when there is sufficient data to cross check PDEs and ensemble average the densities, we can reliably obtain effective PDEs in interacting particle systems. As an outlook, this work could be applied on (experimental) data obtained for complex systems related to the ones studied here, for which it is relatively easy to (automatically) generate enough data. This could include systems inhabiting growth and/or death, such as bacterial colonies or infection models, but also systems that combine (biased) movement with exclusion, such as traffic models or active matter systems.

References

[1] S. L. Brunton, J. L. Proctor, and J. N. Kutz, Proceedings of the National Academy of Sciences **113**, 3932 (2016).

[2] G. E. Uhlenbeck and L. S. Ornstein, Phys. Rev. **36**, 823 (1930).

[3] G. H. Weiss, American Scientist **71**, 65 (1983).

[4] J. C. Chen and A. S. Kim, Advances in Colloid and Interface Science **112**, 159 (2004).

[5] E. A. Codling, M. J. Plank, and S. Benhamou, Journal of The Royal Society Interface **5**, 813 (2008).

[6] J. Gärtner, Stochastic Processes and their Applications **27**, 233 (1987).

[7] J. L. Lebowitz, E. Presutti, and H. Spohn, Journal of Statistical Physics **51**, 841 (1988).

[8] G. Haag, *Modelling with the Master Equation* (Springer Cham, Gewerbestrasse 11, 6330 Cham, Switzerland, 2017), 1st ed.

[9] M. F. Weber and E. Frey, Reports on Progress in Physics **80**, 046601 (2017).

[10] A. Zöttl and H. Stark, Journal of Physics: Condensed Matter **28**, 253001 (2016).

[11] S. Funtowicz and J. R. Ravetz, Futures **26**, 568 (1994), special Issue Complexity: Fad or Future?

[12] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, Annual Review of Fluid Mechanics **52**, 477 (2020).

[13] R. Supekar, B. Song, A. Hastewell, G. P. T. Choi, A. Mietke, and J. Dunkel, *Learning hydrodynamic equations for active matter from particle simulations and experiments* (2021).

[14] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, Science Advances **3** (2017).

[15] B. Derrida, J. L. Lebowitz, and E. R. Speer, Journal of Statistical Physics **107**, 599 (2002).

[16] M. Eden, *Proc. Fourth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 4 (Univ. California Press, Berkeley, 1961).

[17] T. Vicsek, M. Cserző, and V. K. Horváth, Physica A: Statistical Mechanics and its Applications **167**, 315 (1990).

[18] G. C. Barker and M. J. Grimson, Journal of Physics A: Mathematical and General **27**, 653 (1994).

[19] R. A. Fisher, Annals of Eugenics **7**, 355 (1937).

[20] D. Griffeath, Stochastic Processes and their Applications **11**, 151 (1981).

[21] J. R. G. de Mendonça, Journal of Physics A: Mathematical and General **32**, L467 (1999).

[22] C. Bezuidenhout and G. Grimmett, The Annals of Probability **18**, 1462 (1990).

[23] C. Müller and R. Tribe, Probability Theory and Related Fields **102**, 519 (1995).

[24] Š. Birnšteinová, M. Hnatič, T. Lučivjanský, L. Mižišin, and V. Škultéty, Theoretical and Mathematical Physics **200**, 1335 (2019).

[25] M. Matsushita, F. Hiramatsu, N. Kobayashi, T. Ozawa, Y. Yamazaki, and T. Matsuyama, Biofilms **1**, 305 (2004).

[26] M. Deforet, C. Carmona-Fontaine, K. S. Korolev, and J. B. Xavier, *Evolution at the edge of expanding populations* (2017).

[27] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, New York, USA, 2014).

[28] J. E. Guyer, D. Wheeler, and J. A. Warren, Computing in Science & Engineering **11**, 6 (2009).

[29] A. E. Hoerl and R. W. Kennard, Technometrics **12**, 55 (1970).

[30] D. W. Marquardt and R. D. Snee, The American Statistician **29**, 3 (1975).

[31] H. J. Nussbaumer, *The Fast Fourier Transform* (Springer Berlin Heidelberg, Berlin, Heidelberg, 1981), pp. 80–111.

[32] I. Knowles, Electronic Journal of Differential Equations **21**, 235 (2014).

[33] A. Savitzky and M. J. E. Golay, Analytical Chemistry **36**, 1627 (1964).

[34] V. M. T. e. V. M. Tikhomirov (auth.), *Selected Works of A. N. Kolmogorov: Volume I: Mathematics and Mechanics*, vol. 1 of *Mathematics and Its Applications (Soviet Series) 25* (Springer Dordrecht, 1991), 1st ed.

[35] D. Panja, Physics Reports **393**, 87 (2004).

[36] S. P. Lalley and T. Sellke, The Annals of Probability **15**, 1052 (1987).

[37] V. M. Kenkre and M. N. Kuperman, Phys. Rev. E **67**, 051921 (2003).

[38] Y. Zeldovich, Combustion and Flame **39**, 219 (1980).

[39] J. Canosa, Journal of Mathematical Physics vol. 10 iss. 10 **10** (1969).

[40] M. R. Roussel, *Random Walk*, Department of Chemistry and Biochemistry, University of Lethbridge (2009).

[41] T. M. Liggett, The Annals of Probability **25**, 1 (1997).

[42] T. M. Liggett, *Interacting Particle Systems* (Springer, Berlin, Heidelberg, 2005), 1st ed.

[43] S. L. Malte Henkel, Haye Hinrichsen, *Non-Equilibrium Phase Transitions: Volume 1: Absorbing Phase Transitions*, Theoretical and Mathematical Physics (Springer, 2009), 1st ed.