International Statistical Review

International Statistical Review (2023), **91**, 3, 464–492 doi: 10.1111/insr.12554

A Review of Data-Driven Discovery for Dynamic Systems

Joshua S. North[1], **Christopher K. Wikle**[2] and **Erin M. Schliep**[3]

[1]*Department of Earth and Environmental Sciences, Lawrence Berkeley National Laboratory, Berkeley, CA, USA*
[2]*Department of Statistics, University of Missouri, Columbia, MO, USA*
[3]*Department of Statistics, North Carolina State University, Raleigh, NC, USA*
Corresponding to: Joshua North, Department of Earth and Environmental Sciences, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. Email: jsnorth@lbl.gov

Summary

Many real-world scientific processes are governed by complex non-linear dynamic systems that can be represented by differential equations. Recently, there has been an increased interest in learning, or discovering, the forms of the equations driving these complex non-linear dynamic systems using data-driven approaches. In this paper, we review the current literature on data-driven discovery for dynamic systems. We provide a categorisation to the different approaches for data-driven discovery and a unified mathematical framework to show the relationship between the approaches. Importantly, we discuss the role of statistics in the data-driven discovery field, describe a possible approach by which the problem can be cast in a statistical framework and provide avenues for future work.

Key words: differential equations; dynamic equation discovery; probabilistic dynamic equation discovery.

1 Introduction

Recently, there has been a push from within computer science, physics, applied mathematics and statistics to learn the governing equations in complex dynamic systems parameterised through dynamic equations. There are a variety of reasons researchers may want to know the underlying laws driving a system—to reinforce their assumptions, to uncover extra information about the system or to produce a more realistic mathematical representation for the system. Historically, scientists have relied on their ability to represent physical systems using mathematical equations in the form of dynamic equations. Dating back to at least the inference of equations describing the motion of orbital bodies around the sun based on the positions of celestial bodies (Legendre, 1806; Gauss, 1809), dynamic equations have been used to model the evolution of complex processes (e.g. the use of susceptible, infected and recovered models for epidemics) and have become ubiquitous across virtually every area of science and engineering. More recently, the idea of *learning*, rather than representing, the equations underlying complex systems has become popular due in part to the initial ideas presented by Bongard & Lipson (2007),

Schmidt & Lipson (2009), and Brunton *et al.* (2016). We review methods used to discover the governing equations driving complex, potentially non-linear, processes, often referred to as *data-driven discovery.*

Consider the general dynamic equation describing the evolution of a continuous process $\{\mathbf{u}(\mathbf{s}, t) : \mathbf{s} \in D_s, t \in D_t\}$,

$$\mathbf{u}_{t^{(J)}}(\mathbf{s}, t) = \mathcal{M}\big(\mathbf{u}(\mathbf{s}, t), \mathbf{u}_x(\mathbf{s}, t), \mathbf{u}_y(\mathbf{s}, t), \ldots, \mathbf{u}_{t^{(1)}}(\mathbf{s}, t), \ldots, \mathbf{u}_{t^{(J-1)}}(\mathbf{s}, t), \boldsymbol{\omega}(\mathbf{s}, t)\big), \quad (1)$$

where the vector $\mathbf{u}(\mathbf{s}, t) = [u(\mathbf{s}, t, 1), u(\mathbf{s}, t, 2), \ldots, u(\mathbf{s}, t, N)]' \in \mathbb{R}^N$ denotes the state of the N-dimensional system at location $\mathbf{s}$ and time t, $\mathbf{u}_{t^{(j)}}(\mathbf{s}, t)$ is the j-th order temporal derivative of $\mathbf{u}(\mathbf{s}, t)$, J denotes the highest order of the temporal derivative, $\mathcal{M}(\cdot)$ represents the (potentially non-linear) evolution function, and $\boldsymbol{\omega}(\mathbf{s}, t)$ represents covariates that might be included in the system. We denote partial derivatives by a subscript; that is $\dfrac{\partial \mathbf{u}(x, t)}{\partial x} = \mathbf{u}_x(x, t)$ and $\dfrac{\partial \mathbf{u}(x, t)}{\partial t} = \mathbf{u}_t(x, t)$, for example. Here, $\mathbf{s} \in \{\mathbf{s}_1, \ldots, \mathbf{s}_S\} = D_s$ is a discrete location in the domain with $|D_s| = S$ and $t \in \{1, \ldots, T\} = D_t$ is the realisation of the system at discrete times where $|D_t| = T$. Equation (1) is composed of partial derivatives of the system with $D_s \in \mathbb{R}^2$ and $\mathbf{s} = (x, y)'$ (although this can be simplified to $\mathbb{R}^1$ with $\mathbf{s} = x$ or generalised to higher dimensions) and is often referred to as a partial differential equation (PDE). Example 1 details how the general formulation for a PDE relates to a classical example, Burgers' equation. Removing the spatial component from (1) results in a temporal ordinary differential equation (ODE),

$$\mathbf{u}_{t^{(J)}}(t) = \mathcal{M}(\mathbf{u}(t), \mathbf{u}_{t^{(1)}}(t), \ldots, \mathbf{u}_{t^{(J-1)}}(t), \boldsymbol{\omega}(t)), \quad (2)$$

where $\mathcal{M}(\cdot)$ is composed solely of derivatives of the components in time (i.e. no partial derivatives). This review will focus on methods to discover the evolution function $\mathcal{M}$ for both PDEs (e.g. Equation 1) and ODEs (e.g. Equation 2).

The goal of data-driven discovery is to learn the governing equation(s) in (1) and (2)—specifically the (non)linear function $\mathcal{M}(\cdot)$—having only observed noisy realisations of the true process $\mathbf{u}(\mathbf{s}, t)$ (i.e. true derivatives are unknown). Broadly, we group the approaches used for data-driven discovery into three categories—classical sparse methods, classical symbolic methods, and deep modeling methods using either symbolic or sparse regression techniques—but recognise other categorisations are possible. The first approach uses sparse regression where a library of potential solutions are proposed and the correct solution set is obtained by regularisation based techniques, resulting in a sparse solution. The second uses symbolic regression where the solution is learned, or generated, through the estimation procedure. The third uses deep models to facilitate the discovery process of the previous two approaches (e.g. symbolic regression using deep models). As this is an active area of research, we refer the reader to the special issue Epureanu & Ghadami (2022) for emerging areas of research and applications.

While less common than the deterministic counterparts, methods to quantify uncertainty in the discovered equations have been proposed (Zhang & Lin, 2018; Niven *et al.*, 2020; Yang *et al.*, 2020; Fasel *et al.*, 2022; North *et al.*, 2022a, 2022b; Bhouri & Perdikaris, 2022). However, these methods generally do not account for uncertainty in the observed data, missing a vital piece of the statistical puzzle. We draw parallels between traditional statistical models and data-driven discovery, discussing how statistical models can be formulated for data-driven discovery and highlighting possible improvements to the methods.

The remainder of the paper is organised as follows. In Section 2, we review sparse regression methods for data-driven discovery, which are sub-categorised into deterministic and probabilistic approaches. In Section 3, we review symbolic methods for data-driven discovery. In Section 4, we review deep modeling approaches for data-driven discovery, which are sub-divided into

methods approximating and discovering the underlying dynamics. In Section 5, we show how the problem can be formulated in a statistical paradigm, and in Section 6, we review a possible method of data-driven discovery using a fully probabilistic approach.

Example 1: Burgers' equation

To motivate the setup with a classic PDE, consider Burgers' equation

$$u_t(s, t) = -u(s, t)u_x(s, t) + \nu u_{xx}(s, t), \tag{3}$$

which is a simplification of the Navier-Stokes equations describing the speed of a fluid (Bateman, 1915; Burgers, 1948), where $u(s, t)$ is the speed of the fluid at location $s = (x)$ and time t and ν is the viscosity of the fluid. To frame (3) in the context of (1), let $N = 1$ and $J = 1$ so the left-hand side (LHS) of (1) is now $u_t(s, t)$. For the right-hand side (RHS) of (1), the non-linear function $\mathcal{M}(\cdot) = -u(s, t)u_x(s, t) + \nu u_{xx}(s, t)$ will take arguments $u(s, t)$, $u_x(s, t)$, and $u_{xx}(s, t)$, and ν will be a learned parameter value.

An example of the solution surface $u(\mathbf{s}, t)$ with initial condition $u(s, 0) = exp\{-(s + 2)^2\}$ and $\nu = 0.1$ with 256 spatial locations across 101 time points where $D_s = [-8, 8]$ and $D_t = [0, 10]$ is shown in Figure 1(a). The solution is generated using spectral differentiation and the *Tsit5* (Tsitouras, 2011) numerical solver from the Julia package *DifferentialEquations.jl* (Rackauckas & Nie, 2017), and source code for generating this data can be found at https://github.com/jsnowynorth/BayesianDiscovery.jl.

To simulate measurement error, which we refer to as ζ *noise*, we add white noise to the solution. Specifically, let $v(\mathbf{s}, t) = u(\mathbf{s}, t) + \zeta\boldsymbol{\epsilon}(\mathbf{s}, t)$, where $u(\mathbf{s}, t)$ is the simulated data, $\boldsymbol{\epsilon}(\mathbf{s}, t) \sim N(\mathbf{0}, \sigma^2)$ is the additive noise, σ is the standard deviation of the simulated data $u(\mathbf{s}, t)$, and ζ is the proportion of noise ranging from 0 to 1. For all the following examples, we take $\zeta = 0.01$ (Figure 1b) and use the noisy data when estimating model parameters. For cases with larger artificial noise, see the examples provided in Rudy *et al.* (2017) and North *et al.* (2022b). Note that $v(\mathbf{s}, t)$ is analogous to the data model that will be introduced in Section 5.

2 Sparse Regression

Sparse regression approaches for dynamic discovery of ODEs and PDEs are fundamentally the same. We formulate the general approach for PDEs (e.g. Equation 1), noting that the approach for ODEs (e.g. Equation 2) is equivalent but with only one spatial location (i.e. $S = 1$). First, consider rewriting (1) as a linear (in parameters) system

$$\mathbf{u}_{t^{(J)}}(\mathbf{s}, t) = \mathbf{f}(\mathbf{u}(\mathbf{s}, t), \ldots)\mathbf{M},$$

where $\mathbf{M}$ is a $D \times N$ *sparse* matrix of coefficients and $\mathbf{f}(\cdot)$ is a length-D vector-valued non-linear transformation function termed the *feature library*. The input of the arguments for $\mathbf{f}(\cdot)$ are general and contain terms that *potentially* relate to the system (e.g. advection term, polynomial terms and interactions). Sparse identification seeks to identify relevant terms of $\mathbf{M}$, thereby identifying the components of $\mathbf{f}$ that drive the system and discovering the governing dynamics.

We denote the matrix of all data (all components at all time points) for the j-th derivative of the system as

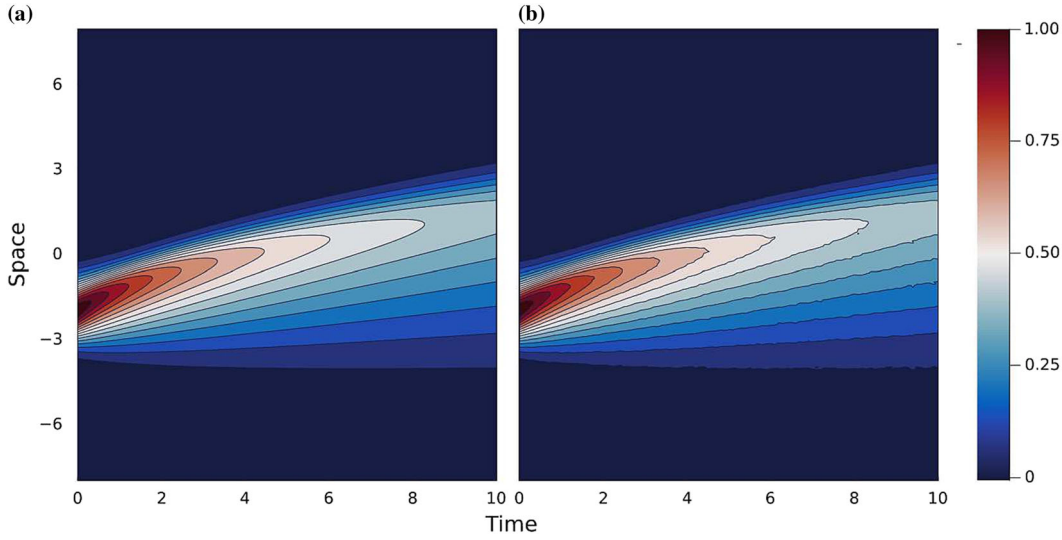

FIGURE 1. *Data simulated from Burgers' equation,* $u_t(s,\,t) = -u(s,\,t)u_x(s,\,t) + \nu u_{xx}(s,\,t)$*, where* $\nu = 0.1$*,* $D_s = [-8, 8]$*,* $D_t = [0, 10]$*,* $|D_s| = 256$*, and* $|D_t| = 101$*. Left (a) is the true data* $u(s,\,t)$ *and right (b) is the noisy data* $v(s,\,t)$ *with* $\zeta = 0.01$

$$\mathbf{U}_{t^{(j)}} = \begin{bmatrix} \mathbf{u}_{t^{(j)}}(\mathbf{s}_1,\,1) \\ \mathbf{u}_{t^{(j)}}(\mathbf{s}_1,\,2) \\ \vdots \\ \mathbf{u}_{t^{(j)}}(\mathbf{s}_S,\,T) \end{bmatrix} = \begin{bmatrix} u_{t^{(j)}}(\mathbf{s}_1,\,1,1) & u_{t^{(j)}}(\mathbf{s}_1,\,1,2) & \dots & u_{t^{(j)}}(\mathbf{s}_1,\,1,N) \\ u_{t^{(j)}}(\mathbf{s}_1,\,2,1) & u_{t^{(j)}}(\mathbf{s}_1,\,2,2) & \dots & u_{t^{(j)}}(\mathbf{s}_1,\,2,N) \\ \vdots & \vdots & \vdots & \\ u_{t^{(j)}}(\mathbf{s}_S,\,T,\,1) & u_{t^{(j)}}(\mathbf{s}_S,\,T,\,2) & \dots & u_{t^{(j)}}(\mathbf{s}_S,\,T,\,N) \end{bmatrix}.$$

The response matrix is then $\mathbf{U}_{t^{(J)}}$ of size $(ST) \times N$, and we generically denote the feature library as

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}(1,\,\mathbf{u}_{t^{(0)}}(\mathbf{s}_1,\,1),\,\dots,\,\mathbf{u}_{t^{(J)}}(\mathbf{s}_1,\,1),\,\mathbf{u}_x(\mathbf{s}_1,\,1),\,\mathbf{u}_y(\mathbf{s}_1,\,1),\,\mathbf{u}_{xx}(\mathbf{s}_1,\,1),\,\dots,\,\boldsymbol{\omega}(\mathbf{s}_1,\,1)) \\ \mathbf{f}(1,\,\mathbf{u}_{t^{(0)}}(\mathbf{s}_1,\,2),\,\dots,\,\mathbf{u}_{t^{(J)}}(\mathbf{s}_1,\,2),\,\mathbf{u}_x(\mathbf{s}_1,\,2),\,\mathbf{u}_y(\mathbf{s}_1,\,2),\,\mathbf{u}_{xx}(\mathbf{s}_1,\,2),\,\dots,\,\boldsymbol{\omega}(\mathbf{s}_1,\,2)) \\ \vdots \\ \mathbf{f}(1,\,\mathbf{u}_{t^{(0)}}(\mathbf{s}_S,\,T),\,\dots,\,\mathbf{u}_{t^{(J)}}(\mathbf{s}_S,\,T),\,\mathbf{u}_x(\mathbf{s}_S,\,T),\,\mathbf{u}_y(\mathbf{s}_S,\,T),\,\mathbf{u}_{xx}(\mathbf{s}_S,\,T),\,\dots,\,\boldsymbol{\omega}(\mathbf{s}_S,\,T)) \end{bmatrix},$$

where $\mathbf{F}$ is a $(ST) \times D$ matrix. We can write the linear system in matrix form

$$\mathbf{U}_{t^{(J)}} = \mathbf{FM}, \tag{4}$$

whereby identifying the terms of $\mathbf{M}$ that are non-zero, the dynamic equation is identified.

The derivatives of the system are rarely observed (i.e. only $\mathbf{U}_{t^{(0)}}$ is measured). To obtain derivatives in space and time, numerical techniques are used to approximate the derivatives. There are multiple methods to approximate derivatives numerically, and the choice of approximation has the potential to impact the discovered equation (de Silva *et al.*, 2020). Originally, a finite difference approach was suggested, but this approach is sensitive to white noise (Chartrand, 2011). When measurement error is present, data are either smoothed *a priori*, and then derivatives are computed, or derivatives are computed using total variation regularisation (Chartrand, 2011) or polynomial interpolation (Knowles & Renka, 2012).

Due to the numerical approximation of the derivative and the potential for noise in the observed data, (4) does not hold exactly. Instead,

$$\mathbf{U}_{t^{(J)}} = \mathbf{FM} + \boldsymbol{\eta}, \tag{5}$$

where $\boldsymbol{\eta} \sim \text{MN}_{ST \times N}(\boldsymbol{0}, \mathbf{I}_{ST}, \sigma^2\mathbf{I}_N)$, $\text{MN}_{A \times B}(\mathbf{C}, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$ denotes the matrix normal distribution with mean $\mathbf{C} \in \mathbb{R}^{A \times B}$ and row and column covariance matrices $\boldsymbol{\Sigma}_1 \in \mathbb{R}^{A \times A}$ and $\boldsymbol{\Sigma}_2 \in \mathbb{R}^{B \times B}$, respectively, and σ^2 is the variance associated with the model approximation and the numerical differentiation. To induce sparsity, and thereby identify the relevant terms governing the system, solutions to (5) of the form

$$\mathbf{M} = \underset{\widehat{\mathbf{M}}}{argmin} ||\mathbf{U}_{t^{(J)}} - \mathbf{F}\widehat{\mathbf{M}}||_2^2 + Pen_\theta(\widehat{\mathbf{M}}), \tag{6}$$

are sought, where $Pen_\theta(\hat{\mathbf{M}})$ generically denotes a penalty term based on parameters θ (i.e. $Pen_\theta(\hat{\mathbf{M}}) = \lambda||\hat{\mathbf{M}}||_1$ where $\theta = \lambda$ for the LASSO penalty) and $||\cdot||_p$ denotes the p-norm for either a vector or matrix depending on the argument. Example 2 illustrates how to frame Burgers' equation from Example 1 in a sparse regression framework and the role the feature library plays in the dynamic equation identification.

While not the main focus of this review, boundary conditions serve an important role for dynamic equations. Recently, Shea *et al.* (2021) and Wei (2022) have proposed approaches to incorporate and/or learn boundary and initial conditions for sparse regression problems. Methods for estimating and incorporating boundary and initial conditions in data-driven discovery is an active and on-going area of research.

Example 2: Burgers' equation for sparse regression

We now illustrate how Burgers' equation from Example 1 can be formulated for the sparse regression framework. Assume the data are observed (e.g. Figure 1) and all derivatives can be computed using numerical methods. To frame Burgers' equation (3) as (4), we first define a viable library of potential functions, $\mathbf{f}(\mathbf{s}, t)$. Let $\mathbf{f}(\mathbf{s}, t) = [u(\mathbf{s}, t), u^2(\mathbf{s}, t), u^3(\mathbf{s}, t), u_x(\mathbf{s}, t), u(\mathbf{s}, t)u_x(\mathbf{s}, t), u_{xx}(\mathbf{s}, t)]$, resulting in $D = 6$ (note that in an actual implementation, one would choose a much larger set of functions). We then compute $\mathbf{F}$ by evaluating $\mathbf{f}(\mathbf{s}, t)$ at every space-time location, resulting in $\mathbf{F}$ being a $(ST) \times 6$ matrix. Expanding this for clarity,

$$\begin{bmatrix} u_t(\mathbf{s}_1, 1) \\ u_t(\mathbf{s}_2, 1) \\ \vdots \\ u_t(\mathbf{s}_2, 1) \\ u_t(\mathbf{s}_2, 2) \\ \vdots \\ u_t(\mathbf{s}_S, T) \end{bmatrix} = \begin{bmatrix} u(\mathbf{s}_1, 1) & u^2(\mathbf{s}_1, 1) & u^3(\mathbf{s}_1, 1) & u_x(\mathbf{s}_1, 1) & u(\mathbf{s}_1, 1)u_x(\mathbf{s}_1, 1) & u_{xx}(\mathbf{s}_1, 1) \\ u(\mathbf{s}_2, 1) & u^2(\mathbf{s}_2, 1) & u^3(\mathbf{s}_2, 1) & u_x(\mathbf{s}_2, 1) & u(\mathbf{s}_2, 1)u_x(\mathbf{s}_2, 1) & u_{xx}(\mathbf{s}_2, 1) \\ & & & \vdots & & \\ u(\mathbf{s}_2, 1) & u^2(\mathbf{s}_2, 1) & u^3(\mathbf{s}_2, 1) & u_x(\mathbf{s}_2, 1) & u(\mathbf{s}_2, 1)u_x(\mathbf{s}_2, 1) & u_{xx}(\mathbf{s}_2, 1) \\ u(\mathbf{s}_2, 2) & u^2(\mathbf{s}_2, 2) & u^3(\mathbf{s}_2, 2) & u_x(\mathbf{s}_2, 2) & u(\mathbf{s}_2, 2)u_x(\mathbf{s}_2, 2) & u_{xx}(\mathbf{s}_2, 2) \\ & & & \vdots & & \\ u(\mathbf{s}_S, T) & u^2(\mathbf{s}_S, T) & u^3(\mathbf{s}_S, T) & u_x(\mathbf{s}_S, T) & u(\mathbf{s}_S, T)u_x(\mathbf{s}_S, T) & u_{xx}(\mathbf{s}_S, T) \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \end{bmatrix},$$

or in matrix form, $\mathbf{u}_t = \mathbf{Fm}$, where $\mathbf{m} = [m_1, m_2, m_3, m_4, m_5, m_6]'$. The goal is then to estimate $\mathbf{m} = [0, 0, 0, 0, -1, 0.1]$, where m_5 is the coefficient corresponding to

$u(s, t)u_x(s, t)$ and $m_6 = v$ is the coefficient associated with $u_{xx}(\mathbf{s}, t)$, which is done by finding the solution to (6). Note that in this example, $\mathbf{u}_t$ and $\mathbf{m}$ are vectors because $N = 1$ (i.e. there is only one system). However, if $N > 1$, which could occur with reaction-diffusion models, then $\mathbf{u}_t$ and $\mathbf{m}$ would be matrices (e.g. $\mathbf{U}_t$ and $\mathbf{M}$).

2.1 Deterministic Approaches

The majority of deterministic approaches are composed of three steps—de-noising and differentiation, construction of a feature library and sparse regression. Assuming data have been properly differentiated and a library has been proposed, the deterministic approach seeks solutions of the form (6). The original sparse regression approach to data-driven discovery, *Sparse Identification of Non-linear Dynamics* (SINDy Brunton *et al.*, 2016), uses sequential threshold least-squares (STLS; Algorithm A1) to discover the governing terms for ODEs. While the original paper does not discuss the algorithm in terms of a penalty term, STLS has been shown to be equivalent to the ℓ_0 penalty, $Pen_\theta(\hat{\mathbf{M}}) = ||\hat{\mathbf{M}}||_0$ (Zhang & Schaeffer, 2019), which removes values of $\mathbf{M}$ less than some pre-specified threshold κ. That is, at each iteration of the minimisation procedure, values of $\mathbf{M} < \kappa$ are set to zero and the remaining values of $\mathbf{M}$ are re-estimated. In the original implementation, the algorithm was only iterated 10 times, but a stopping criteria (e.g. change in loss or identified parameters) could be used. In this manner, a sparse solution set is obtained.

In the literature, SINDy is illustrated on a variety of simulated ODE problems with varying amounts of noise. The examples used generally contain many observations (on the order of hundreds of thousands), and it is unclear the impact of noise if a smaller number of observations is considered. In contrast to the symbolic approaches discussed in Section 3, SINDy can be fit quickly. However, a drawback of the approach is the sensitivity to the thresholding parameter and the dependence on the method approximating the derivative.

To extend SINDy to PDEs, Rudy *et al.* (2017) proposed Sequential Threshold Ridge Regression (STRidge, Algorithm A2), a variant to STLS. Due to the correlation present in $\mathbf{F}$ for data pertaining to PDEs, STLS is insufficient at finding a sparse solution set. Instead, STRidge uses the same iterative technique as STLS, where values of $\mathbf{M} < \kappa$ are set to zero at each iteration, but with the addition of the penalty term $Pen_\theta(\hat{\mathbf{M}}) = \lambda||\hat{\mathbf{M}}||_2^2$. Cross-validation is then used to find the optimal values for κ and λ. The effectiveness of STRidge is shown on multiple simulated data sets with varying noise. In comparison with the symbolic counterparts, the algorithm is quick, but still dependent on the method used to approximate the derivative. Example 3 provides additional details on the STRidge algorithm, what the discovered equation may look like, and where to find code to implement the example.

Example 3: Burgers' equation with STRidge

A worked example using Python can be found in the at https://github.com/snagcliffs/PDE-FIND or in the *PySINDy* (de Silva *et al.*, 2020) documentation at https://pysindy.readthedocs.io/en/latest/index.html. The results presented are taken directly from the PDE-FIND GitHub example.

The final identified equation is dependent on various model specification, such as the feature library and the optimiser's (e.g. STRidge) parameters—default values are used this example. The feature library is specified as

$$[1, u, u^2, u^3, u_x, uu_x, u^2u_x, u^3u_x, u_{xx}, uu_{xx}, u^2u_{xx}, u^3u_{xx}, u_{xxx}, uu_{xxx}, u^2u_{xxx}, u^3u_{xxx}],$$

and using the noisy data from Example 1 (e.g. $\zeta = 0.01$) as the observed data, the system is

de-noised and the appropriate derivatives are computed (note that this happens within the optimisation procedure). After running the optimisation procedure, the identified model is

$$u_t = -1.008uu_x + 0.103u_{xx},$$

which correctly identifies the two correct terms with coefficient estimates that are close to the truth. Additionally, the algorithm does not include extraneous terms in its final solution.

STRidge can be adapted to allow for parametric PDEs by grouping terms either spatially or temporally (Rudy *et al.*, 2019). To incorporate parametric PDEs in 4, the coefficients now vary in space or time (i.e. $\mathbf{M}(\mathbf{s})$ or $\mathbf{M}(t)$) and $\mathbf{F}$ is constructed as a block diagonal matrix of the appropriate form (e.g. either in space or time). Similar to the group LASSO (Meier *et al.*, 2008), coefficients are assigned to a collection $g \in \mathcal{G}$ by grouping the same location in space over the entire time domain (e.g. $g \equiv \mathbf{s}$ and $\mathcal{G} \equiv D_S$) or the same time point over the whole spatial domain (e.g. $g \equiv t$ and $\mathcal{G} \equiv D_T$). Within the STRidge algorithm, all coefficients with the same group index are set to zero if $\|\mathbf{M}(g)\|_2 < \kappa$. In this manner, the same dynamics are identified across space and time and only the coefficient estimate is allowed to vary in space or time.

Champion *et al.* (2020) proposed a robust unifying algorithm (Algorithm A3) for the SINDy framework based on sparse relaxed regularised regression (SR3 Zheng *et al.*, 2019). SR3 introduces an auxiliary variable matrix $\mathbf{W}$ within the penalisation term, resulting in $Pen_\theta(\hat{\mathbf{M}}) = \lambda R(\mathbf{W}) + \frac{1}{2v}\|\hat{\mathbf{M}} - \mathbf{W}\|$, where $R(\cdot)$ is another penalisation term (e.g. ℓ_1), and v controls how closely the auxiliary variable $\mathbf{W}$ is to $\mathbf{M}$. The addition of the auxiliary variables provides a more favourable geometric surface to optimise (see Zheng *et al.*, 2019, for a mathematically regorius explination of the SR3). SR3 is shown to be able to handle outliers (a potential issue when numerically differentiating noisy data), accommodate parametric formulations, and allow for physical constraints in the library.

While not discussed in detail, there are other applications and approaches of dynamic equation discovery using sparse methods within the literature. Applying SINDy to stochastic differential equations (Boninsegna *et al.*, 2018) and systems where the dynamics evolve on a different coordinate system (Champion *et al.*, 2019) further increase the SINDy applicability. Instead of using finite differences or total variation regularisation, Schaeffer (2017) use spectral methods to compute spatial derivatives and the Douglas–Rachford algorithm (Combettes & Pesquet, 2011) to find a sparse solution. Further consideration of highly corrupt signals (Tran & Ward, 2017), convergence properties of the SINDy algorithm (Zhang & Schaeffer, 2019), and the choice of de-noising and differentiation methods (Lagergren *et al.*, 2020) have also been studied within the literature. For ease of use, SINDy and related variants are available in the Python package *PySINDy* (de Silva *et al.*, 2020).

2.2 Addressing Uncertainty

Bayesian and bootstrapping approaches have been proposed to quantify uncertainty in the parameters for the sparse regression formulation of data-driven discovery. These approaches seek to quantify the variability in the discovered equation and parameters of (5).

2.2.1 Bayesian approach

A penalised likelihood estimator of the form shown in (6) can be analogously cast as the posterior mode in a Bayesian framework under the prior $p(\mathbf{M}|\theta)$ where $Pen(\hat{\mathbf{M}})_\theta = \log p(\mathbf{M}|\theta)$. That is, (5) can be formulated in the Bayesian framework where priors are put on $\mathbf{M}$ and σ^2. Instead of an optimisation procedure, the Bayesian approach aims to sample from the joint posterior distribution

$$p(\mathbf{M},\, \boldsymbol{\sigma}^2|\mathbf{F},\, \mathbf{U}_{t^{(j)}}) \propto p(\mathbf{U}_{t^{(j)}}|\mathbf{F},\, \mathbf{M},\, \boldsymbol{\sigma}^2)p(\mathbf{M}|\theta)p(\boldsymbol{\sigma}^2|\theta)p(\theta),$$

where $p(\mathbf{U}_{t^{(j)}}|\mathbf{F},\, \mathbf{M},\, \boldsymbol{\sigma}^2)$ is the data likelihood derived from (5), and $p(\mathbf{M}|\theta)$ and $p(\boldsymbol{\sigma}^2|\theta)$ are prior distributions for $\mathbf{M}$ and $\boldsymbol{\sigma}^2$, respectively, and $p(\theta)$ is the prior distribution for the hyperparameters [or penalisation parameters in (6)]. To enforce a sparse solution set in a Bayesian framework, a regularisation prior is placed on the parameter of interest, in this case $\mathbf{M}$. Further discussion comparing the sparse regression approach to a Bayesian formulation of the problem is explored by Niven *et al.* (2020).

Using the Bayesian framework in an algorithmic setting, Zhang & Lin (2018) proposed using the priors $p(m_d|\alpha_d) = \mathrm{N}(0,\, \alpha_d^{-1}), p(\sigma^2) = \mathrm{IG}(a_s,\, b_s)$, and $p(\alpha_d^{-1}) = \mathrm{IG}(a_a,\, b_a)$, where $\mathrm{IG}(a,\, b)$ is the inverse Gamma distribution with shape a and scale b, and $d = 1,\, \ldots,\, D$. They estimate the parameters using a threshold sparse Bayesian regression algorithm, which maximises the marginal likelihood instead of sampling from the full conditional distributions as in Markov chain Monte Carlo. Their algorithm uses a hard thresholding parameter, similar to the deterministic sparse regression approaches, where at each iteration, values of the posterior $\mathbf{M}(i,\, j) < \kappa$ are set to zero. Their procedure assigns what they term 'error bars' to their parameter estimates based on the ratio of the estimate for the posterior variance to the estimate for the posterior mean squared. Zhang & Lin (2018) consider many of the same simulated ODEs and PDEs used to illustrate the deterministic approaches and provide error bars to the parameter estimates for these systems with varying amounts of noise. As an interesting application, Zanna & Bolton (2020) use this framework to discover unknown equations in ocean mesoscale closures.

Hirsh *et al.* (2021) explore the use of two common Bayesian selection priors on system discovery and uncertainty quantification—the continuous spike and slab (i.e. stochastic search variable selection (SSVS); Mitchell & Beauchamp, 1988; George *et al.*, 1993; George & McCulloch, 1997), and the regularised horseshoe (Carvalho *et al.*, 2010; Piironen & Vehtari, 2017)—calling the approach *uncertainty quantification SINDy* (UQ-SINDy). Their choice of priors are distinct in that SSVS is a mixture of two continuous mean zero Gaussian distributions and the horseshoe is part of the global-local shrinkage prior family. For the SSVS prior, variables that are not to be included in the model are sampled from a mean zero Gaussian distribution with a small variance, rendering their effect on inference negligible, and variables that are to be included are sampled from a mean zero Gaussian distribution with a larger variance. The posterior inclusion probability for a variable is the number of times it was sampled from the Gaussian with a large variance over the total number of samples. In contrast, the horseshoe prior has a hyper-prior performing global shrinkage on all variables in conjunction with individual hyper-priors performing individual shrinkage. To determine the probability a variable is included under the regularised horseshoe, the ratio of each element of the estimate of $\mathbf{M}$ with no prior and with the horseshoe prior is computed, providing *pseudo-inclusion probabilities* (i.e. not necessarily bounded by 0 and 1). Using both of these priors, Hirsh *et al.* (2021) provide inclusion probabilities for multiple simulated ODE systems with varying amounts of noise and to the classic hare-lynx population data set (Elton & Nicholson, 1942).

However, these two approaches are limited in that the uncertainty being quantified is the uncertainty in the numerical approximation of the system (i.e. the numerical differentiation and de-noising). That is, because the approximated derivative is, in fact, a single realisation of the true derivative (which is unknown), the uncertainty estimates recovered by this approach are biased towards this single approximation of the derivative. A more complete treatment of the problem would be to consider the derivative as a random process and account for uncertainty in the random process.

Yang *et al.* (2020) proposed the use of Bayesian differentiable programming as a method by which to discover the dynamics and account for measurement uncertainty when estimating parameters. Generally speaking, Bayesian differentiable programming uses a numerical solver (e.g. Runge–Kutta) to predict the state at a new time, and the loss between the predicted data and observed data is used to estimate parameters. More precisely, letting $M_\theta(\mathbf{u}(t))$ be the output of a numerical solver at time t, Bayesian differentiable programming aims to minimise $\sum \|\mathbf{u}(t+\Delta t) - M_\theta(\mathbf{u}(t))\|^2$, where Δt does not need to be uniformly spaced. The parameters are estimated using Hamiltonian Monte Carlo (HMC) and differentiable programming is used to compute gradients within the HMC algorithm. By directly relating the observed data to the dynamics, measurement uncertainty is accounted for in the estimation procedure, providing a more thorough statistical treatment to the data-driven discovery problem. The approach is illustrated on multiple simulated ODE systems with varying amounts of noise.

Bhouri & Perdikaris (2022) extend the idea of Bayesian differential programming for data-driven discovery by using Gaussian process priors on the state variables to model temporal correlations and use NeuralODEs (Chen *et al.*, 2018) to perform numerical integration. Additionally, they use the 'Finnish Horseshoe prior' to impose variable shrinkage on the learned library, similar to Hirsh *et al.* (2021). The model parameters are estimated using HMC and the No-U-Turn-Sampler (NUTS Hoffman & Gelman, 2014) algorithm to automatically calibrate model parameters. Similar to Yang *et al.* (2020), the approach is illustrated on multiple simulated ODE systems in addition to human motion capture data.

2.2.2 *Bootstrap approach*

Fasel *et al.* (2022) proposed two methods of bootstrapping (4)—either sampling rows of the data (i.e. space-time sampling) or sampling library terms (i.e. columns of $\mathbf{F}$). The first approach samples rows of the data with replacement and uses STRidge to estimate the parameters in the model q times. In the second approach, the columns of $\mathbf{F}$ are sampled *without* replacement to create q data sets, and again STRidge is used to estimate parameters. For both methods, the q models are then averaged and coefficients with an inclusion probability below a pre-specified value are set to zero. Uncertainty is quantified by the inclusion probability and the distribution of values obtained from the q different estimates. However, as with Hirsh *et al.* (2021), the uncertainty associated with the observed data is not considered and the numerical approximation to the derivative is treated as an observation of the derivative, limiting uncertainty quantification. The method is illustrated on multiple simulated ODE and PDE systems with varying levels noise and applied to the classic hare-lynx population data set.

3 Symbolic Regression

Symbolic regression is a type of regression that searches over mathematical expressions (e.g. $+, -, \times$) to find the optimal model for a given data set (Wang *et al.*, 2019). This approach differs from classical regression where the model structure is fixed and a set of parameters are estimated. One of the main challenges underlying symbolic regression is that there are an

infinite number of combinations of expressions that can be used to fit any particular data set. *Genetic programming* is used to efficiently search over the possible model structures (Willis, 1997; Koza *et al.*, 1993; Koza, 1994) and regression techniques are used to determine coefficient values given the model structure. Genetic programming follows Darwin's theory of evolution, selecting the 'fittest' solution that is the product of generations of evolution (i.e. iterating through an algorithm). We give a brief overview of genetic programming and its roll in symbolic regression and subsequently data-driven discovery of dynamics. For a more detailed overview of genetic programming, see Minnebo and Stijven (2011, chapter 4) and Garg & Tai (2012).

Genetic programming relies on a pre-defined *function set* of mathematical expressions. For symbolic regression, the function set typically consists of basic mathematical expressions such as addition, multiplication, and trigonometric terms (see Nicolau & Agapitos, 2018, for details on function set choice). Possible model solutions are constructed using a combination of functions from the function set and encoded in a tree structure (Figure 2). Within the tree, the mathematical expressions are the decision nodes and input data passed into the mathematical expression are the terminal nodes. To make the searchable space smaller, the maximum node size of the tree can be specified. A *population* of potential solutions is composed of *individual* potential solutions. The ability of an individual to properly represent data is determined based on the *fitness function*, which is analogous to an objective or loss function in statistics. Individuals can then *reproduce* to create a copy of themselves, *crossover* with another individual, or *mutate* themselves. Crossover is where two individuals swap sub-trees (i.e. a decision node is randomly selected from each tree and exchanged) to produce two new individuals, which is equivalent to parents producing offspring with shared genetics. Mutation is where an individuals decision node is randomly changed (e.g. plus to multiplication or plus to a variable), which is akin to a genetic mutation.

The general algorithm for genetic programming proposes an initial population, assesses the fitness of each individual, and then generates the next population based on the fittest individuals of the current population (Algorithm A4). Taking the fitness function to be based on regression where the goal is to minimise mean squared error results in symbolic regression (Schmidt & Lipson, 2009). This basic genetic programming/symbolic regression method has generated multiple extensions (Icke & Bongard, 2013; Chen *et al.*, 2017; Amir Haeri *et al.*, 2017; Jin *et al.*, 2019) and incurred extensive discussion (Korns, 2014; Nicolau & Agapitos, 2018; Ahvanooey *et al.*, 2019).

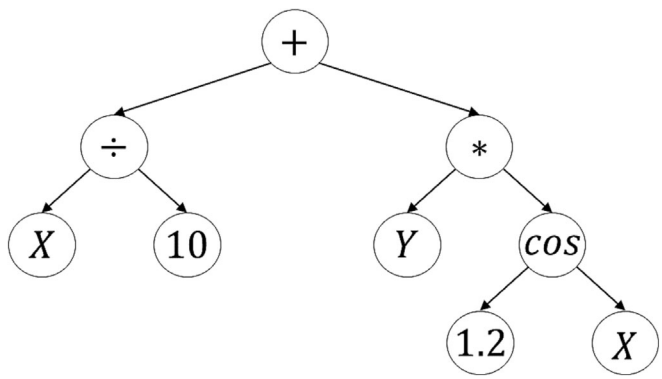

FIGURE 2. *Symbolic representation of* $f(X, Y) = \frac{X}{10} + Y * 1.2\cos(X)$

Within the context of data-driven discovery, symbolic regression attempts to find the evolution function $\mathcal{M}(\cdot)$ in (1) or (2). The difficulty is relating a proposed choice for $M(\cdot)$ that is generated within the genetic algorithm to derivatives of the observed data. Specifically, because derivatives of the system are unknown, either the fitness function needs to account for the derivative or the derivatives must be obtained in order to use a traditional fitness function.

Bongard & Lipson (2007) were the first to apply symbolic regression to data-driven discovery of dynamic systems, focusing on the discovery of ODEs. To use symbolic regression to discover dynamic models with potentially non-linear interactions of multiple variables, the authors introduced partitioning, automated probing, and snipping within a symbolic regression algorithm. Partitioning considers each variable in a system separately, even though they may be coupled, substantially reducing the search space of possible equations. With partitioning, a candidate equation for a single variable is integrated with the others assumed fixed. Automated probing is where initial conditions used for temporal integration of the dynamic equation of the system are found. Last, snipping is the process of simplifying and restructuring models by replacing sub-expressions (sub-trees) in the generated population with a constant. Using these three components, each variable in the system is integrated forward in time to produce a 'test' based on the initial condition and compared to the observed data. The fitness of each potential solution is computed based on the average absolute difference between the observed data and the test. The approach is illustrated on simulated data and on two real-world examples—the classic hare-lynx system and data they collect from a pendulum. Although effective, their method is sensitive to noise in the data and has the same demanding computational requirements as other symbolic regression algorithms.

Schmidt & Lipson (2009) adopt a different approach to data-driven discovery with symbolic regression. They search over a function space constrained by a loss function dependent on partial derivatives computed from the symbolic functions and from the data. Specifically, given two variables observed over time, $x(t)$ and $y(t)$ (i.e. $\mathbf{u}(t) = [x(t), y(t)]'$), the numerical estimate of the partial derivatives between the pair is approximated as $\frac{\Delta x}{\Delta y} \approx \frac{dx}{dt} \Big/ \frac{dy}{dt}$, where $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are estimated using local polynomial fits (Thompson & Wallace, 1998). From a potential solution function (i.e. generated in the genetic algorithm), the partial derivatives can be computed using symbolic differentiation to get $\frac{\delta x}{\delta y}$ (i.e. from the symbolic function). To determine how well the potential function expresses the data, the mean log error between the approximated and symbolic partial derivatives,

$$-\frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \left|\frac{\Delta x}{\Delta y} - \frac{\delta x}{\delta y}\right|\right),$$

is used as the fitness function. This approach can be extended to systems with more than two variables by looking at pairs of the variables in the system (see supplementary material of Schmidt & Lipson, 2009, for details). In this manner, they assign a fitness to each proposed individual based on how well the derivative of the system relates to the derivative of the data, resulting in data-driven discovery using symbolic regression. However, noise can be impactful because the derivatives from the observed data are approximated numerically. To accommodate measurement uncertainty, Schmidt & Lipson (2009) use Loess smoothing (Cleveland & Devlin, 1988) prior to fitting to remove the high frequency noise. Their approach is illustrated using simulated data and data collected by motion tracked cameras, showing an ability to recover the equations on complex, real-world problems. However, similar to Bongard &

Lipson (2007), the method is also computationally cumbersome with some examples reportedly taking days to converge.

Motivated by symbolic and sparse regression, Maslyaev *et al.* (2019) embed sparse regression within the coefficient estimation step in a symbolic regression algorithm to discover the governing equations of PDEs. In their approach, derivatives of the data are computed *a priori* using finite difference (in the same manner as sparse regression discussed in Section 2) and used as the response in symbolic regression. Within the symbolic regression algorithm, after a population has been proposed, sparse regression using an ℓ_1 penalty is employed, the fitness of each individual in the population is assessed, and mutation, crossover, and replication are performed in the usual manner. Because derivatives are computed before the estimation procedure, they are able to be incorporated into the function set. This allows for the discovered equations to contain spatial derivatives. The approach is tested on multiple simulated PDEs with varying amounts of noise. However, the robustness to noise is dependent on the numerical method used to approximate the derivative, and it is unclear how this impacts model results. Additionally, while specifics are not given, the approach is computationally cumbersome, owing in part to the symbolic regression.

4 Deep Models

Deep modeling has been considered for data-driven dynamic discovery in two different ways —approximating dynamics and learning dynamics. Approximating dynamics using deep models provides a computationally cheap method to generate data from complex systems while still preserving physical aspects of the system (i.e. emulation). While this review is concerned with the discovery of the governing equations and refers to 'data-driven discovery' as the discovery of the *functional form* of the governing system, deep models approximating the dynamics are an important part of the literature and we devote a section to them. Deep models coinciding with our definition of data-driven discovery have also been developed. There are multiple approaches by which dynamics can be approximated and subsequently learned, which we also discuss.

4.1 Approximating Dynamics With Deep Models

One method of approximating dynamics considers a so-called *physics-informed neural network* (PINN; Raissi *et al.*, 2017a, 2017b; Raissi, 2018; Raissi *et al.*, 2019; Raissi *et al.*, 2020). PINNs are applicable to both continuous and discrete time models, yet we discuss only the continuous version. Define

$$\mathbf{g}(\mathbf{s},\, t) = \mathbf{u}_{t^{(J)}}(\mathbf{s},\, t) + M(\mathbf{u}(\mathbf{s},\, t),\, \mathbf{u}_x(\mathbf{s},\, t),\, \ldots),$$

where the form of M is assumed known. Approximating $\mathbf{u}(\mathbf{s},\, t)$ with a neural network results in the PINN $\mathbf{g}(\mathbf{s},\, t)$, where the derivatives associated with the PINN are computed using automatic differentiation. The neural network is trained using the loss function $MSE = MSE_u + MSE_g$ where MSE_u is the mean squared error of the neural network approximating $\mathbf{u}(\mathbf{s},\, t)$ and $MSE_g = \frac{1}{N_g}\sum_{i=1}^{N_g} \|\mathbf{g}(\mathbf{s}_i,\, t_i)\|^2$ is the mean squared error associated with the structure imposed by $\mathbf{g}(\cdot)$. In this manner, the neural network obeys the physical constraints imposed by $\mathbf{g}(\cdot)$.

Neural networks have also been used to approximate the evolution operator M using a residual network (ResNet). Reframing the problem according to the Euler approximation $\mathbf{u}(t+\Delta t) \approx \mathbf{u}(t)+\Delta t M(\mathbf{u}(t))$, where $\mathbf{u}(t)$ represents the vector of the process for all locations at time t, the goal is to find a suitable approximation for $M()$, thereby approximating the dynamics. In contrast to PINN, physics are not incorporated into the NN and the structure of the NN is dependent completely on the data. Applying the problem to ODEs, Qin *et al.* (2019) show how a recurrent ResNet with uniform time steps (i.e. uniform Δt) and a recursive ResNet with adaptive time steps can be used to approximate dynamics. This approach is further extended to PDEs (Wu & Xiu, 2020), where the evolution operator is first approximated by basis functions and coefficients, and a ResNet is fit to the basis coefficients.

While not described in detail, there are other approaches to approximating dynamic equation using deep models. Physics-informed candidate functions can be used with numerical integration in an objective function to restrict the temporal evolution of a NN (Sun *et al.*, 2019). NN have also been used to approximate parametric PDEs (Khoo *et al.*, 2021), represent molecular dynamics (Mardt *et al.*, 2018), learn turbulent dynamical systems (Qi & Harlim, 2022) and approximate ODEs with time-varying measurement data (Wu & Xiu, 2019). Using deep models to learn the time-stepping scheme of dynamic equations, Rudy *et al.* (2019) and Liu *et al.* (2022) independently show how the approximation of the dynamics are improved when dynamic time-stepping is accounted for in the estimation proceedure. Wikle & Zammit-Mangion (2022) review methods using deep learning for spatial processes and for approximating spatio-temporal DEs.

4.2 Discovering Dynamics With Deep Models

Deep modeling using neural networks (NNs) has become increasingly popular in recent years due to NNs being a universal approximator (Hornik *et al.*, 1989). Additionally, computing derivatives of NNs is possible through automatic differentiation (e.g. using PyTorch Paszke *et al.*, 2017). Assuming a surface can be approximated using a NN, derivatives of the surface in space or time or both are obtainable. This approach, where derivatives are computed using NN, is used in many of the deep model approaches to data-driven discovery.

4.2.1 Deep models with sparse regression

A common issue with data-driven discovery in the 'classical' sparse regression approach is the sensitivity to noise when approximating derivatives numerically. To address this issue, Both *et al.* (2021) proposed using a NN to approximate the system, and then perform sparse regression within the NN. For example, let $\hat{\mathbf{U}}$ be the output of a NN and construct $\mathbf{F}$ in (4) using $\hat{\mathbf{U}}$ and derivatives computed from $\hat{\mathbf{U}}$ via automatic differentiation. The NN is trained using the loss function

$$\mathcal{L} = \frac{1}{ST}\sum |\mathbf{U} - \hat{\mathbf{U}}|^2 + \frac{1}{ST}\sum |\mathbf{F}\mathbf{M} - \hat{\mathbf{U}}_{t^{(J)}}|^2 + \lambda \sum |\mathbf{M}|.$$

After training the NN and estimating parameters, most terms of $\mathbf{M}$ are still non-zero (but very close to zero), and a thresholding is performed to obtain the final sparse representation. Through this formulation of the problem whereby derivatives are obtained from a NN, Both *et al.* (2021) show their ability to recover highly corrupt signals from traditional PDE systems and apply their approach to a real-world electrophoresis experiment.

4.2.2 Deep models with symbolic regression

Using symbolic regression with a neural network for data-driven discovery has gained popularity in recent years. In a series of papers, Xu et al. (2019-2021) construct a deep learning genetic algorithm for the discovery of parametric PDEs (DLGA-PDE) with sparse and noisy data. DLGA-PDE first trains a NN that is used to compute derivatives and generate meta-data (global and local data), thereby producing a complete de-noised reconstruction of the surface (i.e. noisy sparse data are handled through the NN). Using the *local* metadata produced by the NN, a genetic algorithm learns the general form of the PDE and identifies which parameters vary spatially or temporally. At this step, the coefficients may be incorrect or missrepresent the system because the global structure of the data is not accounted for. To correct the coefficient estimates, a second NN is trained using the discovered structure of the PDE and the *global* metadata. Last, a genetic algorithm is used to discover the general form of the varying coefficients.

One method of implementing symbolic regression within a deep model is to allow the activation functions to be composed of the function set instead of classic activation functions (e.g. sigmoid or ReLU; Martius & Lampert, 2016; Sahoo *et al.*, 2018; Kim *et al.*, 2021). Motivated by this idea, Long *et al.* (2019) proposed a symbolic regression NN, *SymNet*. Similar to a typical NN, the ℓ-th layer of *SymNet* is

$$(c_\ell,\, d_\ell)' = \mathbf{W}^\ell \left[\mathbf{f}^0, f^1, \ldots, f^{\ell-1}\right]' + \mathbf{b}^\ell$$
$$f^\ell = c_\ell \times d_\ell, \quad \ell = 1, \ldots, k$$

where the length-m vector $\mathbf{f}^0$ is the function set that contains partial derivatives (e.g. $\mathbf{f}^0 = [u, u_x, u_y, \ldots]$), $\mathbf{W}^\ell \in \mathbb{R}^{2 \times (m+\ell-1)}$, and $\mathbf{b}^\ell$ is a length-2 vector. In this manner, each subsequent layer adds a dimension to the activation function based on the previous layer, allowing the construction of complex functions. To relate the network to a non-linear function, the final layer is $\mathbf{f}^{k+1} = \mathbf{w}^{k+1}\left[\mathbf{f}^0, f^k\right]' + \mathbf{b}^{k+1}$ where $\mathbf{w}^{k+1}$ is a length-$m+k$ vector. Following Long *et al.* (2017), spatial derivatives are computed using finite difference via convolution operators. To model the time dependence of PDEs, they employ the forward Euler approximation, termed a δt-block, as

$$\mathbf{u}(t+\delta t) \approx \mathbf{u}(t) + \delta t \cdot SymNet_m^k(u, u_x, u_y, \ldots),$$

δt is the temporal discritisation, and $SymNet_m^k(u, u_x, u_y, \ldots)$ has k hidden layers (i.e. $\ell = 0, \ldots, k$) and m variables (i.e. number of arguments $u, u_x, u_y, \ldots$). To facilitate long-term predictions, they train multiple δt-blocks as a group so the system has long-term accuracy.

Distinct from the previous two approaches, Atkinson *et al.* (2019) incorporate differential operators into the function set of a genetic algorithm. They train the NN on the observed data and supply the NN to a genetic algorithm where the function set contains typical operators (e.g. addition, multiplication) and differential operators. The differential operators are computed from the NN using PyTorch (Paszke *et al.*, 2017), enabling the inclusion of derivatives in the search space of the genetic algorithm.

5 Physical Statistical Models

To account for measurement uncertainty and missing data when modeling complex non-linear systems, dynamic equations parameterised by ordinary and partial differential equations have been incorporated into Bayesian hierarchical models (BHMs). While there are

various methods by which to model dynamic equations in a probabilistic framework, we focus on physical statistical models (PSM; Berliner, 1996; Royle *et al.*, 1999; Wikle *et al.*, 2001) due to the similarities with data-driven discovery that will soon become apparent. Broadly, PSM are a class of BHMs where scientific knowledge about a process is known and incorporated into the model structure.

PSMs are generally composed of three modeling stages—data, process, and parameter models—where dynamics are modeled in the process model and the observed data are modeled conditioned on the latent dynamics. That is, the observed data are considered to be a noisy *realisation* of the 'true' latent dynamic process. This formulation results in the data being described conditionally given the process model, simplifying the dependence structure in the data model and enabling complex structure to be captured in the process stage. The evolution of the latent dynamic process is then parameterised by a dynamic equation, incorporating physical dynamics into the modeling framework.

Consider the $R(t) \times 1$ observed data vectors $\mathbf{v}(t) \equiv [v(\mathbf{r}_1, t), \ldots, v(\mathbf{r}_{R(t)}, t)]'$ and $\{v(\mathbf{r}, t) : \mathbf{r} \in D_s, t \in D_t\}$ where $\mathbf{r} \in \{\mathbf{r}_1, \ldots, \mathbf{r}_{R(t)}\} \subset D_s$ is a discrete location in the spatial domain D_s, and $t \in \{1, \ldots, T\} \subset D_t$ is the realisation of the system at discrete times in a temporal window D_t. Assume we are interested in the latent 'true' dynamic process $\{u(\mathbf{s}, t) : \mathbf{s} \in D_s, t \in D_t\}$ where $\mathbf{u}(t) \equiv [u(\mathbf{s}_1, t), \ldots, u(\mathbf{s}_S, t)]'$ is a length S vector. It is common that the observation locations do not coincide with the process (e.g. due to missing data or mismatch in resolution). In the case of missing observations, the observed data are mapped to the latent process using an incidence matrix $\mathbf{H}(t)$, which is a matrix of zeros except for a single one in each row corresponding the observation associated with a process location (see chapter 7 of Cressie & Wikle, 2011, for examples of $\mathbf{H}(t)$). The general data model for time t is

$$\mathbf{v}(t) = \mathbf{H}(t)\mathbf{u}(t) + \boldsymbol{\epsilon}(t),$$

where $\mathbf{H}(t) \in \mathbb{R}^{L(t) \times N}$ and uncertainty in the observations of the process are captured by $\boldsymbol{\epsilon}(t) \overset{\text{indep.}}{\sim} \mathrm{N}_{L(t)}(\boldsymbol{\theta}, \boldsymbol{\Sigma}_V(t))$ where $\boldsymbol{\Sigma}_V(t)$ is the covariance matrix.

The dynamic process is characterised through the specification of the evolution of $\mathbf{u}(t)$ over time. For example, the process model, which specifies this evolution under a first-order Markov assumption, is given as

$$\mathbf{u}(t) = \mathcal{M}(\mathbf{u}(t - 1), \boldsymbol{\theta}) + \boldsymbol{\eta}(t), \tag{7}$$

where $\mathcal{M}(\cdot)$ is a (non)linear function relating a previous space-time location (or multiple locations) to the next, $\boldsymbol{\theta}$ are parameters associated with $\mathcal{M}$, and $\boldsymbol{\eta}(t) \overset{\text{i.i.d.}}{\sim} \mathrm{N}_N(\boldsymbol{\theta}, \boldsymbol{\Sigma}_U)$ is a mean zero Gaussian process with variance/covariance matrix $\boldsymbol{\Sigma}_U$. While not discussed here, the error term $\boldsymbol{\eta}(t)$ can be considered multiplicative (see chapter 7 of Cressie & Wikle, 2011, for more detail).

Physical dynamics are encoded through the parameterisation of $\mathcal{M}$. We consider physical dynamic parameterisations (i.e. ODEs and PDEs), but a general autoregressive structure for $\mathcal{M}$ (i.e. not parameterised with differential equations) can also be considered. Consider the general PDE

$$\mathbf{u}_t(t) = \mathcal{M}(\mathbf{u}(t), \boldsymbol{\theta}),$$

analogous to the motivating PDE (1), which can be approximated using finite differences

$$\mathbf{u}(t) = \mathbf{u}(t\ -\ 1)+\Delta t\mathcal{M}(\mathbf{u}(t\ -\ 1),\boldsymbol{\theta}),$$

where Δt is the difference in time between time t and $t\ -\ 1$ and $\boldsymbol{\theta}$ are parameters associated with the PDE. Because the finite difference approximation can be written as a linear system, we can write

$$\mathbf{u}(t) = \mathbf{M}\mathbf{u}(t\ -\ 1), \tag{8}$$

where $\mathbf{M}$ is a sparse matrix derived from the finite difference scheme. Replacing (7) with (8), the process model parameterised by a linear finite difference equation is

$$\mathbf{u}(t) = \mathbf{M}\mathbf{u}(t\ -\ 1)+\boldsymbol{\eta}(t),$$

where $\boldsymbol{\eta}(t)$ may now account for approximation error due to the finite difference approximation.

As a clarifying example, assume a spatio-temporal process $u(x,\ t)$ in one-dimensional space $0 \le x \le L$ and time t. Assume the process is approximated by the diffusion equation $u_t(x,\ t) = bu_{xx}(x,\ t)$ where b is a diffusion constant and the boundary conditions $u(0,\ t) = u_0$ and $u(L,\ t) = u_L$ and initial condition $\{u(x,\ 0): 0 \le x \le L\}$ are known. Using numerical analysis, the time derivative can be approximated using the forward difference

$$u_t(x,\ t) \approx \frac{u(x,\ t+\Delta t) - u(x,\ t)}{\Delta t},$$

and the spatial derivative can be approximated by the central difference

$$u_{xx}(x,\ t) \approx \frac{u(x+\Delta x,\ t) - 2u(x,\ t)+u(x\ -\ \Delta x,\ t)}{\Delta x^2}.$$

Using the finite difference approximation, we can reformulate the diffusion equation as

$$u(x,\ t+\Delta t) \approx u(x,\ t)+\frac{b\Delta t}{\Delta x^2}(u(x+\Delta x,\ t) - 2u(x,\ t)+u(x\ -\ \Delta x,\ t)).$$

Assuming three internal spatial locations, $x_1,\ x_2,\ x_3$ and boundary locations $x_0,\ x_L$, let $\mathbf{u}(t) = [u(x_1,\ t),\ u(x_2,\ t),\ u(x_3,\ t)]'$ and $\mathbf{u}^b(t) = [u(x_0,\ t),\ u(x_L,\ t)]'$. Then,

$$\mathbf{u}(t+\Delta t) \approx \begin{bmatrix} 1\ -\ \frac{2b\Delta t}{\Delta x^2} & \frac{b\Delta t}{\Delta x^2} & 0 \\ \frac{b\Delta t}{\Delta x^2} & 1\ -\ \frac{2b\Delta t}{\Delta x^2} & \frac{b\Delta t}{\Delta x^2} \\ 0 & \frac{b\Delta t}{\Delta x^2} & 1\ -\ \frac{2b\Delta t}{\Delta x^2} \end{bmatrix}\mathbf{u}(t)+\begin{bmatrix} \frac{b\Delta t}{\Delta x^2} & 0 \\ 0 & 0 \\ 0 & \frac{b\Delta t}{\Delta x^2} \end{bmatrix}\mathbf{u}^b(t),$$

which can be written more compactly as $\mathbf{u}(t+\Delta t) \approx \mathbf{M}\mathbf{u}(t)+\mathbf{M}^b\mathbf{u}^b(t)$. Thus, the PDE dynamics have been 'encoded' into the structure of the transition operator, $\mathbf{M}$. In most PSM implementations, the (banded) structure of $\mathbf{M}$ is retained, but the specific elements are estimated from the data, rather than given by the finite difference representation. This adds flexibility and explicitly assumes that the PDE is not an exact representation of the data. Note that other PDE representations, such as finite element, or spectral, can be used to motivate such models.

This simple example can be made more complex by considering a parametric diffusion equation (i.e. replacing $\mathbf{M}$ with $\mathbf{M}(\boldsymbol{\theta})$) or by putting priors on the boundary and/or the initial conditions (see Cressie & Wikle, 2011, for details). Additionally, there are certain numerical conditions that need to be satisfied to guarantee numerical stability from the approximation, which can vary based on the system and approximation scheme considered (e.g. see CFL condition in Higham *et al.*, 2016). There are cases where the numerical constraints, or more specifically the parameters dictating the discritisation scale(s), can be modeled as statistical parameters through *non-constrained PSMs* (Berliner *et al.*, 2003). For a more complete overview of PSMs and possible parameterisations, see Berliner (2003), Cressie & Wikle (2011), Kuhnert (2017), and references therein.

PSMs have been used to study a variety of real-world systems. PSMs parameterised using shallow-water equations (Wikle, 2003) and the Rayleigh friction equation (Milliff *et al.*, 2011) have been used to study ocean surface winds. Using a parametric diffusion equation (Wikle, 2003) and parametric reaction-diffusion equation (Hooten & Wikle, 2008), PSMs have modeled the spread of invasive avian species. PSMs can be grouped into a larger category of models called general quadratic non-linear model (GQN; Wikle & Hooten, 2010; Wikle & Holan, 2011; Gladish & Wikle, 2014), which accommodate multiple classes of scientific-based parameterisation such as PDEs and integro-difference equations.

5.1 General Quadratic Non-Linear Models

General quadratic non-linear models provide a nice generalisation to the PSM framework and, as discussed in the subsequent section, provide an interesting link between data-driven discovery methods and PSMs. The general GQN model is

$$u(\mathbf{s}_i, t) = \sum_{j=1}^{S} a_{ij} u(\mathbf{s}_j, t-1) + \sum_{k=1}^{S}\sum_{l=1}^{S} b_{i,kl} u(\mathbf{s}_k, t-1) g(u(\mathbf{s}_l, t-1); \boldsymbol{\theta}) + \eta(\mathbf{s}_i, t), \quad (9)$$

for $i = 1, \ldots, S$, where a_{ij} are linear evolution parameters, $b_{i,kl}$ are non-linear evolution parameters, $g()$ is a transformation function of $u(t-1)$ dependent on parameters $\boldsymbol{\theta}$, and $\eta(\mathbf{s}_i, t)$ is an error process. This is motivated by the fact that many real-world mechanistic processes have been described by PDEs that have quadratic (non-linear) interactions, often where the interaction of system components consists of the multiplication of one component by a transformation of another (see Wikle & Hooten, 2010, for details).

Equation (9) can be condensed in matrix form as

$$\mathbf{u}(t) = \mathbf{A}\mathbf{u}(t-1) + (\mathbf{I}_S \otimes g(\mathbf{u}(t-1); \boldsymbol{\theta})')\mathbf{B}\mathbf{u}(t-1) + \boldsymbol{\eta}(t), \quad (10)$$

where $\mathbf{A}$ and $\mathbf{B}$ are matrices constructed from a_{ij} and $b_{i,kl}$, respectively, and $\mathbf{I}_S$ is a size S identity matrix (see Wikle & Hooten, 2010, for specific details). From (10), we see that letting

$$\mathbf{M}(\mathbf{u}(t-1), \boldsymbol{\theta}) = \mathbf{A}\mathbf{u}(t-1) + (\mathbf{I}_S \otimes g(\mathbf{v}(t-1); \boldsymbol{\theta})')\mathbf{B}\mathbf{u}(t-1)$$

recovers the PSM model. The GQN framework is very flexible, due in part to the over-parameterisation of the model from all possible quadratic interactions. To constrain the parameter space, thereby learning which dynamic interactions are important, either physics-informed priors or strong shrinkage priors are used. For examples on what these constraints may be and their underlying physical motivation, see Wikle & Hooten (2010).

5.2 Relation to Data-Driven Discovery

While unexplored in the literature, there is a strong connection between PSMs (particularly, the more general GQNs) and data-driven discovery. Formulating a BHM where the latent process evolves according to the generic PDE (1), the two-stage data-process model for location $\mathbf{s}$ and time t is

$$\begin{aligned}\mathbf{v}(\mathbf{s}, t) &= \mathbf{H}(\mathbf{s}, t)\mathbf{u}(\mathbf{s}, t)+\boldsymbol{\epsilon}(\mathbf{s}, t)\\ \mathbf{u}_{t^{(J)}}(\mathbf{s}, t) &= \mathcal{M}(\mathbf{u}(\mathbf{s}, t), \mathbf{u}_x(\mathbf{s}, t), \ldots)+\boldsymbol{\eta}(\mathbf{s}, t),\end{aligned}$$

where $\boldsymbol{\epsilon}(\mathbf{s}, t) \sim \mathrm{N}_{L(t)}(\boldsymbol{0}, \boldsymbol{\Sigma}_V(\mathbf{s}, t))$ is the measurement error process with $\boldsymbol{\Sigma}_V(\mathbf{s}, t)$ a variance/covariance matrix, $\boldsymbol{\eta}(\mathbf{s}, t) \sim \mathrm{N}_N(\boldsymbol{0}, \boldsymbol{\Sigma}_U(\mathbf{s}, t))$ the process model error with $\boldsymbol{\Sigma}_U(\mathbf{s}, t)$ a variance/covariance matrix. However, as discussed in Section 5, PSMs rely on $\mathcal{M}$ to be parameterised according to known dynamics. Instead, borrowing the notion of a feature library from the sparse regression approach to data-driven discovery, linearising the process model results in a matrix of coefficients $\mathbf{M}$ and a feature library $\mathbf{f}(\cdot)$. Given the feature library, the goal is to find the correct values of $\mathbf{M}$ (as in sparse regression). In the case of GQN, we rarely need the whole set of quadratic interactions, so the 'discovery' is selecting which quadratic components are needed to describe the data.

As an example, consider two approaches that can be used to incorporate dynamic discovery into PSMs—employing a finite difference scheme or using (5) for the process model—each of which have their own pros and cons. The finite difference approach results in the same model as in Section 5, where

$$\begin{aligned}\mathbf{v}(\mathbf{s}, t) &= \mathbf{H}(\mathbf{s}, t)\mathbf{u}(\mathbf{s}, t)+\boldsymbol{\epsilon}(\mathbf{s}, t),\\ \mathbf{u}(\mathbf{s}, t) &= \mathbf{M}\mathbf{f}(\mathbf{u}(\mathbf{s}, t-1), \mathbf{u}_x(\mathbf{s}, t-1), \ldots)+\boldsymbol{\eta}(\mathbf{s}, t),\end{aligned} \tag{11}$$

and $\mathbf{M}$ represents the coefficients associated with the finite difference and the discovered equation. Directly incorporating (5) in the process model results in

$$\begin{aligned}\mathbf{v}(\mathbf{s}, t) &= \mathbf{H}(\mathbf{s}, t)\mathbf{u}(\mathbf{s}, t)+\boldsymbol{\epsilon}(\mathbf{s}, t)\\ \mathbf{u}_{t^{(J)}}(\mathbf{s}, t) &= \mathbf{M}\mathbf{f}(\mathbf{u}(\mathbf{s}, t), \mathbf{u}_x(\mathbf{s}, t), \ldots)+\boldsymbol{\eta}(\mathbf{s}, t),\end{aligned} \tag{12}$$

where now the temporal derivative is directly related to a library of potential functions and $\mathbf{M}$ represents the coefficients associated only with the discovered equation.

The benefit of formulating the problem using (11) is that a Kalman filter or ensemble Kalman filter can be used to estimate parameters (see Stroud *et al.*, 2018; Katzfuss *et al.*, 2020, and Pulido *et al.*, 2018, for examples of the Kalman filter with dynamic systems in statistics). Additionally, as mentioned previously, the GQN framework provides an over-parameterised library of potential dynamical interactions by construction. However, interpreting parameters can be difficult and incorporating spatial derivatives into the library is not as straightforward as with traditional PSMs. In contrast, (12) has a very clear interpretation of parameters but requires a method to relate the previous state to the current state (e.g. numerical differentiation scheme). Additionally, model estimation will rely on Metropolis–Hastings Monte Carlo as the Markov assumption required for Kalman filter and EnKF methods is violated. For both approaches, parameter shrinkage or variable selection or both will need to be employed on $\mathbf{M}$ to produce a sparse solution set. The field of Bayesian variable selection is quite large and there are a variety

of priors that can be used (see George *et al.*, 1993; Park & Casella, 2008; Carvalho *et al.*, 2010; Li & Lin, 2010, for possible choices)

Assuming model estimation is possible, formulating the problem using either (11) or (12) provides significant contributions to the data-driven discovery. In contrast to the sparse regression approaches with uncertainty quantification discussed in Section 2.2.1, (11) and (12) treat the latent process $\mathbf{u}(\mathbf{s}, t)$ as a random process and do not disregard the measurement error when estimating the system. That is, instead of computing derivatives and de-noising prior to model estimation, uncertainty in the derivatives as a product of measurement error is accounted for. This makes estimation more challenging as the derivatives are no longer assumed known *a priori*. Additionally, missing data can be handled through the incidence matrix $\mathbf{H}$. By formulating the problem within a BHM, known methods accounting for missing data can be used, providing more real-world applicability than the deterministic counterparts.

6 Bayesian Dynamic Discovery

In a sequence of papers, North et al. (2022a, 2022b) proposed a fully probabilistic Bayesian hierarchical approach to data-driven discovery of dynamic equations. Similar to the sparse regression approached discussed in Section 2, the Bayesian approach uses a library of potential functions to identify the governing dynamics. However, in contrast to the methods discussed in Sections 2,3, and 4, the dynamic system is modeled as a random process and assumed latent. Specifically, North et al. (2022a, 2022b) use the approach detailed by (12), where the process model in the BHM,

$$\mathbf{u}_{t^{(J)}}(\mathbf{s}, t) = \mathbf{M}\mathbf{f}(\mathbf{u}(\mathbf{s}, t), \mathbf{u}_x(\mathbf{s}, t), \ldots) + \boldsymbol{\eta}(\mathbf{s}, t), \tag{13}$$

directly relates the time derivative of the dynamic system to a library of potential functions.

In its most general form, (13) has three dimensions, space (S), time (T), and the number of components (N), and can be represented as a tensor—a higher-order representation of a matrix (see Kolda, 2006, for a details). Let $\mathcal{U} = \{u(\mathbf{s}, t, n): \mathbf{s} \in D_s, t = 1, \ldots, T, n = 1, \ldots, N\}$ where $\mathcal{U} \in \mathbb{R}^{S \times T \times N}$ is the tensor of the dynamic process. Similarly, let $\mathcal{F} \in \mathbb{R}^{S \times T \times D}$ be the tensor of the function $\mathbf{f}(\cdot)$ evaluated at each location in space-time and $\tilde{\boldsymbol{\eta}} \in \mathbb{R}^{S \times T \times N}$ the space-time-component uncertainty tensor. The process model can represented compactly using tensor notation as

$$\mathcal{U}_{t^{(J)}} = \mathcal{F} \times_3 \mathbf{M} + \tilde{\boldsymbol{\eta}}. \tag{14}$$

While not explicitly stated, $\mathcal{F}$ is still a function of the state process $\mathcal{U}$ and its derivatives.

The process $\mathcal{U}$ can be defined using a finite collection of spatial, temporal and component basis functions. Let

$$\mathcal{U} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} a(p, q, r)\boldsymbol{\psi}(p) \circ \boldsymbol{\phi}(q) \circ \boldsymbol{\theta}(r) = \mathcal{A} \times_1 \boldsymbol{\Psi} \times_2 \boldsymbol{\Phi} \times_3 \boldsymbol{\Theta} := [[\mathcal{A}; \boldsymbol{\Psi}, \boldsymbol{\Phi}, \boldsymbol{\Theta}]],$$

where $\mathcal{A} \in \mathbb{R}^{P \times Q \times R}$, $\boldsymbol{\Psi} \in \mathbb{R}^{S \times P}$, $\boldsymbol{\Phi} \in \mathbb{R}^{T \times Q}$, and $\boldsymbol{\Theta} \in \mathbb{R}^{N \times R}$. Here, $\boldsymbol{\Psi}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\Theta}$ are matrices of spatial, temporal, and component basis functions, respectively, and $\mathcal{A}$ is a tensor of basis coefficients (traditionally called the *core tensor*). Defining $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ to be

matrices of basis functions differentiable up to at least the highest order considered in (1), derivatives of $\mathcal{U}$ can be obtained analytically by computing derivatives of the basis functions. For example,

$$\frac{\partial^3}{\partial x \partial y \partial t}\mathcal{U} = \mathcal{U}_{xyt} = \mathcal{A} \times_1 \boldsymbol{\Psi}_{xy} \times_2 \boldsymbol{\Phi}_t \times_3 \boldsymbol{\Theta} \equiv [[\mathcal{A};\ \boldsymbol{\Psi}_{xy},\ \boldsymbol{\Phi}_t,\ \boldsymbol{\Theta}]],$$

where we use $[[\cdot]]$ as shorthand for the tensor product. Defining the compact tensor representation of the dynamic process (14) using the basis decomposition, we can write

$$[[\mathcal{A};\ \boldsymbol{\Psi},\ \boldsymbol{\Phi}_{t^{(J)}},\ \boldsymbol{\Theta}]] = \mathcal{F} \times_3 \mathbf{M} + \boldsymbol{\eta}, \tag{15}$$

where $\boldsymbol{\eta}$ may now include truncation error due to the approximation of the process using basis function. While not explicitly stated, the arguments of $\mathcal{F}$ are now $\boldsymbol{\Psi}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Theta}$, $\mathcal{A}$, and appropriate derivatives of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$.

Taking the mode-3 matricisation of (15)—the flattening of a tensor to a matrix (see Kolda, 2006, for detail on this operation)—yields

$$\boldsymbol{\Theta}\mathbf{A}(\boldsymbol{\phi}_{t^{(J)}}(t) \otimes \boldsymbol{\psi}(\mathbf{s}))' = \mathbf{Mf}(\mathbf{A},\ \boldsymbol{\psi}(\mathbf{s}),\ \boldsymbol{\psi}_x(\mathbf{s}),\ \boldsymbol{\psi}_y(\mathbf{s}),\ \boldsymbol{\psi}_{xy}(\mathbf{s}),\ \ldots,\ \boldsymbol{\phi}_{t^{(0)}}(t),\ \ldots,\ \boldsymbol{\phi}_{t^{(J)}}(t),\ \boldsymbol{\omega}(\mathbf{s},\ t)) + \boldsymbol{\eta}(\mathbf{s},\ t),$$

where $\mathbf{A}$ is a $R \times PQ$ matrix of basis coefficients, $\boldsymbol{\psi}(\mathbf{s})$ is a length-P vector of spatial basis functions, $\boldsymbol{\phi}(t)$ is a length-Q vector of temporal basis functions, and $\boldsymbol{\Theta}$ is a $N \times R$ matrix of component basis functions (see North *et al.*, 2022b, for a detailed explanation). This form is convenient because only $\mathcal{A}$ (or $\mathbf{A}$) needs to be estimated to fully define the dynamic process and its derivatives as opposed to requiring all spatial, temporal, or spatio-temporal derivatives of the process.

Equation (1) can be extended by rewriting the left-hand side (LHS) to accommodate spatio-temporal derivatives of the process (e.g. $\nabla^2 \mathbf{u}_t(\mathbf{s},\ t) = \mathbf{u}_{xxt}(\mathbf{s},\ t) + \mathbf{u}_{yyt}(\mathbf{s},\ t)$), which is common in fluid dynamics (see Higham *et al.*, 2016, for examples). Specifically, consider the more general PDE

$$g(\mathbf{u}_{t^{(J)}}(\mathbf{s},\ t)) = \mathcal{M}\Big(\mathbf{u}(\mathbf{s},\ t),\ \mathbf{u}_x(\mathbf{s},\ t),\ \mathbf{u}_y(\mathbf{s},\ t),\ \boldsymbol{\psi}_{xy}(\mathbf{s}),\ \ldots,\ \mathbf{u}_{t^{(1)}}(\mathbf{s},\ t),\ \ldots,\ \mathbf{u}_{t^{(J-1)}}(\mathbf{s},\ t),\ \boldsymbol{\omega}(\mathbf{s},\ t)\Big), \tag{16}$$

where $g(\cdot)$ is a linear differential operator. The basis formulation of (16) is

$$\boldsymbol{\Theta}\mathbf{A}(\boldsymbol{\phi}_{t^{(J)}}(t) \otimes g(\boldsymbol{\psi}(\mathbf{s})))' = \mathbf{Mf}(\mathbf{A},\ \boldsymbol{\psi}(\mathbf{s}),\ \boldsymbol{\psi}_x(\mathbf{s}),\ \boldsymbol{\psi}_y(\mathbf{s}),\ \boldsymbol{\psi}_{xy}(\mathbf{s}),\ \ldots,\ \boldsymbol{\phi}_{t^{(0)}}(t),\ \ldots,\ \boldsymbol{\phi}_{t^{(J)}}(t),\ \boldsymbol{\omega}(\mathbf{s},\ t)) + \boldsymbol{\eta}(\mathbf{s},\ t),$$

where $\boldsymbol{\eta}(\mathbf{s},\ t) \overset{i.i.d.}{\sim} \mathrm{N}_N(\boldsymbol{0},\ \Sigma_U)$ in space and time (see North *et al.*, 2022b, for details). This results in the general BHM for location $\mathbf{s}$ and time t

$$\mathbf{v}(\mathbf{s},\ t) = \mathbf{H}(\mathbf{s},\ t)\boldsymbol{\Theta}\mathbf{A}(\boldsymbol{\phi}_{t^{(0)}}(t) \otimes \boldsymbol{\psi}(\mathbf{s}))' + \boldsymbol{\epsilon}(\mathbf{s},\ t)$$

$$\boldsymbol{\Theta}\mathbf{A}(\boldsymbol{\phi}_{t^{(J)}}(t) \otimes g(\boldsymbol{\psi}(\mathbf{s})))' = \mathbf{Mf}(\mathbf{A},\ \boldsymbol{\psi}(\mathbf{s}),\ \boldsymbol{\psi}_x(\mathbf{s}),\ \boldsymbol{\psi}_y(\mathbf{s}),\ \boldsymbol{\psi}_{xy}(\mathbf{s}),\ \ldots,\ \boldsymbol{\phi}_{t^{(0)}}(t),\ \ldots,\ \boldsymbol{\phi}_{t^{(J)}}(t),\ \boldsymbol{\omega}(\mathbf{s},\ t)) + \boldsymbol{\eta}(\mathbf{s},\ t),$$

where $\boldsymbol{\epsilon}(\mathbf{s},\ t) \overset{indep.}{\sim} \mathrm{N}_{L(\mathbf{s},\ t)}(\boldsymbol{0},\ \boldsymbol{\Sigma}_V(\mathbf{s},\ t))$ and $\boldsymbol{\eta}(\mathbf{s},\ t) \overset{i.i.d.}{\sim} \mathrm{N}_N(\boldsymbol{0},\ \boldsymbol{\Sigma}_U)$.

Model parameters are estimated by sampling from their full-conditional distributions using Markov chain Monte Carlo, requiring the specification of prior distributions. We provide a brief

summary of the model priors, but for complete model specification, we refer the reader to North et al. (2022a, 2022b). Standard diffuse priors can be assigned to $\boldsymbol{\Sigma}_V(\mathbf{s}, t)$ and $\boldsymbol{\Sigma}_U$. To induce sparsity in $\mathbf{M}$, the spike-and-slab prior (Mitchell & Beauchamp, 1988; George *et al.*, 1993) is used, where a latent indicator variable matrix $\boldsymbol{\Gamma}$ of the same dimension as $\mathbf{M}$ denotes if an element of $\mathbf{M}$ is included in the discovered equation or not. A constant issue in discovering equations for PDE systems is multicollinearity in the library. See North *et al.* (2022b) for a subsampling approach proposed to mitigate the impacts of multicollinearity. Last, the elastic net prior (Li & Lin, 2010) is assigned to $\mathbf{A}$ to help regularise the basis coefficients. Because $\mathbf{A}$ is embedded in the non-linear function $\mathbf{f}(\cdot)$, estimation can be problematic (see North *et al.*, 2022b, for more detail). To provide a conjugate updating scheme and reduce computation time, $\mathbf{A}$ is sampled using an adapted version of stochastic gradient descent (SGD) with a constant learning rate (SGDCL; Mandt *et al.*, 2016).

While the Bayesian dynamic discovery model proposed in this section relies on more model assumptions and parameters to estimate compared to the methods discussed in Section 2, the benefit is a fully probabilistic discovery of the dynamic system. Most notably, the latent variable $\boldsymbol{\Gamma}$ provides an inclusion probability for each element of $\mathbf{M}$, enabling a researcher to identify a model based on their own desired confidence (i.e. a model identified where each component is included with at least 50% probability). Additionally, uncertainty quantification can be obtained for the dynamic system $\mathbf{u}(\mathbf{s}, t)$ and all of its subsequent derivatives given the full posterior distribution of $\mathbf{A}$. Example 4 provides more detail on model specification and results for Bayesian dynamic discovery.

Example 4: Burgers' equation with Bayesian discovery

A worked example using Julia can be found in the *BayesianDiscovery.jl* (North *et al.*, 2022b) documentation at https://jsnowynorth.github.io/BayesianDiscovery.jl/dev/#. This example is entirely based on this documentation.

The general setup for the Bayesian approach is the same as the sparse regression setup (Example 2), except derivatives are not calculated *a priori* to model estimation. Instead, this approach requires the number of spatial and temporal basis functions, size of the minibatch for SGDCL, the learning rate and the feature library to be chosen. For reference, the example in the documentation specifies the feature library as

$$\left[u,\ u^2,\ u^3,\ u_x,\ uu_x,\ u^2u_x,\ u^3u_x,\ u_{xx},\ uu_{xx},\ u^2u_{xx},\ u^3u_{xx},\ u_{xxx},\ uu_{xxx},\ u^2u_{xxx},\ u^3u_{xxx}\right].$$

Note that, different from Example 3, this library does not include an intercept term.

There are two major differences in the Bayesian approach compared to the deterministic approach in Example 3. First because this approach allows for uncertainty quantification, the user is able to look at uncertainty bounds on the estimated equation—we present the upper and lower 95% highest posterior density intervals (HPDs). Second, the user can specify the desired inclusion probability due to the latent indicator variable $\boldsymbol{\Gamma}$—for this example, we only keep terms with greater than a 50% inclusion probability to be included in the identified equation. Using the code supplied in the documentation and use the noisy data from Example 1 (e.g. $\zeta = 0.01$) as input data, we obtain 10,000 posterior samples, discarding the first 5,000 as burnin. The identified equation is shown in Table 1, where we see the equation is recovered and no extraneous terms are included.

Table 1. *Discovered Burgers' equation (mean) and lower and upper 95% HPD intervals. The true Burgers' equation is* $u_t = -uu_x + 0.1u_{xx}$.

Statistic	*Discovered equation*
Mean	$u_t = -0.995uu_x + 0.096u_{xx}$
Lower HPD	$u_t = -1.037uu_x + 0.086u_{xx}$
Upper HPD	$u_t = -0.968uu_x + 0.102u_{xx}$

Table 2. *Summary of select discussed papers where the columns are: Library—method used to construct the library, System—type of system, either ODE or PDE, considered, Type—our categorisation of the model (combined with library to get the section it is discussed in) where T is Traditional, B is Bayesian, BO is Bootstrap, and NN is Neural Network, UQ—if uncertainty quantification is considered, Noise—if the approach considers or can accommodate measurement error, Missing Data—if the approach considers or can accommodate missing data, Real Data—if the approach is illustrated using real data.*

Reference	*Library*	*System (ODE/PDE)*	*Type*	*UQ*	*Noise*	*Missing data*	*Real data*
Bongard et al. (2007)	Symbolic	ODE	T	No	No	No	Yes
Schmidt et al. (2009)	Symbolic	ODE	T	No	Yes	No	Yes
Maslyaev et al. (2019)	Symbolic	PDE	T	No	Yes	No	No
Brunton et al. (2016)	Sparse	ODE	T	No	Yes	No	No
Rudy et al. (2017)	Sparse	PDE	T	No	Yes	No	No
Rudy et al. (2019)	Sparse	PDE	T	No	Yes	No	No
Schaeffer (2017)	Sparse	PDE	T	No	Yes	No	No
Hirsh et al. (2021)	Sparse	ODE	B	Yes	Yes	No	Yes
Zhang et al. (2018)	Sparse	PDE	B	Yes	Yes	No*	No
Yang et al. (2020)	Sparse	ODE	B	Yes	Yes	No	No
Bhouri et al. (2022)	Sparse	ODE	B	Yes	Yes	No	No
Fasel et al. (2021)	Sparse	PDE	BO	Yes	Yes	No	Yes
Both et al. (2021)	Sparse	PDE	NN	No	Yes	No	Yes
Xu et al. (2021)	Symbolic	PDE	NN	No	Yes	Yes	No
Long et al. (2019)	Symbolic	PDE	NN	No	No	No	No
Atkinson et al. (2019)	Symbolic	PDE	NN	No	No	No	Yes
North et al. (2022a)	Sparse	ODE	B	Yes	Yes	Yes	Yes
North et al. (2022b)	Sparse	PDE	B	Yes	Yes	Yes	Yes

*Zanna & Bolton (2020) applied this framework to real data.

7 Discussion

While relatively young, the field of data-driven discovery is expanding quickly. Areas currently under-studied include methods that properly account for uncertainty quantification and missing data and applications to real-world data sets (see Table 2). BHMs can address these issues; however, they rely on the same assumptions as the sparse regression approach—the library is pre-specified. Relaxing the pre-specified library assumption while retaining the benefits of the statistical approach promises to be a major improvement in the data-driven discovery realm. To this aim, one promising approach is the recent extension in symbolic regression to the Bayesian framework (Jin *et al.*, 2019). The incorporation of Bayesian symbolic regression into a BHM could provide the next step to a truly user-free, unbiased, method at data-driven discovery. Recent advances in deep modeling such as embedding NNs in the BHM (Zammit-Mangion *et al.*, 2022) could also be explored. To this aim, symbolic regression using a NN can be combined with a BHM, providing an alternate method of joining the approaches.

Real-world data come from a variety of sources such as gridded model output (e.g. reanalysis models), *in situ* observations, and remotely sensed (e.g. satellite) measurements. While gridded model output is convenient because it is generally complete and spatially and temporally continuous, the discovered dynamics are biased due to the nature of how the data product is

constructed. Conversely, *in situ* and remotely sensed measurements, which are a more direct and unbiased observation of the true dynamic process, are more difficult to use because of they can be missing data and the observations are generally inconsistent in space and time. Discovery methods that can either use *in situ* measurements, remotely sensed measurements, or both would be beneficial in that the discovered dynamics would be of the 'true' process and not of the model output. Additionally, methods combining the model output with *in situ* and remotely sensed measurements, where the spatial and temporal domains may be different (e.g. change of support problem), could provide an extension to not only data-driven discovery but also for change-of-support related methods.

A final direction for future work we discuss are methods that can accommodate mixed data types. For example, a predator–prey system that takes into account the vegetation coverage of the prey's spatial domain. Vegetation coverage, a positive continuous variable, is dependent on the number of prey, a positive integer valued variable, which in turn is dependent on the number of predators, another positive integer valued variable. In a system such as this, the vegetation coverage, which is a positive continuous variable, is dependent on the number of prey, a positive integer valued variable, which in turn is dependent on the number of predators, another positive integer valued variable. Methods that are able to discover the dynamics of a system with various observed data types provide a much wider range for real-world applications.

Acknowledgments

The authors would like to acknowledge Drs. Scott Holan and Marianthi Markatou for comments on an early draft. This research was partially supported by the US National Science Foundation (NSF) grant SES-1853096 and the US Geological Survey Midwest Climate Adaptation Science Center (CASC) grant no. G20AC00096.

Data Availability Statement

The data that supports the examples in this article are available at the from the included URL links

References

Ahvanooey, M.T., Li, Q., Wu, M. & Wang, S. (2019). A survey of genetic programming and its applications. *KSII Trans. Int. Inform. Syst.*, **13**(4), 1765–1794.

Amir Haeri, M., Ebadzadeh, M.M. & Folino, G. (2017). Statistical genetic programming for symbolic regression. *Appl. Soft Comput. J.*, **60**, 447–469.

Atkinson, S., Subber, W., Wang, L., Khan, G., Hawi, P. & Ghanem, R. 2019. Data-driven discovery of free-form governing differential equations. arXiv preprint arXiv:1910.05117, pages 1–7.

Bateman, H. (1915). Some recent researches on the motion of fluids. *Monthly Weather Rev.*, **43**(4), 163–170.

Berliner, L.M. 1996. Hierarchical Bayesian time series models. In *Maximum Entropy and Bayesian Methods*, Springer Netherlands: Dordrecht, pp. 15–22.

Berliner, L.M. (2003). Physical-statistical modeling in geophysics. *J. Geophys. Res.: Atmos.*, **108**(D24), 8776.

Berliner, L.M., Milliff, R.F. & Wikle, C.K. (2003). Bayesian hierarchical modeling of air-sea interaction. *J. Geophys. Res.*, **108**(C4), 3104.

Bhouri, M.A. & Perdikaris, P. (2022). Gaussian processes meet NeuralODEs: a Bayesian framework for learning the dynamics of partially observed systems from scarce and noisy data. *Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci.*, **380**(2229), 20210201. https://royalsocietypublishing.org/doi/10.1098/rsta.2021.0201

Bongard, J. & Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proc. Nat. Acad. Sci.*, **104**(24), 9943–9948.

Boninsegna, L., Nüske, F. & Clementi, C. (2018). Sparse learning of stochastic dynamical equations. *J. Chem. Phys.*, **148**(24), 241723.

Both, G.-J., Choudhury, S., Sens, P. & Kusters, R. (2021). DeepMoD: deep learning for model discovery in noisy data. *J. Comput. Phys.*, **428**(1), 109985.
Brunton, S.L., Proctor, J.L. & Kutz, J.N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Nat. Acad. Sci.*, **113**(15), 3932–3937.
Burgers, J.M. 1948. A mathematical model illustrating the theory of turbulence. In *Advances in Applied Mechanics*, Vol. **1**, Elsevier, pp. 171–199.
Carvalho, C.M., Polson, N.G. & Scott, J.G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, **97**(2), 465–480.
Champion, K., Lusch, B., Kutz, J.N. & Brunton, S.L. (2019). Data-driven discovery of coordinates and governing equations. *Proc. Nat. Acad. Sci.*, **116**(45), 22445–22451.
Champion, K., Zheng, P., Aravkin, A.Y., Brunton, S.L. & Kutz, J.N. (2020). A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access*, **8**, 169259–169271.
Chartrand, R. (2011). Numerical differentiation of noisy, nonsmooth data. *ISRN Appl. Math.*, **2011**, 1–11.
Chen, R.T.Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D.K. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, Eds. Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N. & Garnett, R., Vol. **31**, Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf
Chen, Q., Zhang, M. & Xue, B. (2017). Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. *IEEE Trans. Evol. Comput.*, **21**(5), 792–806.
Cleveland, W.S. & Devlin, S.J. (1988). Locally weighted regression: an approach to regression analysis by local fitting. *J. Am. Stat. Assoc.*, **83**(403), 596–610.
Combettes, P.L. & Pesquet, J.-C. 2011. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Springer: New York, pp. 185–212.
Cressie, N.A.C. & Wikle, C.K. (2011). *Statistics for Spatio-Temporal Data*. John Wiley & Sons.
de Silva, B., Champion, K., Quade, M., Loiseau, J.-C., Kutz, J. & Brunton, S. (2020). PySINDy: a Python package for the sparse identification of nonlinear dynamical systems from data. *J. Open Source Softw.*, **5**(49), 2104.
Elton, C. & Nicholson, M. (1942). The ten-year cycle in numbers of the lynx in Canada. *J. Anim. Ecol.*, **11**(2), 215–244.
Epureanu, B.I. & Ghadami, A. (2022). Data-driven prediction in dynamical systems. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, **380**(2229), 20210213. https://royalsocietypublishing.org/toc/rsta/2022/380/2229. Special Edition.
Fasel, U., Kutz, J.N., Brunton, B.W. & Brunton, S.L. (2022). Ensemble-SINDy: robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proc. R. Soc. A*, **478**(2260), 20210904.
Garg, A. & Tai, K. (2012). Review of genetic programming in modeling of machining processes. In *Proceedings of 2012 International Conference on Modelling, Identification and Control, ICMIC 2012*, pp. 653–658.
Gauss, C.F. 1809. Theoria motus corporum coelestium in sectionibus conicis solem ambientium.
George, E.I. & McCulloch, R.E. (1997). Approaches for Bayesian variable selection. *Stat. Sin.*, **7**(2), 339–373.
George, E.I., Mcculloch, R.E., George, E.I. & Mcculloch, R.E. (1993). Variable selection via Gibbs sampling. *J. Am. Stat. Assoc.*, **88**(423), 881–889.
Gladish, D.W. & Wikle, C.K. (2014). Physically motivated scale interaction parameterization in reduced rank quadratic nonlinear dynamic spatio-temporal models. *Environmetrics*, **25**(4), 230–244.
Higham, N.J., Dennis, M.R., Glendinning, P., Martin, P.A., Santosa, F. & Tanner, J. (2016). *The Princeton Companion to Applied Mathematics*. Princeton University Press.
Hirsh, S.M., Barajas-Solano, D.A. & Kutz, J.N. 2021. Sparsifying priors for Bayesian uncertainty quantification in model discovery. arXiv preprint arXiv:2107.02107, pages 1–22.
Hoffman, M.D. & Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Machine Learn. Res.*, **15**, 1593–1623. http://mcmc-jags.sourceforge.net
Hooten, M.B. & Wikle, C.K. (2008). A hierarchical Bayesian non-linear spatio-temporal model for the spread of invasive species with application to the Eurasian Collared-Dove. *Environm. Ecolog. Stat.*, **15**(1), 59–70.
Hornik, K., Stinchcombe, M. & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, **2**(5), 359–366.
Icke, I. & Bongard, J.C. (2013). Improving genetic programming based symbolic regression using deterministic machine learning. In *2013 IEEE Congress on Evolutionary Computation*, pp. 1763–1770. IEEE.
Jin, Y., Fu, W., Kang, J., Guo, J. & Guo, J. 2019. Bayesian symbolic regression. arXiv preprint arXiv:1910.08892.
Katzfuss, M., Stroud, J.R. & Wikle, C.K. (2020). Ensemble Kalman methods for high-dimensional hierarchical dynamic space-time models. *J. Am. Stat. Assoc.*, **115**(530), 866–885.
Khoo, Y., Lu, J. & Ying, L. (2021). Solving parametric PDE problems with artificial neural networks. *Eur. J. Appl. Math.*, **32**(3), 421–435.

Kim, S., Lu, P.Y., Mukherjee, S., Gilbert, M., Jing, L., Ceperic, V. & Soljacic, M. (2021). Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Trans. Neural Netw. Learn. Syst.*, **32**(9), 4166–4177.

Knowles, I. & Renka, R.J. (2012). Methods for numerical differentiation of noisy data. *Electron. J. Differ. Equat. Conf.*, **21**(2012), 235–246.

Kolda, T. (2006). Multilinear operators for higher-order decompositions. In Technical report, Sandia National Laboratories (SNL). Albuquerque, NM, and Livermore, CA (United States).

Korns, M.F. 2014. Extreme accuracy in symbolic regression. In *Genetic Programming Theory and Practice XI*, Springer: New York, pp. 1–30.

Koza, J. (1994). Genetic programming as a means for programming computers by natural selection. *Stat. Comput.*, **4** (2), 1–26.

Koza, J., Keane, M.A. & Rice, J.P. (1993). Performance improvement of machine learning via automatic discovery of facilitating functions as applied to a problem of symbolic system identification. In *IEEE International Conference on Neural Networks*, pp. 191–198. IEEE.

Kuhnert, P.M. 2017. Physical-statistical modeling. In *Wiley statsref: Statistics Reference Online*, Wiley, pp. 1–5.

Lagergren, J.H., Nardini, J.T., Michael Lavigne, G., Rutter, E.M. & Flores, K.B. (2020). Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, **476**(2234), 20190800.

Legendre, A.M. (1806). *Nouvelles méthodes pour la détermination des orbites des cometes*. F. Didot.

Li, Q. & Lin, N. (2010). The Bayesian elastic net. *Bayesian Anal.*, **5**(1), 151–170.

Liu, Y., Kutz, J.N. & Brunton, S.L. (2022). Hierarchical deep learning of multiscale differential equation time-steppers. *Philosoph. Trans. R. Soc. A: Math. Phys. Eng. Sci.*, **380**(2229), 20210200.

Long, Z., Lu, Y. & Dong, B. (2019). PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.*, **399**, 108925.

Long, Z., Lu, Y., Ma, X. & Dong, B. (2017). PDE-Net: learning PDEs from data. In *35th International Conference on Machine Learning, ICML 2018*, Vol. **7**, pp. 5067–5078.

Mandt, S., Hoffman, M. & Blei, D. (2016). A variational analysis of stochastic gradient algorithms. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. **48**, pp. 354–363.

Mardt, A., Pasquali, L., Wu, H. & Noé, F. (2018). VAMPnets for deep learning of molecular kinetics. *Nat. Commun.*, **9**(1), 5.

Martius, G. & Lampert, C.H. 2016. Extrapolation and learning equations, 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings, pp. 1–13.

Maslyaev, M., Hvatov, A. & Kalyuzhnaya, A. 2019. Data-driven partial derivative equations discovery with evolutionary approach. In *Computational Science – ICCS 2019*, Springer International Publishing, pp. 635–641.

Meier, L., Van De Geer, S. & Bühlmann, P. (2008). The group lasso for logistic regression. *J. R. Stat. Soc.: Ser. B (Stat. Methodol.)*, **70**(1), 53–71.

Milliff, R.F., Bonazzi, A., Wikle, C.K., Pinardi, N. & Berliner, L.M. (2011). Ocean ensemble forecasting. Part I: ensemble Mediterranean winds from a Bayesian hierarchical model. *Quart. J. R. Meteorolog. Soc.*, **137**(657), 858–878.

Minnebo, W. & Stijven, S. (2011). Empowering knowledge computing with variable selection - on variable importance and variable selection in regression random forests and symbolic regression. Ph.D. Thesis, Antwerp University, Belgium.

Mitchell, T.J. & Beauchamp, J.J. (1988). Bayesian variable selection in linear regression. *J. Am. Stat. Assoc.*, **83**(404), 1023.

Nicolau, M. & Agapitos, A. 2018. On the effect of function set to the generalisation of symbolic regression models. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ACM: New York, NY, USA, pp. 272–273.

Niven, R., Mohammad-Djafari, A., Cordier, L., Abel, M. & Quade, M. (2020). Bayesian identification of dynamical systems. *Proceedings*, **33**(1), 33.

North, J.S., Wikle, C.K. & Schliep, E.M. (2022a). A Bayesian approach for data-driven dynamic equation discovery. *J. Agricult. Biol. Environm. Stat.*, **1**(1), 1–28.

North, J.S., Wikle, C.K. & Schliep, E.M. (2022b). A Bayesian approach for spatio-temporal data-driven dynamic equation discovery, 1–42. arXiv preprint arXiv:2209.02750.

Park, T. & Casella, G. (2008). The Bayesian lasso. *J. Am. Stat. Assoc.*, **103**(482), 681–686.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. & Lerer, A. (2017). Automatic differentiation in PyTorch Adam. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, pp. 1–4.

Piironen, J. & Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electron. J. Stat.*, **11**(2), 5018–5051.

Pulido, M., Tandeo, P., Bocquet, M., Carrassi, A. & Lucini, M. (2018). Stochastic parameterization identification using ensemble Kalman filtering combined with maximum likelihood methods. *Tellus A: Dyn. Meteorol. Oceanograp.*, **70**(1), 1442099. https://doi.org/10.1080/16000870.2018.1442099

Qi, D. & Harlim, J. (2022). Machine learning-based statistical closure models for turbulent dynamical systems. *Philosoph. Trans. R. Soc. A: Math. Phys. Eng. Sci.*, **380**(2229), 20210205.

Qin, T., Wu, K. & Xiu, D. (2019). Data driven governing equations approximation using deep neural networks. *J. Comput. Phys.*, **395**, 620–635.

Rackauckas, C. & Nie, Q. (2017). DifferentialEquations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia. *J. Open Res. Softw.*, **5**(1), 15.

Raissi, M. (2018). Deep hidden physics models: deep learning of nonlinear partial differential equations. *J. Machine Learn. Res.*, **19**, 1–24.

Raissi, M., Perdikaris, P. & Karniadakis, G.E. 2017a. Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561, Part I:1–22.

Raissi, M., Perdikaris, P. & Karniadakis, G.E. 2017b. Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations. arXiv preprint arXiv:1711.10566, Part II:1–19.

Raissi, M., Perdikaris, P. & Karniadakis, G.E. (2019). Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, **378**, 686–707.

Raissi, M., Yazdani, A. & Karniadakis, G.E. (2020). Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. *Science*, **367**(6481), 1026–1030.

Royle, J.A., Berliner, L.M., Wikle, C.K. & Milliff, R. 1999. A hierarchical spatial model for constructing wind fields from scatterometer data in the Labrador Sea. In *Case Studies in Bayesian Statistics*, Springer: New York, NY, pp. 367–382.

Rudy, S., Alla, A., Brunton, S.L. & Kutz, J.N. (2019). Data-driven identification of parametric partial differential equations. *SIAM J. Appl. Dyn. Syst.*, **18**(2), 643–660.

Rudy, S.H., Brunton, S.L., Proctor, J.L. & Kutz, J.N. (2017). Data-driven discovery of partial differential equations. *Sci. Adv.*, **3**(4), e1602614.

Rudy, S.H., Nathan Kutz, J. & Brunton, S.L. (2019). Deep learning of dynamics and signal-noise decomposition with time-stepping constraints. *J. Comput. Phys.*, **396**, 483–506.

Sahoo, S.S., Lampert, C.H. & Martius, G. (2018). Learning equations for extrapolation and control. In *35th International Conference on Machine Learning, ICML 2018*, Vol. **10**, pp. 7053–7061.

Schaeffer, H. (2017). Learning partial differential equations via data discovery and sparse optimization. *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, **473**(2197), 20160446.

Schmidt, M. & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, **324**(5923), 81–85.

Shea, D.E., Brunton, S.L. & Kutz, J.N. (2021). SINDy-BVP: sparse identification of nonlinear dynamics for boundary value problems. *Phys. Rev. Res.*, **3**(2), 23255.

Stroud, J.R., Katzfuss, M. & Wikle, C.K. (2018). A Bayesian adaptive ensemble Kalman filter for sequential state and parameter estimation. *Monthly Weather Rev.*, **146**(1), 373–386.

Sun, Y., Zhang, L. & Schaeffer, H. (2019). NeuPDE: neural network based ordinary and partial differential equations for modeling time-dependent data. *arXiv preprint arXiv:1908.03190*, **107**(2016), 352–372.

Thompson, D.W.J. & Wallace, J.M. (1998). The Arctic oscillation signature in the wintertime geopotential height and temperature fields. *Geophys. Res. Lett.*, **25**(9), 1297–1300.

Tran, G. & Ward, R. (2017). Exact recovery of chaotic systems from highly corrupted data. *Multiscale Model. Simul.*, **15**(3), 1108–1129.

Tsitouras, C. (2011). Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption. *Comput. Math. Appl.*, **62**(2), 770–775.

Wang, Y., Wagner, N. & Rondinelli, J.M. (2019). Symbolic regression in materials science. *MRS Commun.*, **9**(3), 793–805.

Wei, B. (2022). Sparse dynamical system identification with simultaneous structural parameters and initial condition estimation. *Chaos, Solitons Fract.*, **165**(P2), 112866.

Wikle, C.K. (2003). Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology*, **84**(6), 1382–1394.

Wikle, C.K. & Holan, S.H. (2011). Polynomial nonlinear spatio-temporal integro-difference equation models. *J. Time Ser. Anal.*, **32**(4), 339–350.

Wikle, C.K. & Hooten, M.B. (2010). A general science-based framework for dynamical spatio-temporal models. *TEST*, **19**(3), 417–451.

Wikle, C.K., Milliff, R.F., Nychka, D. & Berliner, L.M. (2001). Spatiotemporal hierarchical Bayesian modeling: tropical ocean surface winds. *J. Am. Stat. Assoc.*, **96**(454), 382–397.

Wikle, C.K. & Zammit-Mangion, A. 2022. Statistical deep learning for spatial and spatio-temporal data. arXiv preprint arXiv:2206.02218.

Willis, M.-J. (1997). Genetic programming: an introduction and survey of applications. In *Second International Conference on Genetic Algorithms in Engineering Systems*, pp. 314–319. IET.

Wu, K. & Xiu, D. (2019). Numerical aspects for approximating governing equations using data. *J. Comput. Phys.*, **384**, 200–221.

Wu, K. & Xiu, D. (2020). Data-driven deep learning of partial differential equations in modal space. *J. Comput. Phys.*, **408**, 109307.

Xu, H., Chang, H. & Zhang, D. (2019). DL-PDE: deep-learning based data-driven discovery of partial differential equations from discrete and noisy data. *Commun. Comput. Phys.*, **29**(3), 698–728.

Xu, H., Chang, H. & Zhang, D. (2020). DLGA-PDE: discovery of PDEs with incomplete candidate library via combination of deep learning and genetic algorithm. *J. Comput. Phys.*, **418**, 109584.

Xu, H., Zhang, D. & Zeng, J. (2021). Deep-learning of parametric partial differential equations from sparse and noisy data. *Phys. Fluids*, **33**(3), 37132.

Yang, Y., Aziz Bhouri, M. & Perdikaris, P. (2020). Bayesian differential programming for robust systems identification under uncertainty. *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, **476**(2243), 20200290.

Zammit-Mangion, A., Ng, T.L.J., Vu, Q. & Filippone, M. (2022). Deep compositional spatial models. *J. Am. Stat. Assoc.*, **117**(540), 1787–1808.

Zanna, L. & Bolton, T. (2020). Data-driven equation discovery of ocean mesoscale closures. *Geophys. Res. Lett.*, **47** (17), e2020GL088376. https://onlinelibrary.wiley.com/doi/10.1029/2020GL088376

Zhang, S. & Lin, G. (2018). Robust data-driven discovery of governing physical laws with error bars. *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, **474**(2217), 20180305.

Zhang, L. & Schaeffer, H. (2019). On the convergence of the SINDy algorithm. *Multiscale Model. Simul.*, **17**(3), 948–972.

Zheng, P., Askham, T., Brunton, S.L., Kutz, J.N. & Aravkin, A.Y. (2019). A unified framework for sparse relaxed regularized regression: SR3. *IEEE Access*, **7**, 1404–1423.

Appendix A: Algorithms

Algorithm 1: Sequential threshold least-squares: SINDy

Input: K, κ
Data: $\mathbf{U}_{t^{(J)}}, \mathbf{F}$
Result: $\mathbf{M}$
Initialise: $\mathbf{M} = (\mathbf{F}'\mathbf{F} + \lambda\mathbf{I})^{-1}\mathbf{F}'\mathbf{U}_{t^{(J)}}$
for $k = 1$ **to** K **do**
  $\boldsymbol{\gamma} = |\mathbf{M}| < \kappa$; /* Matrix identifying small coefficients */
  $\mathbf{M}(\boldsymbol{\gamma}) = 0$; /* Threshold $\mathbf{M}$ */
  for $n = 1, \ldots, N$ **do**
    $i := \boldsymbol{\gamma}(n) == 0$; /* Identify non-zero columns */
    $\mathbf{m}(n) = (\mathbf{F}(i)'\mathbf{F}(i))^{-1}\mathbf{F}(i)'\mathbf{U}_{t^{(J)}}$; /* Regress */
  end
end

Algorithm 2: Sequential threshold ridge regression: PDE-FIND

Input: K, κ, λ
Data: $\mathbf{U}_{t^{(J)}}, \mathbf{F}$
Result: $\mathbf{M}$
Initialise: $\mathbf{M} = (\mathbf{F}'\mathbf{F} + \lambda\mathbf{I})^{-1}\mathbf{F}'\mathbf{U}_{t^{(J)}}$
for $k = 1$ **to** K **do**
  $\boldsymbol{\gamma} = |\mathbf{M}| < \kappa$; /* Matrix identifying small coefficients */
  $\mathbf{M}(\boldsymbol{\gamma}) = 0$; /* Threshold $\mathbf{M}$ */
  for $n = 1, \ldots, N$ **do**
    $i := \boldsymbol{\gamma}(n) == 0$; /* Identify non-zero columns */
    $\mathbf{m}(n) = (\mathbf{F}(i)'\mathbf{F}(i) + \lambda\mathbf{I})^{-1}\mathbf{F}(i)'\mathbf{U}_{t^{(J)}}$; /* Regress */
  end
end

Algorithm 3: Sparse Relaxed Regularised Regression: SR3

Input: $K, \kappa, \lambda, tolerance$
Data: $\mathbf{U}_{t^{(J)}}, \mathbf{F}, \mathbf{W}^0$
Result: $\mathbf{M}$
Initialise: $k = 0$, $err = 2 * tolerance$, $\mathbf{M} = (\mathbf{F}'\mathbf{F} + \lambda\mathbf{I})^{-1}\mathbf{F}'\mathbf{U}_{t^{(J)}}$
while $err > tolerance$ **do**
  $k = k + 1$;
  $\mathbf{M}^k = \underset{\widehat{\mathbf{M}}}{argmin} \frac{1}{2}\|\mathbf{U}_{t^{(J)}} - \mathbf{F}\widehat{\mathbf{M}}'\|^2 + \frac{1}{2\nu}\|\widehat{\mathbf{M}} - \mathbf{W}^{k-1}\|^2$;
  $\mathbf{W}^k = \text{prox}_{\lambda,\nu,R}(\mathbf{M}^k)$; /* prox is the proximal gradient */
  $err = \|\mathbf{W}^k - \mathbf{W}^{k-1}\|/\nu$;
end

Algorithm 4: General genetic algorithm

Input: Stopping criteria - ξ, function set, fitness function - $f()$, summary statistic - $T()$
Result: Best individual
Initialise: P = *Randomly generate the initial population based on the defined functional set,* $\Delta_C = 2\xi, \Delta_N = 0$

while $|T(\Delta_C) - T(\Delta_N)| > \xi$ **do**
  $\Delta_C = f(P)$; `/* Evaluate fitness of current individuals */`
  P = Generate new population based on reproduction, crossover, and mutation where individuals are chosen based on fitness level (i.e. higher fitness equals higher probability of being chosen) ;
  $\Delta_N = f(P)$; `/* Evaluate fitness of new individuals */`
end

[*Received October 2022; accepted September 2023*]