Bayesian uncertainty quantification for data-driven equation learning

Simon Martina-Perez\textsuperscript{1}, Matthew J. Simpson\textsuperscript{2} and Ruth E. Baker\textsuperscript{1}
\textsuperscript{1}Mathematical Institute, University of Oxford, Oxford, United Kingdom
\textsuperscript{2}School of Mathematical Sciences, Queensland University of Technology, Brisbane, Australia

February 24, 2021

Abstract

Equation learning aims to infer differential equation models from data. While a number of studies have shown that differential equation models can be successfully identified when the data are sufficiently detailed and corrupted with relatively small amounts of noise \cite{1, 2, 3, 4}, the relationship between observation noise and uncertainty in the learned differential equation models remains unexplored. We demonstrate that for noisy data sets there exists great variation in both the structure of the learned differential equation models as well as the parameter values. We explore how to combine data sets to quantify uncertainty in the learned models, and at the same time draw mechanistic conclusions about the target differential equations. We generate noisy data using a stochastic agent-based model and combine equation learning methods with approximate Bayesian computation (ABC) to show that the correct differential equation model can be successfully learned from data, while a quantification of uncertainty is given by a posterior distribution in parameter space. Code can be found at \url{https://github.com/simonmape/UQ-for-pdefind}.

1 Introduction

Many phenomena in mathematical biology arise as a result of complex interactions between individual agents at the microscale that result in emergent properties at the macroscale. Understanding the mechanistic basis for the observed macroscale behaviour, in order to gain fundamental insights into biological phenomena, is one of the key challenges in biology. Mathematical models are well-placed to help provide such insights, providing a rigorous framework where hypotheses can be generated, tested and refined. While the interactions between individual agents can be naturally described by agent-based models (ABMs) that prescribe precise rules for the interactions between agents, predicting the macroscale behaviour of ABMs can be a challenging
task, since their governing equations are often intractable and stochastic simulations can be computationally expensive, often prohibitively so in the context of parameter sensitivity analysis or parameter inference [2]. This makes partial differential equation (PDE) models an indispensable tool to describe the macroscale properties of the population. Benefits of PDE models is that they are fast to solve numerically, their different terms often carry a physical interpretation, and they can be explored using a range of analytical and numerical approaches. Knowing how such a model is parametrized, then, can provide key insights into the system under consideration, and aid in making quantitative as well as qualitative predictions.

Traditional approaches to mathematical modelling use experimentally-derived mechanistic hypotheses to derive PDE models in which the various terms of a given model are designed to describe the hypothesised mechanisms for that scenario. Calibration of the model to data then involves finding the parameters that optimise the discrepancy between the model output and data. The ensuing, iterative process of testing and refining the model against further experimental data allows the original hypotheses to be refined, and so new insights gained.

Equation learning (EQL) methods aim to infer the dynamical systems model that best describes a given time series by leveraging statistical and machine learning tools to learn the appropriate terms of a differential equation directly from the data. In particular, the PDE-FIND algorithm [1] takes as input quantitative data, together with a large library of candidate terms for a PDE, and aims to learn which terms to include in the PDE model, as well as their coefficients. User-defined parameters can be tuned to enable a balance between the requirement for good model fit with the desire for a simple, interpretable model.

EQL methods have rapidly gained popularity, mainly thanks to increases in computational power, and a number of other techniques to establish models from data now exist. For example, biologically informed neural networks [3], an extension of physically informed neural networks [5], have been developed to learn different terms of a PDE model without the need to specify a library of possible terms. A major advance has come from the use of techniques such artificial neural nets (ANNs) [6] to accurately recover models from artificially generated noisy data from PDEs. The fact that EQL could discover previously undetected mechanisms, choose between competing models or estimate biological quantities of interest that are difficult to measure experimentally makes EQL attractive to scientists working with real-world data. However, practitioners wishing to obtain models that they can use in real-world settings require, in addition, a thorough quantification of uncertainty. This need comes from the fact that the, often significant, noise in real-world data can impact the models predicted by EQL methods, and hence the predictive capability of the models for unseen data or scenarios. For
example, Nardini et al. \cite{2} have recently shown, through the use of several case studies, that it is possible to infer differential equation models that describe noisy data generated by stochastic ABMs. However, the stochasticity in the ABM results in variability in the learned macroscale differential equation. This means that, for a particular realisation generated from a stochastic model, the learned differential equation is essentially a point estimate of the underlying PDE, and there is no quantification of uncertainty in the learned equation.

Recently, some authors \cite{1,7} have attempted to address this problem by analyzing the robustness of PDE-FIND with noisy or sparse data. For example, the approach in \cite{1} amounts to choosing a PDE with ground truth parameters, simulating from this target equation, corrupting the simulated data by noise, training PDE-FIND on this data, and finally comparing PDE-FIND parameters with the ground truth. Li et al. \cite{7} investigate how to increase the signal-to-noise ratio prior to using EQL techniques. While both works show that model parameters can be retrieved in within an impressive margin of error when data is corrupted with relatively small amounts of noise, these approaches give no statistical quantification of uncertainty in model predictions, as the methods are still designed to output a single learned equation.

![Figure 1: Proposed framework for uncertainty quantification in equation learning. A noisy data ensemble is used to create an informative prior distribution efficiently, which can then be used to obtain a practical posterior distribution.](image)

In this work we demonstrate that noise can significantly impact both the structure and the parameters of learned differential equations, rendering uncertainty quantification a crucial
component of the equation learning process. As such, our overarching aim is to develop and showcase a method for uncertainty quantification in the context of EQL, where we harness the immense computational efficiencies of PDE-FIND in learning point estimates of governing equations, together with the power of computational Bayesian inference in evaluating the level uncertainty in the learned equation. Figure 1 shows our proposed framework for uncertainty quantification. We start from the basis that it is possible to collect an ensemble of spatiotemporal datasets from a given system, and develop an approach to understand how the data can be best used to learn a governing equation while simultaneously estimating the uncertainty in that learned equation.

Our motivation is thus: on the one hand, PDE-FIND provides a computationally cheap method to obtain a point estimate for the governing equation from a single time series. However, when the data are noisy, the individual predictions are unreliable. On the other hand, the field of computational Bayesian inference provides a number of methods to estimate, for a given model and data, posterior parameter distributions i.e. provide estimates of model parameters and quantify the uncertainty in those estimates. In principle, a likelihood-free method such as approximate Bayesian computation (ABC) could be used directly with the candidate library of the PDE-FIND method to estimate the posterior distribution of the library coefficients. However, due to the very large number of candidate terms in the PDE-FIND library, the computational cost associated with applying ABC (or other likelihood-free methods for Bayesian inference) on the entire high-dimensional parameter space is prohibitive. Instead, we propose a framework that combines the strengths of each approach: we train the PDE-FIND algorithm on individual datasets from the ensemble to obtain an informed prior distribution that makes it possible to use ABC in a computationally realistic setting. We demonstrate the potential of our approach using synthetic data generated from a widely used ABM that describes the behaviour of a motile and proliferative cell population and can be accurately coarse-grained to a mean-field PDE in certain regions of parameter space.

In Section 2 we describe the ABM and discuss in detail its relation with a governing PDE. We describe the PDE-FIND algorithm and propose how the PDE-FIND algorithm may be used to construct a prior for Bayesian inference. In Section 3 we show that the learned PDEs are intrinsically variable in the presence of observation noise, and regardless of the method to deal with observation noise, there is still variability that needs to be quantified. We then demonstrate that PDE-FIND can learn unphysical solutions to the PDE and provide an explanation in terms of the objective function of the algorithm. We bring all these issues together through the use of Bayesian methods where we can evaluate uncertainty in a framework that optimizes the fit of
the model density profile to the data.

2 Model and methods

We begin by describing the ABM and its coarse-grained mean-field PDE, and then we outline the PDE-FIND algorithm and the means by which we use it to construct a prior distribution for ABC.

2.1 An agent-based model

ABMs allow practitioners to investigate the collective behaviour of cells and tissues based on a description of the behaviour of individual cells. Here, in order to realistically describe the cell-cell interactions, we follow a volume-exclusion model presented in [9] for a population of agents that move and proliferate according to a discrete random walk model. This is a simple model that can be used to analyse a range of phenomena, including the collective migration of cells in a tissue, for example. In this model, we will assume that agents occupy sites on a square lattice and their possible locations are \((i, j)\), where \((i, j)\) are integer coordinates. Agent movements are constrained by exclusion: each lattice site may only be populated by one agent at a time. Agents can only move to either one of their four adjacent lattice sites. The lattice is initialized according to some initial conditions, and the parameters \(p_m, p_p \in [0, 1], \rho_x, \rho_y \in [-1, 1]\) are set.

At each time step \(\tau\), a random sequential updating procedure is carried out. At each time step, agents are selected, one at a time, with replacement, and are allowed to attempt a movement and proliferation event. When an agent is selected, a movement event is attempted with probability \(p_m\) and the random numbers \(S_1, S_2 \sim U(0, 1)\) are independently drawn. If \(S_1 \leq p_p\), then the agent proliferates and a daughter agent is placed in one of the adjacent sites, if it is empty. If a movement is attempted, then, \(S_2\) determines the motion of the agent according to Table 1. We recall that proliferation and movement into any site can only occur if that site is empty.

| Move chosen       | Target site | Probability     | Where random number \(S_2\) falls |
|-------------------|-------------|-----------------|----------------------------------|
| vertically down   | \((i, j - 1)\) | \(\frac{1 - \rho_y}{4}\) | \(0 \leq S_2 \leq \frac{1 - \rho_y}{4}\) |
| vertically down   | \((i, j + 1)\) | \(\frac{1 + \rho_y}{4}\) | \(\frac{1 - \rho_y}{4} \leq S_2 \leq \frac{1}{2}\) |
| horizontally left | \((i - 1, j)\) | \(\frac{1 - \rho_x}{4}\) | \(\frac{1}{2} \leq S_2 \leq \frac{1}{2} + \frac{1 - \rho_x}{4}\) |
| horizontally right| \((i + 1, j)\) | \(\frac{1 + \rho_x}{4}\) | \(\frac{1}{2} + \frac{1 - \rho_x}{4} \leq S_2 \leq 1\) |

Table 1: Algorithm by which an agent at site \((i, j)\) selects a target site, as presented in [9].

From a biological perspective, the parameters control the collective behavior of the cells.
A high value of $p_m$ reflects cells that move easily through their environment, whereas a lower value of $p_m$ corresponds to lower motility in the process. This will be reflected by fewer cell movements on average in each time interval. The movement bias parameters $\rho_x, \rho_y$ reflect the preferences of agents in a certain direction. This can be due to factors such as chemotaxis or other factors driving cell movement. Setting $\rho_x = \rho_y = 0$ corresponds to unbiased motion on the lattice. Finally, $p_p$ determines the rate at which cells proliferate: a high value of $p_p$ corresponds to cells that rapidly proliferate, whereas a lower value signifies lower mean proliferation rates.

In this paper, we consider a uniformly spaced lattice, with lattice spacing equal to 1, and time increments $\tau = 1$. We also consider $p_m = 1$. As noted in [9], all simulations can be easily generalized to arbitrary lattice spacings and arbitrary time increments, but we keep things simple here by choosing uniform lattice spacing and constant time steps.

As done in [9], we simulate the ABM on a square lattice with integer coordinates $(i,j)$ such that $1 \leq i \leq 200$ and $1 \leq j \leq 20$. That is, the horizontal dimension of the lattice is 200 and the vertical dimension is 20. This corresponds to looking at a thin strip of cells that move and proliferate. This, for example, corresponds to experiments involving a spatial invasion process. We take a pseudo-one-dimensional initial condition by populating all lattice sites with $90 \leq i \leq 110$, and leaving the rest of the lattice empty. Let $C_{ij}$ denote the occupancy of cell $(i,j)$ by $C_{ij} = 1$ if $(i,j)$ is occupied by an agent and 0 if it is empty. We can average the occupancy of the columns of the lattice, by defining the mean occupancy of column $i$ as

$$C_i = \frac{1}{20} \sum_{j=1}^{20} C_{ij} \quad (1)$$

When doing this averaging at all later time points, this yields a simulation of a time-varying one-dimensional spatial density profile.

### 2.1.1 Coarse-grained PDE model

The dimensions of the lattice considered and the choice of a pseudo-one-dimensional initial condition suggest to consider how the density profile, obtained by averaging the occupancy of each column of the lattice, varies in time. To do this, let $C_i$ be the averaged occupancy in the $i$-th column of the lattice, that is, the average number of agents per site in the column. Then $\{C_i\}$ is a one-dimensional density profile of the lattice. As noted by Simpson et al. in [9], the rules of the stochastic model induce a one-dimensional conservation of mass statement between $C_i(t)$ and $C_i(t + \tau)$, the occupancy of site $i$ at time $t$ and at the next time step, $t + \tau$. Taking
again $p_m = 1$,

$$C_i(t + \tau) = C_i(t) + C_{i-1}(t)\left(\frac{1 + \rho_x}{4}\right)(1 - C_i(t)) + C_{i+1}(t)\left(\frac{1 - \rho_x}{4}\right)(1 - C_i(t))$$

$$- C_i(t)\left(\frac{1 + \rho_x}{4}\right)(1 - C_{i+1}(t)) - C_i(t)\left(\frac{1 - \rho_x}{4}\right)(1 - C_{i+1}(t))$$

$$+ \frac{p_p}{2} C_{i-1}(t)(1 - C_i(t)) + \frac{p_p}{2} C_{i+1}(t)(1 - C_i(t)).$$

Note how the different terms in the conservation of mass statement correspond to the flux of agents in and out of the adjacent lattice sites. Note also how $\rho_y$ does not appear in the conservation of mass statement, justifying the choice of $\rho_y = 0$ henceforth. For the general derivation of the conservation of mass statements, we refer to [9]. The general one-dimensional conservation of mass statement for the average occupancy between $t$ and $t + \tau$ corresponds to an explicit finite difference approximation of the PDE

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} - V \frac{\partial}{\partial x}[c(1 - c)] + P[c(1 - c)],$$

in the limiting case where $\tau$ and the mesh size $\Delta$ satisfy $\tau, \Delta \to 0$ simultaneously while keeping $\Delta^2/\tau$ constant. In Equation 2

$$D = \lim_{\Delta, \tau \to 0} p_m \Delta^2/4\tau, \quad V = \lim_{\Delta, \tau \to 0} \Delta \rho_x/2\tau \quad \text{and} \quad P = \lim_{\tau \to 0} p_p/\tau.$$  

For the full derivation and details, we refer to [9]. The identification of this agent based model with the resulting PDE motivates us to investigate the performance of equation learning methods trained on data generated by the agent based model, since the PDE describes the time-evolution of the expected value of the density profile.

### 2.1.2 Comparison of the ABM and PDE model predictions

As test cases for learning the governing equations from data, we explore three different parameter regimes in the model, which each correspond to a biologically relevant setting. In Case I, we consider agents moving without bias and without proliferation ($\rho_x = \rho_y = p_p = 0$). In Case II, we consider agents moving with bias, but without proliferation ($\rho_x = 0.075$ and $\rho_y = p_p = 0$). In Case III, we consider agents moving without bias, but with proliferation ($\rho_x = \rho_y = 0$ and $p_p = 0.001$). Table 2 outlines the different cases.

To be able to control for different levels of noise in the data set, we create two data sets for each case. In the first data set, for each case, we simulate 1000 independent replicates of the ABM for a total time of 1000 time steps and record the density profile at each time step, across the different space points. In the second data set, we simulate 250 independent replicates of the ABM for a total time of 1000 time steps and record the density profile at each time step, across the different space points. We write $u$ for the unbiased case without proliferation, $b$ the biased case without proliferation, and $p$ the unbiased case with proliferation. We let $i = 1$ for
Case | $\rho_x$ | $p_p$ | Coarse-grained PDE
--- | --- | --- | ---
Case I: no bias, no proliferation | 0 | 0 | $c_t = 0.25u_{xx}$
Case II: bias, no proliferation | -0.0375 | 0 | $c_t = 0.25c_{xx} + 0.0375[c(1 - c)]_x$
Case III: proliferation, no bias | 0 | 0.01 | $c_t = 0.25c_{xx} + 0.01[c(1 - c)]$

Table 2: Coefficients of the mean-field PDEs describing evolution of the mean population density over time for the three example cases used in this work.

the case where we average over one iterate in each density profile and $i = 2$ for the case where we average over 50 iterates for each density profile, and denote the resulting data sets as $\{D^r_i\}$, where $r \in \{u, b, p\}$ and $i \in \{1, 2\}$. For example, $D^u_1$ refers to the data set corresponding to simulations of Case I where we average over one replicate per density profile.

In Figure 2 it can that the obtained data traces of the ABM are in good agreement with the coarse-grained PDE. Note, however, that for the plots at $t = 50$, the solutions for Case I, II and III are very similar. Still, at $t = 100$ the plots for Case I and III are very similar and it is not until later time scales ($t = 150$) that there can be seen a clear difference between the solution owing to the role of proliferation. We can forecast based on this visual inspection that the question of reliably estimating the PDEs is difficult from the average data, and all time scales need to be taken into account to properly distinguish the data traces.

2.2 Equation learning: PDE-FIND

In the following, assume that we have time series data for an unknown function $u(x, t)$ on a grid of $n$ points in time and $m$ points in space. This data is stored in a matrix $U \in \mathbb{R}^{n \times m}$. We assume that the data is a noisy discretization of a function $u(x, t)$, the solution of an unknown PDE, and the aim is to learn the PDE that best describes the governing equation of the observed data. Henceforth, and to avoid confusion, we will write $U(x, t)$ for the observed data, $u(x, t)$ for the learned, data-driven, solution of the governing PDE, and $c(x, t)$ for the solution to the approximate mean-field PDE defined in Equation (2). We follow Rudy et al. [1] in assuming that the PDE governing $u(x, t)$ is given in the following form

$$u_t = \mathcal{N}(u, u_x, u_{xx}, \ldots, x, \bar{p}) = \sum_{i=1}^{d} \mathcal{N}_i(u, u_x, u_{xx}, \ldots, \bar{\xi})\xi,$$

where $\mathcal{N}$ is a nonlinear function of $u(x, t)$ and its partial derivatives, $\bar{p}$ and $\bar{\xi}$ are constants. Furthermore, it is assumed that $\mathcal{N}$ is the linear combination of a finite number of distinct library terms. We point out that Equation (2) falls within this class of PDEs. By design, $\mathcal{N}$
Figure 2: Typical data traces of the 1D density profiles obtained with the ABM. For each of the three cases we show the density at times $t = 0$, $t = 50$, $t = 150$, $t = 500$ for one of the simulated realizations of the model where no averaging is done (left column), where time series consist of averages over 50 realizations of the model (middle column) and the corresponding continuum model. It can be seen that the continuum model is a very good approximation of the average density obtained from the ABM simulations.
has polynomial nonlinearities, as is common in many equations in the natural sciences. PDE-FIND creates a large library of terms that could appear in \( \mathcal{N} \), with the view to select a small subset of relevant terms from this list.

The first step of the PDE-FIND pipeline is to numerically approximate both sides of Equation (3). This is done by taking derivatives of the data with respect to space and time. The standard PDE-FIND implementation in [1] takes finite difference approximations when the data contains little noise and polynomial differentiation when data is highly noisy. The data and its derivatives are combined in a matrix \( \Theta(U) \), where each column of \( \Theta \) contains all of the values of a particular candidate function across the entire \( n \times m \) grid. For example, if the candidate library consists of all polynomials up to degree 2 and all derivatives up to the second order, \( \Theta(U) \) will look like

\[
\Theta(U) = [1, U, U^2, U_x, UU_x, U^2U_x, Uxx, UUxx, U^2Uxx].
\]

So, if there are \( N \) terms in the candidate library, \( \Theta(U) \) is a \( n \cdot m \times N \) matrix. In this example, \( N = 9 \). The left hand side of Equation (3) is similarly approximated using time differentiation, and we obtain a linear equation representing the PDE in Equation (3)

\[
U_t = \Theta(U)\xi.
\]

Taking the same example for \( \Theta(U) \) as in Equation 4, this matrix equation looks like

\[
\begin{pmatrix}
U_1(x_0, t_0) \\
U_1(x_1, t_0) \\
U_1(x_2, t_0) \\
\vdots \\
U_1(x_{n-1}, t_m) \\
U_1(x_n, t_m)
\end{pmatrix} = \begin{pmatrix}
1 & U(x_0, t_0) & U^2(x_0, t_0) & U_x(x_0, t_0) & \cdots & U^2Uxx(x_0, t_0) \\
1 & U(x_1, t_0) & U^2(x_1, t_0) & U_x(x_1, t_0) & \cdots & U^2Uxx(x_1, t_0) \\
1 & U(x_2, t_0) & U^2(x_2, t_0) & U_x(x_2, t_0) & \cdots & U^2Uxx(x_2, t_0) \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & U(x_{n-1}, t_m) & U^2(x_{n-1}, t_m) & U_x(x_{n-1}, t_m) & \cdots & U^2Uxx(x_{n-1}, t_m) \\
1 & U(x_n, t_m) & U^2(x_n, t_m) & U_x(x_n, t_m) & \cdots & U^2Uxx(x_n, t_m)
\end{pmatrix} \begin{pmatrix}
\xi_1 \\
\xi_2 \\
\xi_3 \\
\vdots \\
\xi_8 \\
\xi_9
\end{pmatrix}
\]

(6)

Note how this representation shows that each row in the matrix equation represents the governing dynamics behind the data at one point in time and space. Note also that the values of \( \xi \) determine the form of the PDE, and so the aim is to learn the coefficients of \( \xi \) optimally. Following Rudy et al. [1], we will take \( \Theta \) to be overspecified, meaning that the dynamics can be represented well as linear combinations of the columns of \( \Theta \). However, many PDEs in the natural sciences contain only a few terms. Therefore, we wish to learn a sparse vector \( \xi \) in the solution of Equation (3). This is done in PDE-FIND by considering the optimisation criterion

\[
\xi = \arg\min_\xi \|\Theta(U, Q)\xi - U_t\|^2 + \lambda\|\xi\|^2,
\]

(7)

for the coefficients \( \xi \), where \( \lambda \in \mathbb{R}_{>0} \) is a free parameter that penalises large terms. This is the method of ridge regression. We note here that the term \( \|\xi\|^2 \) can be replaced with
∥ξ∥_2^2$, which corresponds to performing LASSO. Here, it is worth noting that the choice of implementation is largely problem-dependent, and various choices for the regularization method have been compared in the literature, for example by Rudy et al. [1], although no method is to be definitively preferred over another. The default implementation of PDE-FIND as proposed by Rudy et al. [1] supplements the ridge regression problem with a sequential thresholding procedure: a solution to (7) and a hard threshold is performed on the regression coefficients by eliminating all coefficients smaller than some pre-specified parameter $d_{tol}$. This process is repeated on the remaining coefficients. This is done to enforce sparsity as the solution to the ridge regression problem in Equation (7) may contain several small, but non-zero values. The combined algorithm is called Sequential Thresholding Ridge regression (STRidge). For details and motivation of the method, we refer to [1]. We summarize the PDE-FIND algorithm in Algorithm 1 as reproduced from [1].

**Algorithm 1: STRidge**

**Input:** Library matrix $\Theta(U)$, time derivative of data $U_t$, $\lambda$, $d_{tol}$

**Output:** Sparse vector $\xi$

1. Compute $\hat{\xi} = \arg\min_\xi \|\Theta(U,Q)\xi - U_t\|_2^2 + \lambda\|\xi\|_2^2$;
2. Set $B = \{ j : |\hat{\xi}| \geq d_{tol} \}$;
3. Set $\hat{\xi}_i = 0$ for all $i \notin B$;
4. Compute $\hat{\xi}[B] = \text{STRidge}(\Theta[:,B],U_t,d_{tol})$ recursively;

### 2.2.1 Application of PDE-FIND to the ABM data

For the simulated ABM data in Case I, II and III, we collect the column-averaged density at each time step. We denote the resulting data sets $D_i^r$ for $r = u,b,p$, $i = 1,2$, where $u$ stands for unbiased motion (Case I), $b$ stands for biased motion (Case II) and $p$ stands for proliferation (Case III). Then, $i = 1$ corresponds to data traces with no averaging and $i = 2$ corresponds to data traces where there is averaging over 50 simulations of the ABM. Per data set, we then subsample $n = 500$ equidistant time points, subsampled from the 1000 time points at which the simulation was carried out, with $\Delta t = 2$. So, each sample in the data set contains information for $n = 500$ time points and $m = 200$ space points. We use the standard implementation of PDE-FIND [1] and use polynomial differentiation at fourth order to find both the time and space derivatives.

We select a library of candidate terms that includes all polynomial terms up to order two and up to the second derivative since interactions between agents happen only between direct neighbors and so we do not expect other terms to show up in the PDE. Table 3 shows the values of the coarse-grained PDE coefficients, according to Equation 2. These are the values of the
coefficients that we would expect the PDE-FIND algorithm to return for perfect spatio-temporal data. In Table 3 and the rest of this work, we use the notation $c_i$ for the coefficient of term $i$ in the learned PDE.

| Case  | $c_1$ | $c_u$ | $c_u^2$ | $c_{ux}$ | $c_{u^2ux}$ | $c_{uxx}$ | $c_{u^2uxx}$ |
|-------|-------|-------|---------|---------|-------------|---------|-------------|
| Case I| 0     | 0     | 0       | 0       | 0           | 0.25    | 0           |
| Case II| 0    | 0     | 0       | -0.0375 | 0.075       | 0       | 0           |
| Case III| 0   | 0.01  | -0.01   | 0       | 0           | 0.25    | 0           |

Table 3: Coefficients of the mean-field PDEs describing evolution of the mean population density over time for the three example cases used in this work. The coefficients correspond to the coarse-grained PDEs described in table 2.

2.3 Using PDE-FIND to define a prior distribution for ABC

ABC is a popular likelihood-free tool to obtain a data-informed posterior distribution over the space of possible parameter values. The main idea of ABC is that the user specifies a prior distribution $\pi$ reflecting prior knowledge of model parameters, which the ABC algorithm then uses to create a posterior distribution by evaluating the performance of the parameters in reproducing the observed data. For an overview of ABC methods, we refer to Sunnåker et al. [8] - see the SI for implementation details.

Such posterior distributions provide information as to the uncertainty in parameter estimates that are learned from the data, and also allow practitioners to understand the range of realistic parameter values that can produce the observed data. Despite its simplicity, unless an informative prior is used to constrain the space of possible parameters, a high-dimensional parameter search using ABC is generally computationally prohibitive for the kinds of models routinely used in mathematical biology. This high computational cost means that direct application of ABC methods to estimate the coefficients $\xi$ of Equation (3) is infeasible. Here, we propose a method that uses the predictions of PDE-FIND to construct an informed prior so that ABC can then be used to estimate the library coefficients $\xi$.

Consider performing sparse linear regression using Algorithm 1 to the PDE-FIND solution $\xi$ to the problem defined in Equation (4). Let $\hat{\beta}(\lambda)$ denote the ridge regression solution - that is the solution obtained without applying the hard thresholding part of Algorithm 1, i.e. by $d_{tol} = 0$. Let $\hat{\beta}(\lambda, ST)$ denote the STRidge coefficients obtained when using a fixed value of the threshold parameter $d_{tol} > 0$. It is well-known - see [10] - that in the presence of observation noise, the empirical distribution of $\hat{\beta}(\lambda)$ can be well-approximated with a multivariate normal
distribution. Now, the exact distribution of \( \hat{\beta}(\lambda, \text{ST}) \) in the presence of observation noise will be different due to the hard thresholding imposed by the STRidge algorithm, but we expect that the nonzero coefficients of \( \hat{\beta}(\lambda, \text{ST}) \) follow a similar distribution to those of \( \hat{\beta}(\lambda) \). This justifies approximating the empirical distribution of the coefficients of \( \hat{\beta}(\lambda, \text{ST}) \) as a multivariate normal. Here, it is important to note that the purpose of the method is not to find the exact distribution the PDE-FIND coefficients. Rather, it is to find a model that can capture the prior uncertainty about the PDE-FIND coefficients so that this can be used in a Bayesian context.

Our approximation that the nonzero STRidge coefficients approximately follow a normal distribution suggests that we must find a mixture distribution that allows for coefficients to be exactly zero or to follow a normal distribution. With other words, we wish to approximate the marginal distribution of each coefficient of the PDE-FIND solution with a mixture of a Dirac-\( \delta \) at 0 and a normal distribution. In the following, we propose a straightforward method to use the observed data to construct such a distribution. Assume that for each sample in the observed data set, PDE-FIND has produced an estimate of the parameters using STRidge. For each term \( j \) in the library, define the identification ratio \( a_j \) as

\[
a_j^\lambda = \frac{1}{N_s} \sum_{i=1}^{N_s} 1(\hat{\beta}(\lambda, \text{ST})^i_j \neq 0).
\] (8)

When the data set is sufficiently large, this parameter is informative of how often the term is included in the PDE-FIND predictions. When \( a_j \) is large, the term is identified across the samples in the data set as being relevant for the dynamics. When \( a_j \) is small, the term is identified in only some samples as being relevant. Dropping the dependence on \( \lambda \), since this will always be evident from context, we propose to model the empirical distribution of the \( j \)-th term in the library of the PDE-FIND coefficients obtained from the data set as

\[
\hat{\beta}(\lambda, \text{ST})_j \sim a_j \mu^j + (1 - a_j) \delta_0,
\] (9)

where \( \mu^j = \mathcal{N}(m_j, s_j) \) with \( m_j, s_j \) the sample mean and variance over the nonzero values of the \( j \)-th coefficient and \( \delta_0 \) the Dirac \( \delta \) function at 0. We note that using the identification ratio yields an efficient method of constructing a prior. This is because PDE-FIND is a sparsity-promoting method and so for most coefficients the identification ratio \( a_j \) will be either close to 0, or close to 1. If \( a_j \approx 0 \), the empirical marginal of the \( j \)-th coefficient is approximately \( \delta_0 \), which means that PDE-FIND does not identify this term as relevant across many samples in the data set, implying that the term can be confidently eliminated from the model and not considered in the prior. If \( a_j \approx 1 \), PDE-FIND has consistently identified the term as relevant for the dynamics across the samples in the data set, and so this term should be included in the model, but uncertainty must still be quantified. If \( a_j \) is neither close to 0 nor close to 1, PDE-FIND includes these terms in
the PDE in for a non-trivial number of data points, but fails to confidently include them. In this scenario, Bayesian models can be used to investigate the joint posterior distribution of these parameters with the rest of the model by considering the performance of models that include and exclude the term respectively.

To perform this term elimination procedure more rigorously, let $0 < \delta \ll 1$ and define $A = \{ j : a_j > \delta \}$, that is, the set of terms which a practitioner feels have a significant identification ratio. The key will be to perform Bayesian inference solely for the terms in $A$, while excluding the rest of the terms from the model on the basis of their low identification ratio. We recall that PDE-FIND promotes sparsity, so the size of $A$ will be very small compared to that of the library, making ABC computationally much more efficient. With the information available from training the PDE-FIND coefficients on the ensemble of data points, and in the absence of further information about the problem, the most straightforward prior that can be defined is an independent prior over the coefficients in $A$, giving the following choice

$$\pi = \bigotimes_{j \in A} (a_j \mu_j + (1 - a_j) \delta_0). \quad (10)$$

We summarize the Bayes-PDE-FIND algorithm below in Algorithm 2.

| Algorithm 2: Bayes-PDE-FIND |
|-----------------------------|
| **Input:** Data set of time series; PDE-FIND hyperparameters: library $\Theta$ of size $N$, $\lambda$, $d_{tol}$; minimum identification ratio $\delta > 0$. |
| **Output:** Posterior distribution over coefficients of library PDE. |

1. **for** Sample $i$ in data set **do**
2. Compute $\hat{\xi}^i$ with sample $i$ using Algorithm 1.
3. **end**
4. **for** $j \in 1, \ldots, N$ **do**
5. Compute sample mean $m_j$ and sample variance $s_j$ and set $\mu^j = N(m_j, s_j)$;
6. Compute identification ratio $a_j$ by $a_j = \frac{1}{N_s} \sum_{i=1}^{N_s} 1(\hat{\xi}_{ij}^i \neq 0)$
7. **end**
8. Compute $A = \{ j : a_j > \delta \}$;
9. Define prior distribution
10. $\pi = \bigotimes_{j \in A} (a_j \mu_j^j + (1 - a_j) \delta_0)$. \quad (11)
11. Perform Approximate Bayesian Computation to obtain posterior distribution.
3 Results and discussion

In this section, we showcase three different, but related, directions in the uncertainty quantification of the learned differential equations. In subsection 3.1, we showcase the intrinsic variability of the learned coefficients in the presence of observation noise. We evaluate how uncertainty changes as the noise is varied and suggest that even when using state-of-the-art denoising approaches, a need for uncertainty quantification remains. Although increasing the signal-to-noise ratio helps - regardless of the method - there is still variability and that needs to be quantified. In particular, this is important in biology where observations are often very noisy practitioners rarely have access to very large amounts of data. In subsection 3.2, we investigate the impact of varying algorithm hyperparameters in PDE-FIND with a view to asking whether these can be optimized to drive down uncertainty. We find that while this is possible, parameter estimates are still uncertain and this uncertainty needs to be quantified. In subsection 3.3, we demonstrate that a key issue with PDE-FIND is that it aims to fit the time derivative and does not evaluate the fit of the observed density to the data, leading to unphysical predictions. We bring all these issues together in subsection 3.4 through the use of Bayesian methods as outlined in Algorithm 2 where we can both evaluate uncertainty in a framework that optimizes the fit of the model density profile to the data.

3.1 Variability of relevant PDE-FIND coefficients with noisy observations

We first demonstrate that a naive application of PDE-FIND on noisy synthetic data yields variable and unreliable parameter estimates. For this application, we do not carry out hyperparameter tuning, but simply use widely used parameter settings to learn the coefficients. For each of the two data sets associated to each of Case I, Case II and Case III, we train the PDE-FIND algorithm using Algorithm 1 (STRidge) with fixed hyperparameter settings \( \lambda = 10^{-2} \) and \( d_{\text{tol}} = 0.001 \). These settings were also used in the context of estimating the diffusion parameter in a random walk model in the supplementary information of [1]. For each of the resulting data sets, we also compute the corresponding identification ratios, shown in Table 5 to quantify the degree of identification of the different terms in the model, and compare the performance of PDE-FIND on the different case studies.

3.1.1 Case I

For Case I, recall that the true PDE model contains only the term \( u_{xx} \). Table 5 shows that the two terms identified regularly by PDE-FIND on \( \mathcal{D}_u^1 \) are \( u_{xx} \) and \( uu_{xx} \), with identification ratios of 0.83 and 0.20, respectively. On \( \mathcal{D}_u^2 \), only \( u_{xx} \) is identified and no other terms are identified.
Table 4: Identification ratios for terms considered by PDE-FIND in all experiments.

| Experiment       | \(c_1\) | \(c_u\) | \(c_{u^2}\) | \(c_{ux}\) | \(c_{u-u_x}\) | \(c_{u-ux}\) | \(c_{u^2-u_{xx}}\) | \(c_{u-u_{xx}}\) | \(c_{u^2-u_{xx}}\) |
|------------------|---------|---------|-------------|-------------|---------------|---------------|----------------|---------------|---------------|
| STRidge \(D^i_u\) | 0.001   | 0.0     | 0.002       | 0.0         | 0.008         | 0.008         | 0.826          | 0.199         | 0.05          |
| STRidge \(D^2_u\) | 0.0     | 0.0     | 0.0         | 0.0         | 0.0           | 0.0           | 1.0            | 0.0           | 0.0           |
| STRidge \(D^1_b\) | 0.0     | 0.0     | 0.0         | 0.999       | 0.0           | 0.0           | 0.012          | 0.002         | 0.0           |
| STRidge \(D^2_b\) | 0.0     | 0.0     | 0.0         | 1.0         | 0.0           | 0.0           | 0.0            | 0.0           | 0.0           |
| Subsampling \(D^1_u\) | 0.0 | 0.0     | 0.003       | 0.001       | 0.002         | 0.007         | 0.998          | 0.365         | 0.063         |

Figure 3: Histograms showing the empirical distribution of PDE-FIND coefficients. A: histograms for \(c_{uxxx}\) trained on \(D^1_u\) (blue) and trained on \(D^2_u\) (red) compared to true parameter value (black line). B: histogram for \(c_{uu_{xx}}\) trained on \(D^1_u\) compared to true value (black line). C: joint distribution of \(c_{uxxx}\) and \(c_{uu_{xx}}\) compared to true parameter (black star).

To investigate the variability of the learned parameters, we display histograms of the learned coefficients in Figure 3. In addition, since the identification ratios of \(u_{xx}\) and \(uu_{xx}\) on \(D^1_u\) suggest that the terms have a nontrivial joint distribution, we display the joint distribution of their coefficients as well.

Figure 3 reveals that the learned parameters are highly variable and that with noisy data, the terms \(uu_{xx}\) and \(u_{xx}\) have a nontrivial joint distribution. This means that in some cases, PDE-FIND will identify either of the two terms, and in others it will identify a combination of the two. Since it is assumed that all of the empirical data is governed by the same governing PDE, this points towards non-identification of the model by PDE-FIND.

3.1.2 Case II

For Case II, Table 5 shows that the two terms identified regularly by PDE-FIND on \(D^1_b\) are \(u_x\) and \(u_{xx}\) with an identification ratio of 0.999 and 0.012 respectively. Note that the true model should contain the term \(uu_x\) as well, but PDE-FIND fails to identify this term across the data.
Figure 4: Histograms showing the empirical distribution of relevant PDE-FIND parameters for Case II. A: histogram for the $c_{u_{xx}}$ trained on $D^1_b$ (blue) compared to true parameter value (black line). B: histogram of $c_{u_x}$ trained on $D^1_b$ (blue) compared to true parameter value (black line). C: histogram of $c_{u_x}$ trained on $D^2_b$ compared to true parameter value (black line).

Similarly, for $D^2_b$, only $u_x$ is identified, with an identification ratio of 1.0. This suggests to investigate the variability of the learned coefficients for $u_x$ and $u_{xx}$ learned on $D^1_b$ and only the coefficient for $u_x$ learned on $D^2_b$.

Figure 4 again reveals a large amount of variability in the learned parameters. For instance, note that when trained on $D^1_b$ and on $D^2_b$, parameters are distributed on a wide interval away from the true parameter value. Note also that the variability is lessened when there is less observation noise.

3.1.3 Case III

For Case III, Table 5 shows that the terms identified regularly by PDE-FIND on $D^1_p$ are $u$, $u^2$ and $u_{xx}$ with identification ratios equal to 0.659, 0.482 and 0.571 respectively. Note that this means that all correct terms have been identified. On $D^2_p$ the parameters identified are $u$ and $u_{xx}$, both with identification ratio equal to 1.0. In addition, since the identification ratios of $u$ and $u^2$ on $D^1_p$ suggest that the terms have a nontrivial joint distribution, we display the joint distribution of their coefficients as well.

Figure 5 reveals that the learned parameters are much less variable when less noise is present in observation data. However, it does not mean that the model is identified better per se when there is little observation noise. For instance, the term $u^2$ is not learned by the model on the less noisy data.

3.1.4 Analysis with spatially subsampled data

We investigate whether choosing a coarser grid helps in reducing observation noise with the aim to improve PDE-FIND predictions. Note that choosing a coarser spatial discretization will
Figure 5: Histograms for relevant PDE-FIND parameters trained for Case III. A: histogram for $c_u$ trained on $D^1_p$ (blue) and $D^2_p$ (red) compared to true parameter value (black line). B: histogram for $c_{u^2}$ trained on $D^1_p$ (blue) compared to true parameter value (black line). C: histogram for $c_{u\times u}$ trained on $D^1_p$ (blue) and $D^2_p$ (red) compared to true parameter value (black line).

result in a smoother density profile, while incurring more errors in the approximation of the spatial derivatives, as well as having fewer data points. For this experiment, we subsample the $x$-dimension by averaging the occupancy over 4 columns at a time time. Mathematically, from the empirical densities $C_i$, at each time point, we estimate the average occupancies $\tilde{C}_i$ for $1 \leq i \leq 50$, where

$$\tilde{C}_i = \frac{1}{4} \sum_{\ell=4i+1}^{4(i+1)} C_i.$$  \quad (12)

Table 5 summarizes the identification ratios found. With spatial subsampling, the share of correctly identified diffusion parameters increases significantly, although there is no marked improvement in the identification of other terms. Even if the correct coefficient is identified more often, there remains a need to deal with the rest of terms that are spuriously identified.

3.2 Role of algorithm hyperparameters

The PDE-FIND algorithm affords practitioners a great amount of flexibility in specifying the tuning parameters to carry out the sparse regression. Recall that in Algorithm 1, a free parameter $\lambda$ controls penalty incurred by large coefficients in the solution of Equation (3). It is well known that the choice of regularization parameter is nontrivial as it modulates the amount of sparsity that the sparse regression problem will enforce on the estimated coefficients. The issue of how to choose the optimal value of this hyperparameter was addressed recently by Nardini et al. [2], where cross-validation is discussed among other options. As a test case to investigate the effect of algorithm hyperparameters on the uncertainty of learned coefficients, we perform cross-validation on the data set $D^1_u$ and then train PDE-FIND using the optimal value.
of \( \lambda \) found. To do this, we apply the grid search implementation of cross-validation suggested in [2], as detailed in the SI to arrive at an optimal value of \( \lambda = 0.5 \). We note here that this value is problem-dependent, and whenever a new data set is being investigated, typically, a different value of \( \lambda \) will be appropriate.

While the findings in the supplementary information show that cross-validation improves the performance of the method dramatically, as the number of misspecified coefficients decreases sharply when the regularization parameter is optimized, cross-validation does not provide a sufficient solution to manage the uncertainty associated with the noise in the predicted coefficients. Figure 6 shows how even with the optimal value of the regularization coefficient, there is still much uncertainty in the parameter, as the support of the histogram is large. While the atom at 0 has nearly vanished, uncertainty quantification is still necessary, the empirical distribution still suggests a large degree of stochasticity. Moreover, Figure 6 shows that at the optimal value of the tuning parameter, the coefficients \( c_{uxx} \) and \( c_{uuxxx} \) still have a nontrivial joint distribution, implying that even with an optimal choice of the regularization parameter, Bayesian methods are needed to analyze the joint behaviour of these two coefficients. Lastly, one can investigate the role of using different regularization techniques for learning the parameters of the model. The existing literature on PDE-FIND contains objections to the use of LASSO when the different observations are correlated. For example, Rudy et al. [1] argue for using STRidge as a sparse regression method on the basis of performance. However, other methods exist and given information about the dynamical system to be learnt, one may wish to choose different methods.

Figure 6: Empirical distributions of relevant PDE-FIND coefficients found on ABM data of unbiased motion without proliferation. (a) Histogram of \( c_{uxx} \) trained with \( \lambda = 0.01 \), compared to true value 0.25 (black line). (b) Histogram of \( c_{uxx} \) trained with \( \lambda = 0.05 \), compared to true value 0.25 (black line). (c) Histogram of \( c_{uuxx} \) trained with optimal \( \lambda = 0.5 \), compared to true value 0.25 (black line). (d) Empirical joint distribution of \( c_{uxx} \) and \( c_{uuxxx} \) trained with optimal \( \lambda = 0.5 \), compared to location of true parameter (0.25,0) (black star).
3.3 Comparison of model predictions

We propose an explanation for the poor performance of PDE-FIND on the ABM data. The PDE-FIND algorithm solves a sparse regression function to fit linear combinations of spatial derivatives to the time derivative. When data is noiseless or near noiseless, the temporal and spatial derivatives can be accurately estimated, and so the relationship between spatial and temporal derivatives can be inferred from observed data. In this context, comparing model predictions by their performance with respect to the $L^2$-loss in the learned temporal derivative retrieves the ground truth. However, when data is noisy, different linear combinations of spatial derivatives can have a performance comparable to the ground truth. In the SI, we visualize this by selecting, for each data set, two time data points whose PDE-FIND coefficients contain terms different to the ground truth PDE. While both the ground truth and the PDE-FIND temporal derivative fit reproduce the observed temporal derivative qualitatively, there is no guarantee that this yields solutions that resemble the observed data when the PDE is numerically evaluated. We illustrate this in Figure 7 where for each of Case I, II, and III, we select two parameter combinations estimated by PDE-FIND: one that when integrated resembles the observed data, and one where the model prediction bears no resemblance to the observed data.

We note that the poor performance of the model predictions when integrated should not be surprising. PDE-FIND does not take into account modeling constraints, for example that the model be given in standard conservation form. A possible solution to this specific problem could be to write the different terms in the candidate library in flux form. More generally, though,
it is not obvious how to include all possible constraints on solutions in how the candidate library is composed. Especially given the need to use PDE-FIND to discover new solutions of the governing PDEs, there should be a low expectation of practitioners to include many such constraints into the candidate library. What this striking difference in predictive capabilities of the learned PDEs reveals is that coefficients that optimize Equation (7) do not per se perform well when optimizing a criterion that takes into account the spatial differences of the solution. One such metric is the $L^2$-loss for square-integrable functions, defined as

$$
\|f - g\|^2 = \int (f(x) - g(x))^2 \, dx. \tag{13}
$$

This contrasts with the optimization criterion in Equation (7), which we will refer to as the $L^2$ loss between the derivatives. We show that these two loss functions are indeed very different in Figure 23. As our reference data, we choose the average over all time series from the data set for unbiased motion, and for each point in a region of parameter space that only contains $c_{u_{xx}}$ and $c_{uu_{xx}}$, we approximate the two different loss functions: first between derivatives - as considered by PDE-FIND in Equation (3) - and subsequently estimate the $L^2$-loss between the integrated solution and the empirical density from Equation (13) by averaging the squared error at 5 time points,

$$
d(X^{\text{obs}}, X^{\text{sim}}) = \frac{1}{5} \sum_{i=1}^{5} \|X^{\text{obs}}_{50i} - X^{\text{sim}}_{50i}\|^2, \tag{14}
$$

where $\|\cdot\|^2_2$ is the squared $L^2$ norm for vectors. We choose this metric so that it captures both early and late time dynamics, which we noted is important when distinguishing Case I and III, for example. Here, the choice for 5 time points can be improved in future work, as more refined choices for the time discretization may yield better results, but we note that this is a reasonable choice to capture the dynamics throughout the time series. Figure 23 demonstrates a significant difference between the loss landscapes of the different error metrics. While both loss landscapes show a minimum around the true parameter value, the derivative loss landscape in Figure 23 is unable to distinguish different regions of parameter space. For example, errors on the horizontal and vertical axes are equal, meaning that PDE-FIND is unable to distinguish between these two cases. On the other hand, considering the fit of integrated solutions to reference data is much more consistent. The loss landscape shows clear regions of parameter space with increasingly higher error when the model is further away from the ground truth: the worst performance is found in the model where $u_{xx}$ is not included. This is a much desirable property, since it helps in identifying the true model from the data, which cannot be done when only looking at the observed differences in derivatives.
3.4 ABC for parameter estimation

We showcase how Bayesian methods can be used to significantly improve the quality of model predictions arising from the learned PDEs from observed data. Recall that the aim is to reduce uncertainty regarding which coefficients to include in the model, to reduce uncertainty in the estimated model parameters, and to improve the posterior predictive ability of the model by finding a posterior distribution in parameter space that takes into account properties of the actual observed density profiles. For each of the noisy test cases, that is, data sets $D^1_u$, $D^1_b$ and $D^1_p$, we apply Algorithm 2. As a proof of concept in the implementation of Algorithm 2, we choose the simplest possible ABC implementation and perform ABC rejection sampling, as detailed in the SI. To capture agreement between model predictions and the observed data, we carry out ABC inference by numerically computing, for each sampled parameter, the solution of the predicted PDE. Recall that solutions similar to the ground truth will retrieve time series in good agreement with the observed densities. Writing then $X^\text{sim}_i$, $X^\text{obs}_i$ for the simulated and observed densities at time $i$, we choose the distance function defined in Equation (14). We recall that this analyzes the goodness of fit at selected time points in the trajectory. Note that we do not evaluate the entire time series, but rather just consider an appropriate summary statistic. We choose a wide range of time points to capture the different behaviors of the model at different time scales. We use the Pakman package [11], a tool to parallelize ABC inference. The files to
carry out the Bayesian inference are found in the electronic supplementary materials.

For Case I, the identification ratios in Table 5 suggest that the two coefficients that are included in a large number of simulations are $c_{u_{xx}}$ and $c_{u_{u_{xx}}}$, while other model parameters can be confidently excluded from the model. This suggests to these two parameters in the candidate model and excluding the other parameters. For Case II, we note that the terms identified consistently by PDE-FIND are $u_x$ and $u_{xx}$. For Case III the terms are $u$, $u^2$ and $u_{xx}$. Note that PDE-FIND failed to identify one relevant model term in Case II, which is $uu_x$. Nevertheless, we proceed with the application of the rest of Algorithm 2 to find posterior distributions in parameter space. For the ABC implementation, we set $\epsilon = 0.3$ for Case I and $\epsilon = 2$ for Case II and Case III.

Figure 9 reveals that in the three cases ABC rejection can significantly reduce the uncertainty associated to the PDE coefficients. In Cases I and II, uncertainty regarding which parameter to include is completely removed, as the posterior is confidently supported only on the diffusion parameter axis in Case I and on a region where both $c_{u_x}$ and $c_{u_{xx}}$ are nonzero in Case II. Hence, Algorithm 2 removes uncertainty for the model choice. In Case III, the obtained posterior still estimates some diffusion parameters as nonzero, however, the share of nonzero $c_{u_{xx}}$ in the posterior is equal to 0.75, compared to 0.57 in the prior, meaning that the posterior has more mass close to the true parameter value. To assess the quality of the resulting posteriors in generating output that matches the observed data, we run a posterior predictive check. For each parameter value in the posterior distribution, we use numerical integration to obtain a prediction for the density at $t = 250$. We then plot the 5% and 95% quantiles and overlay this with a data trace from the data sets for Cases I, II and III in Figure 10.

The posterior predictive check in Figure 10 reveals that model solutions resemble the real-world data, even if the wide spread of parameters leads to big uncertainty bounds. Especially in Case III, with the large degree of model misspecification, we see that the noise in the data

![Figure 9: Comparisons of prior (grey) with observed data and posteriors obtained by ABC rejection sampling (red). A: Case I, $\epsilon = 0.3$. B: Case II, $\epsilon = 2$, Case III, $\epsilon = 2$.](image)
Figure 10: Posterior predictive check for the three cases. All parameter values in the posterior distribution are used to compute a prediction for the density at $t = 250$. We plot the 5% and 95% percentiles at each spatial location, and overlay with a randomly chosen data point from the data set.

directly translates to low confidence around the origin, reflecting that even when ABC is applied, some model misspecification is picked up. This problem is much less apparent in Case I, where the posterior distribution learned the correct distribution, and in Case II, where the posterior has learned the correct structure of the model. We conclude that even in the presence of model misspecification, such as in Case II, it is possible to obtain a posterior distribution in parameter space that has good predictive power. Compare this to the case of Figure 7 where the integrated model solutions failed to resemble the empirical data. We conclude that taking a Bayesian approach to supplement the PDE-FIND yields reasonable estimates, whereas the point estimates otherwise obtained using PDE-FIND would not have matched the observed data.

4 Conclusions and Outlook

This paper proposed a framework to perform uncertainty quantification for implementing EQL methods on noisy data. By simulating a large amount of synthetic data from an ABM, we show that the PDE-FIND algorithm produces predictions that have great variability both in the structure of the learned models as well as in the parameter values. We conclude that when data is corrupted with large amounts of noise, the PDE-FIND algorithm can learn models that are uninformative of the correct governing equation. In contrast, when we control for the noise and present the PDE-FIND algorithm with data that is sufficiently detailed and with little noise, the structure of the learned equations is correct, but there is still variability in the learned parameters. By using a combined approach of PDE-FIND and ABC, it is possible to obtain posterior distributions in parameter space that translate directly into uncertainty quantification
in model predictions. We believe that this can enable practitioners to both confidently report the range of possible models that can explain a given data set as well as obtain practical error bounds for model predictions.

After applying ABC on the PDE-FIND generated prior distribution, not all model misspecification was removed from the observations. Researchers may find it worthwhile to find different, more efficient implementations of the ABC algorithm used here. For example, one may consider using a Sequential Monte Carlo (ABC-SMC) approach [5], which carefully propagates the prior through a series of intermediate distributions to a final, posterior distribution. There is a wide range of studies, see for example [12] that elucidate how ABC-SMC is well-suited to find complicated posterior distributions from a wide prior distribution. We expect that better tuned approaches to ABC can help improve the quality of the posteriors found in this work. Likewise, within the given ABC implementation, one may wish to consider a more detailed set of summary statistics, or a different choice of the distance function. Nevertheless, we find that even the simplest possible ABC implementation yields an effective and practical uncertainty quantification. For practitioners wishing to obtain these error bounds without needing to consider the mathematical intricacies of more sophisticated ABC implementations, we believe that our ABC implementation provides an appropriate tool.

Another, orthogonal, direction of research concerns the application of denoising methods to the data prior to training the PDE-FIND algorithm. In this paper, we controlled for the intrinsic noise in the observations to compare the performance of PDE-FIND in predictions. In applications of PDE-FIND to real-life data, practitioners will typically not be able to control for the amount of observations noise in their data. Hence, methods to smooth data and reduce observation noise may be an attractive extension to PDE-FIND. When using spatial subsampling as a method to denoise data, we found that while the degree of model misspecification was decreased, the structure of the model was still highly variable with big uncertainty in the learned parameters. We note that even state-of-the-art methods to reduce noise will not retrieve the true signal behind the data, and so uncertainty quantification is still important, since ultimately the precise relation between noise in the data and uncertainty in the learned equations is unknown. We highlight two methods to denoise data. Convolution with a Gaussian kernel is ubiquitous in the information sciences and is often used to denoise data. In addition, an implementation implementation PCA for EQL has been proposed by Li et al. [7]. In the supplementary information, we show that these two methods still provide variable and unreliable predictions, meaning that with the present selection of denoising methods uncertainty quantification remains necessary for applications of PDE-FIND to realistic biological data.
This article proposed as an explanation for the poor performance of PDE-FIND on noisy data that the loss function used by PDE-FIND does not take into account spatial information of the predictions made by the learned model. To avoid the problem that predictions that may fit the temporal derivative well in the presence of noise, but fail to capture important features of the observed data, we show that Bayesian methods can help incorporate this information into predictions. PDE-FIND is not the only method for learning differential equations from data. When data is heavily corrupted by data, it may be worthwhile using a Deep Neural Network approach [6, 3] instead of the PDE-FIND algorithm to simultaneously learn the spatio-temporal dynamics while correcting for the presence of noise.

References

[1] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4), 2017.

[2] John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. Learning differential equation models from stochastic agent-based model simulations, 2020.

[3] John H. Lagergren, John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data, 2020.

[4] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[5] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[6] John H. Lagergren, John T. Nardini, G. Michael Lavigne, Erica M. Rutter, and Kevin B. Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2234):20190800, 2020.

[7] Jun Li, Gan Sun, Guoshuai Zhao, and Li-wei H. Lehman. Robust low-rank discovery of data-driven partial differential equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):767–774, Apr. 2020.
[8] Mikael Sunnåker, Alberto Giovanni Busetto, Elina Numminen, Jukka Corander, Matthieu Foll, and Christophe Dessimoz. Approximate bayesian computation. *PLOS Computational Biology*, 9(1):1–10, 01 2013.

[9] M J Simpson, B D Hughes, and K A Landman. Diffusing populations: Ghosts or folks? *Australasian Journal of Engineering Education*, 15(2):59–68, 2009.

[10] Wessel N. van Wieringen. Lecture notes on ridge regression, 2020.

[11] Thomas F. Pak, Ruth E. Baker, and Joe M. Pitt-Francis. Pakman: a modular, efficient and portable tool for approximate bayesian inference. *Journal of Open Source Software*, 5(47):1716, 2020.

[12] Sarah Filippi, Chris Barnes, Julien Cornebise, and Michael P. H. Stumpf. On optimality of kernels for approximate bayesian computation using sequential monte carlo, 2012.
Supplementary Information: Bayesian uncertainty quantification for data-driven equation learning

Variability of PDE-FIND coefficients

In this section, we report the full data on the learned PDE-FIND coefficients trained on the different data sets. This includes the identification ratios for all coefficients in all experiments as well as plots of the joint distributions of all the learned parameters.

Full data on identification ratios

We report the identification ratios for the problems considered in the main text together with the additional experiments performed for this supplementary information in the table below.

| Experiment        | $c_1$ | $c_u$ | $c_{u^2}$ | $c_{u_x}$ | $c_{u^2,u_x}$ | $c_{u_{xx}}$ | $c_{u^2,u_{xx}}$ |
|-------------------|-------|-------|-----------|-----------|---------------|--------------|------------------|
| STRidge $D_u^1$   | 0.001 | 0.0   | 0.002     | 0.0       | 0.008         | 0.008        | 0.826            | 0.199            |
| STRidge $D_u^2$   | 0.0   | 0.0   | 0.0       | 0.0       | 0.008         | 1.0          | 0.0              | 0.0              |
| STRidge $D_b^1$   | 0.0   | 0.0   | 0.0       | 0.999     | 0.0           | 0.002        | 0.0              | 0.0              |
| STRidge $D_b^2$   | 0.0   | 0.0   | 0.0       | 1.0       | 0.0           | 0.0          | 0.0              | 0.0              |
| $D_p$, $d_{tol} = 5e-3$ | 0.234 | 0.2   | 0.009     | 0.005     | 0.001         | 0.007        | 0.993            | 0.012            |
| $D_p$, $d_{tol} = 1e-2$ | 0.013 | 0.659 | 0.482     | 0.002     | 0.002         | 0.007        | 0.571            | 0.014            |
| $D_p$, $d_{tol} = 5e-3$ | 0.0   | 1.0   | 0.0       | 0.0       | 0.0           | 1.0          | 0.12             | 0.0              |
| $D_p$, $d_{tol} = 1e-2$ | 0.0   | 1.0   | 0.0       | 0.0       | 0.0           | 1.0          | 0.0              | 0.0              |
| rPCA $D_u^1$      | 0.0   | 0.408 | 0.69      | 0.0       | 0.01          | 0.001        | 0.384            | 0.143            |
| Convolution $D_u^1$ | 0.0   | 0.003 | 0.222     | 0.0       | 0.001         | 0.765        | 0.008            | 0.017            |
| Convolution $D_b^1$ | 0.0   | 0.002 | 0.109     | 0.882     | 0.028         | 0.005        | 0.181            | 0.002            |
| Convolution $D_p^1$ | 0.0   | 0.922 | 0.086     | 0.0       | 0.001         | 0.046        | 0.002            | 0.026            |
| Subsampling $D_u^1$ | 0.0   | 0.0   | 0.003     | 0.001     | 0.002         | 0.007        | 0.998            | 0.365            |

Table 5: Values of $a_j$ for all experiments, where $j$ is given as the term in the PDE in each column.

Empirical distribution of all learned PDE-FIND parameters

In this section we report the pairwise distribution of all learned PDE-FIND parameters in Case I, Case II, Case III, both in the low-noise and high-noise regime. All terms in the library are ordered as: $1, u, u^2, u_x, uu_x, u^2 u_x, u_{xx}, uu_{xx}, u^2 u_{xx}$.
Figure 11: Pairwise correlation plots for coefficients trained on $\mathcal{D}^1_u$
Figure 12: Pairwise correlation plots for coefficients trained on $D^2_{uu}$
Figure 13: Pairwise correlation plots for coefficients trained on $D_b^1$
Figure 14: Pairwise correlation plots for coefficients trained on $D^2_b$
Figure 15: Pairwise correlation plots for coefficients trained on $D_p^1$.
Figure 16: Pairwise correlation plots for coefficients trained on $D_p^2$
Denoising approaches

In this section, we detail how denoising approaches were used to denoise the ABM data and show the variability of the learned coefficients using each of these approaches.

Rescaled Robust PCA

Li et al. [7] propose an approach to perform PDE-FIND on noisy data. Mathematically, denote the data matrix $U \in \mathbb{C}^{n \times m}$ given on a discretized domain $x \in [a, b]$ and $t \in [0, T]$ for the discretely and noisily sampled measurements from the function $u(x, t)$ that satisfies the PDE. The authors observe that the underlying data $u(x, t)$ is often low-rank and so they suppose that $U$ and its time derivative $U_t$ can be decomposed as

$$U = Z + E_1, \quad U_t = \mathcal{D}(Z, Q)\xi + E_2,$$

(15)

where $Z$ and $\mathcal{D}(Z, Q)$ have low rank and $E_1, E_2$ are sparse. Informally, $Z$ represents the clean data, $\mathcal{D}(Z, Q)$ the clean derivative matrix, and $E_1, E_2$ the perturbations of the clean data and derivatives respectively. The method developed in [7] is to optimally find $Z$ and $\mathcal{D}(Z, Q)$ from the data matrix. This problem is addressed by solving the optimization problem

$$\min_{\xi, Z, E_1, E_2} \|Z\|_* + \|\mathcal{D}(Z, Q)\xi\|_* + \lambda_1 \mathcal{R}(\xi) + \lambda_2 \|E_1\|_1 + \lambda_3 \|E_2\|_1$$

(16)

such that $U = Z + E_1$ and $U_t = \mathcal{D}(Z, Q)\xi + E_2$. Additionally, $\mathcal{R}(\xi)$ is a sparse regularization of the parameters $\xi$, and $\lambda_i, i = 1, 2, 3$ are positive constants. For details on this nonconvex optimization problem, we refer to [7]. The essence of the technique is that the original data is processed first with an algorithm called Robust Principal Component Analysis (rPCA) to obtain $Z$, after which Low-rank STRidge (LrSTR) is used to construct $\mathcal{D}(Z, Q)$ from $Z$ and $U_t$ is denoised prior to applying PDE-FIND. The final algorithm, which uses rPCA and LrSTR in tandem, is called Double Low-rank Sparse Regression (DLrSR). While the examples given by Li et al. in [7] prove capable of retrieving the true parameters of the PDEs with remarkable accuracy, the choice of regularization parameters is non-trivial and makes implementation challenging, since the many degrees of freedom complicate choosing a robust parameter set. As shown in Algorithm 1 of [7], $\lambda_2$ corresponds to the penalty in rPCA. One must choose $\lambda_2$ carefully, since different sizes of this parameter can yield unfavorable predictions when used in tandem with PDE-FIND. We start by plotting the output of empirical data processed by rPCA at two different values of $\lambda_2$. As seen in Figure 17, choosing a small value of $\lambda_2$ in rPCA acts in favor of interpreting the entire signal as noise and output $Z = 0$, effectively rendering the output useless. Larger values of $\lambda_2$
Figure 17: Plots of denoised data $Z$ (points) compared to original data (solid lines), plotted at $t = 0$ (red) and $t = 1000$ (black). (a) $\lambda_2 = 0.001$ (b) $\lambda_2 = 0.015$ (c) $\lambda_2 = 0.06$ (d) $\lambda_2 = 0.1$. Small values of $\lambda_2$ favor interpreting the entire signal as noise and output $Z = 0$. Larger values of $\lambda_2$ do not penalize the noise and return a solution that is identical to the original signal. In the intermediate regime, the order of magnitude of the rPCA output is distorted.

Small values of $\lambda_2$ favor interpreting the entire signal as noise and output $Z = 0$. Larger values of $\lambda_2$ do not penalize the noise and return a solution that is identical to the original signal. In the intermediate regime, the order of magnitude of the rPCA output is distorted. In the intermediate regime, the order of magnitude of the rPCA output is distorted, as rPCA interprets the data matrix $U$ as sparse and so the maximum of the rPCA output for the densities has comparable order of magnitude $t = 0$ and $t = 500$, effectively obscuring the biological mechanism at play. However, the behavior of the rPCA output in the intermediate regime is qualitatively good, so we suggest to scale the rPCA output so that it is in the appropriate order of magnitude. Since the maximum of the density is influenced by noise too, it should not be the only factor considered in the scaling. Rather, the mean of the data should be included too. Heuristically, we let $Z$ be the output of rPCA and $U$ the original matrix, such that $Z_i$, $U_i$ denote the $i$-th row of $Z$, $U$ respectively (corresponding to the $i$-th time point). Let $\overline{Z_i}$, $\overline{U_i}$ be the means of the vectors $Z_i$, $U_i$ respectively and $Z_i^{\text{max}}$, $U_i^{\text{max}}$ be their maxima. Then, we re-scale $Z$ to $\tilde{Z}$ such that

$$\tilde{Z}_i = \frac{1}{2} \left( \frac{U_i}{Z_i} + \frac{U_i^{\text{max}}}{Z_i^{\text{max}}} \right) \cdot Z_i.$$  

(17)

Focusing on intermediate values of $\lambda_2$, we now plot the rescaled rPCA output compared to the original solutions in Figure 18. From these plots we decide to use the rescaled output of the DLrSR algorithm in [7] on dataset $D_i^u$, choosing for the rPCA the value of $\lambda_2 = 0.035$.

Table 5 contains the values of $a$ for the coefficients found with the rPCA method. Comparing these values with the values found using STRidge for trained on $D_i^u$ reveals that the method has a much higher variability of terms that are included and excluded in the resulting PDE model. Moreover, Figure 19 shows that coefficients are correlated in a complicated, non-trivial, manner.
Figure 18: Plots of rescaled denoised data $\tilde{Z}$ (points) compared to original data (solid lines), plotted at $t = 0$ (red) and $t = 1000$ (black) chosen at values of $\lambda_2$ in the intermediate parameter regime. (a) $\lambda_2 = 0.0175$ (b) $\lambda_2 = 0.03$ (c) $\lambda_2 = 0.035$ (d) $\lambda_2 = 0.04$. The order of magnitude of the maximum of the rPCA output is no longer distorted.

The correlation plot also reveals the problem with taking point estimates. In the unbiased case, for example, the pairs of the predicted coefficients for the terms $u_{xx}$ and $uu_{xx}$ are contained in a region of parameter space where either of the two terms is discovered or when both terms are picked up and are not small. That implies that any point estimate when derived from this data is unreliable and careful consideration needs to be given to the correlations between terms. Figure 19b also shows that the support of PDE-FIND predictions for the diffusion coefficient obtained using DLrSR on $D^u_1$ is similar in size to that obtained with Naive STRidge. This means that with the intrinsic noise in the sample, rPCA yields estimates for the diffusion coefficient that are no less variable than the original PDE-FIND method. Table 5 summarizes the values of the coefficients $a_j$. It can be seen that no terms in the PDE are conclusively identified, as all terms for $a$ are either small or in the intermediate range. Most saliently, rPCA identifies the diffusion coefficient in only 38% of samples in the data set. Hence, even when using a state-of-the-art method of denoising observations it is important to statistically quantify uncertainty in the learned coefficients.

**Convolution with Gaussian kernel: an alternative approach to denoising**

In many applications, noisy data can be successfully smoothed using a suitable kernel. We choose to smooth our data using a Gaussian kernel with standard deviation 2, so that it is roughly equivalent to the scale used in the spatial discretization. Summary values for $a$ are found in Table 5. While the convolution is highly successful in recovering one of the relevant terms in the dynamical system (for example, the diffusion term in $D^u_1$, the bias term in $D^b_1$ and the proliferation term in $D^p_1$, the method fails to recover the diffusion term regularly in $D^b_1$ and
Figure 19: Joint distribution of $c_{ux}$ and $c_{uxxx}$ trained on ABM data of unbiased motion without proliferation, where each sample is one time series. Location of true parameter value is indicated by black star. (a) Naive application of STRidge on the ABM data. (b) ABM data is pre-processed with rPCA as detailed in SI before training PDE-FIND. (c) ABM data is pre-processed by smoothing with a Gaussian kernel with $\sigma^2 = 2$ before training PDE-FIND.

Tuning of PDE-FIND hyperparameters

Following Nardini et al. [2], we implement cross-validation for finding the optimal value of the regularization parameter in the STRidge algorithm. As training data, we randomly select a subset of 10 data points from the ensemble. For each $\lambda \in \{0.05 \cdot x, 0 \leq x \leq 50\}$, we perform leave-one-out cross validation, that is, we train PDE-FIND on the left-out data sample, and compute the $L^2$-loss of the integrated solution at $t = 250$ with the samples not used for training at $t = 250$. This gives an optimal value of $\lambda = 0.5$. To illustrate the role of increasing the tuning parameter, Table 6 shows the values of the $a_j$ coefficients at different values of $\lambda$. For these values, the main text reports the empirical distribution of the PDE-FIND coefficients. Note that $\lambda = 0.01$ equals the original experiment.

| Value of $\lambda$ | $c_1$   | $c_u$   | $c_{u^2}$ | $c_{ux}$ | $c_{u^2ux}$ | $c_{uxx}$ | $c_{uxxx}$ | $c_{u^2uxxx}$ |
|-------------------|---------|---------|-----------|----------|-------------|-----------|------------|---------------|
| 0.01              | 0.001   | 0.0     | 0.002     | 0.0      | 0.008       | 0.008     | 0.826      | 0.199         |
| 0.05              | 0.0     | 0.0     | 0.011     | 0.001    | 0.01        | 0.019     | 0.996      | 0.34          |
| 0.1               | 0.0     | 0.0     | 0.003     | 0.001    | 0.002       | 0.007     | 0.998      | 0.365         |
| 0.5               | 0.0     | 0.0     | 0.0       | 0.0      | 0.0         | 0.0       | 0.996      | 0.098         |

Table 6: Values of $a_j$ for all experiments, where $j$ is given as the term in the PDE in each column.
Comparison of model predictions

When data is noisy, different linear combinations of spatial derivatives can outperform the temporal derivative when measured according to the $L^2$-loss. We visualize this by selecting two time data points for each of the data sets whose PDE-FIND coefficients contain terms different to the ground truth PDE. We fit the predicted PDE to the temporal derivative at $t = 20$ and at $t = 250$ and compare it to the fitted ground truth using the estimated spatial derivatives. Figure 20 compares the observed temporal derivatives with these fitted values for the temporal derivative. In five of the six plots, both the ground truth and the PDE-FIND temporal derivative fit reproduce the observed temporal derivative qualitatively. A priori, there is no guarantee that the $L^2$-loss is optimized by the fitted ground truth.

However, Figure 22 shows that some of the mis-specified coefficients do not yield predictions that are in close agreement with the observed density data. This illustrates the finding in the main text that a close agreement between the fitted temporal derivatives need not imply that the model predictions are adequate.

![Figure 20](image)  
Figure 20: Comparisons of the observed time derivative at $t = 20$ with predicted time derivative using misspecified PDE-FIND coefficients (blue) compared to fitted ground truth (red). Fitted PDEs in panels (a)-(f) match the panels in Figure 22.
Figure 21: Comparisons of the observed time derivative at $t = 250$ with predicted time derivative using misspecified PDE-FIND coefficients (blue) compared to fitted ground truth (red). Fitted PDEs in panels (a)-(f) match the panels in Figure 22.

**Approximate Bayesian Computation**

In the main text we identify a significant difference between the loss landscapes of the different error metrics for the computed derivative and spatial solution of the PDE. Here, we show that we may also take as a reference point a single, noisy, observation instead of the average of the observations in the ensemble, albeit that the quality of the resulting posterior is decreased. For completeness, we plot the different loss landscapes in Figure 23.

The high amount of noise in the single measurement case means that the loss landscape when considering the derivative does not have a minimum around the true parameter value, implying that minimizing with respect to this norm is not guaranteed to converge to the correct solution when many measurements are being taken. This is very different in the case of low noise, the loss landscape clearly has a minimum around the true parameter value. On the other hand, comparing the fit of integrated solutions is much more consistent in the case of a single, very noisy measurement and retrieves low measurement error in a region around the true parameter value. When the loss is considered on the averaged data, this region is smaller, and the range of the error is bigger, suggesting that this loss function is better able to distinguish between solutions. The poor performance of the model predictions when integrated should not be surprising. PDE-FIND does not take into account modeling constraints, for example that the model be mass-conserving. A possible solution to this specific problem could be to write the
Figure 22: Comparisons of model predictions with misspecified PDE-FIND predictions (blue) compared to ground truth (red). (a) Unbiased motion, misspecified model with $c_{uu,xx} = 0.104$ and $c_{uuu,xx} = 0.26$ (b) Unbiased motion, misspecified model with $c_{uu,xx} = 0$ and $c_{uuu,xx} = 1.52$. (c) For biased motion, misspecified model with $c_{uu,xx} = 0.20$ and $c_{uuu,xx} = 0.026$. (d) For biased motion, misspecified model with $c_{uu,xx} = 0$ and $c_{uuu,xx} = 0.64$. (e) For proliferation and unbiased motion, misspecified model with $c_{uu,xx} = 0$ and $c_{u} = 0.0013$. (f) For proliferation and unbiased motion, misspecified model with $c_{uu,xx} = 0.12$ and $c_{u} = 0$.

different terms in the candidate library in flux form. More generally, though, it is not obvious how to include all possible constraints on solutions in how the candidate library is composed. Especially given the need to use PDE-FIND to discover new solutions of the governing PDEs, there should be a low expectation of practitioners to include many such constraints into the candidate library. If a modeler were able to construct the candidate library exactly to incorporate all constraints, perhaps other methods of equation fitting for the relevant equation parameters could be more appropriate, avoiding the need to use EQL methods.

Approximate Bayesian Computation is a popular tool to perform likelihood-free Bayesian inference. This is useful in cases where the likelihood function of the observed data given the model parameters is intractable or difficult to compute, but simulation given model parameters is feasible. In ABC, a prior $\pi$ is specified over parameter space that reflects prior information about the model parameter $\theta$. In each step, $\theta \sim \pi$ is drawn, and a model simulation is done using $\theta$, yielding the simulated data $X^{\text{Sim}}$. The simulated data is compared to the observed data $X^{\text{Obs}}$
Figure 23: Heatmap showing $L^2$ error for each pair of $c_{uu_{xx}}$ and $c_{u_{xx}}$, overlayed with actual PDE-FIND predictions. (a) Derivative $L^2$ loss when compared to a single point. (b) Derivative $L^2$ loss compared to average. (c) Comparison is done with respect to a single data point. (d) Comparison is done with respect to the average over all of the time series in the ensemble.

...according to a distance function $d(X^{\text{Sim}}, X^{\text{Obs}})$ provided by the practitioner. Given a tolerance $\epsilon > 0$, the sampled $\theta$ is accepted whenever $d(X^{\text{Sim}}, X^{\text{Obs}}) < \epsilon$. The process finishes when an pre-specified number of parameter samples have been accepted. The posterior distribution is then given by the accepted parameter samples. We refer to [8] for the details.

In the main text, we choose a two-dimensional parameter space that contains only $u_{xx}$ and $uu_{xx}$ as terms in the PDE and parametrize the normal marginals as used in in the main text by computing the empirical means of $c_{u_{xx}}$ and $c_{uu_{xx}}$, conditional on them being nonzero, and assign the following relative weights $m^j_0$ for the atom at 0 for coefficient $j \in \{u_{xx}, uu_{xx}\}$:

$$m^j_0 = \frac{\sum_{i=1}^{N} 1(c^j_i = 0)}{N - \sum_{i=1}^{N} 1(c_{u_{xx}}^i c_{uu_{xx}}^i = 0)}.$$  

(18)
This gives the prior distribution shaded in grey in each of the panels of Figure 9 in the main text.

Letting $X_i^{\text{sim}}$, $X_i^{\text{obs}}$ be the simulated and observed densities at time $i$, we choose the distance function

$$d(X^{\text{obs}}, X^{\text{sim}}) = \sum_{i=1}^{5} \| X_{50i}^{\text{obs}} - X_{50i}^{\text{sim}} \|_2^2,$$

where $\| \cdot \|_2$ represents the $L^2$-norm. We perform two experiments. In the first, we choose for $X^{\text{obs}}$ the time series used to compute the loss landscape of 23 and choose $\epsilon = 0.8$ for the ABC rejection tolerance. For the second experiment, we choose for $X^{\text{obs}}$ the average over all time series in the ensemble and set $\epsilon = 0.3$. 

43