

Sparse identification of delay equations with distributed memory

Dimitri Breda^a, Muhammad Tanveer^{a,*}, Jianhong Wu^b

^a CDLab - Computational Dynamics Laboratory Department of Mathematics, Computer Science and Physics - University of Udine, via delle Scienze 206, Udine, 33100, Italy

^b Department of Mathematics and Statistics, Centre of Excellence in AI for Public Health Advancement, York University, Toronto, ON M3J 1P3, Canada

ARTICLE INFO

2020 MSC:

34K29

37M10

65R32

92D25

93B30

Keywords:

Sparse identification

Distributed delay

Delay integro-differential equations

Renewal equations

Quadrature

ABSTRACT

We present a novel extension of the SINDy framework to delay differential equations with *distributed delays* and *renewal equations*, where typically the dependence from the past manifests via integrals in which the history is weighted through specific functions that are in general nonautonomous. Using sparse regression following the application of suitable quadrature formulas, the proposed methodology aims at directly reconstructing these kernel functions, thereby capturing the dynamics of the underlying infinite-dimensional systems. Numerical experiments confirm the effectiveness of the presented approach in identifying accurate and interpretable models, thus advancing data-driven discovery towards systems with distributed memory.

1. Introduction

Data-driven discovery of governing equations has become an important approach for studying dynamical systems, revealing their underlying laws based on observational data. A key framework in this area is the Sparse Identification of Nonlinear Dynamics (SINDy) [1,2]. Originally introduced for Ordinary Differential Equations (ODEs), SINDy offers a method for identifying parsimonious, interpretable models from time series data by using sparse regression on a possibly rich library of candidate functions.

The success of SINDy has led to many extensions, including partial differential equations (PDEs) [3,4], stochastic differential equations [5,6], differential equations with discrete time delays [7–10] and, more recently, stochastic delay differential equations [11,12]. These developments demonstrate the flexibility of such an identification framework, solidifying its importance as a key technique in the consolidated, yet growing, field of data-driven paradigms.

However, a significant gap remains in applying SINDy to systems with distributed memory, which more realistically capture past dependence. These dynamics, modeled by Delay Integro-Differential Equations (DIDEs) and Renewal Equations (REs), are central to many biological, ecological, and engineering systems, in which evolution relies on a range of past states [13–16]. Distributed delays and renewal-type Volterra integral equations model this memory by integrating over a continuum of past states via suitable kernel functions. This paper represents a first attempt to recover the right-hand side of DIDEs and REs from data by directly identifying these constitutive integral kernels to obtain interpretable models rather than black-box tools to just progress simulation in time.

* Corresponding author.

E-mail addresses: dimitri.breda@uniud.it (D. Breda), tanveer.muhammad@spes.uniud.it (M. Tanveer), wujh@yorku.ca (J. Wu).

Note that obtaining information on the form of constitutive kernels is essential to advance knowledge in the complex modeling of emerging or strategic fields such as, e.g., behavioural epidemiology [17–19], tick-borne disease transmission [20,21] or supply chains for sustainable innovation [22–25].

Our method is inspired by recent improvements of the SINDy proposed to address the integral formulation of ODEs [26,27], introduced mainly for better noise resistance by replacing numerical differentiation with integration. Integrals are instead inherently present in DIDEs and REs, with memory kernels that are typically nonautonomous: they both depend on the delay or integration variable (which might represent, e.g., age in the case of age-structured population models) as well as on system state. This continuous nature of memory integration requires careful handling of numerical quadrature, and the main innovation of our work is a consistent approach for creating libraries of candidate functions that clearly include both the state and delay variables. The proposed framework also adopts improved optimization techniques, such as particle swarm optimization (PS) [28–30], to estimate other essential system parameters such as the relevant integration extrema (e.g., unknown maturation or maximum age of individuals in a population). The resulting tool is thus able to simultaneously reveal both the functional form of the memory kernel as well as the integration span.

The contents of the paper are organized as follows. In Section 2 we introduce the basic necessary mathematical description of DIDEs and REs. In Section 3 we revisit the integral SINDy framework of [1,2] for ODEs as a starting basis for the proposed extension. The main methodology for identifying distributed delay systems is introduced in Section 4, where we detail the construction of a library based on quadrature and the formulation of sparse regression. Relevant pseudocodes (provided in Appendix A, with MATLAB demo codes freely available at <https://cdlab.uniud.it/software>) are also discussed therein, together with an explanatory comparison between the newly proposed approach and a standard “black-box” application of SINDy. In Section 5 we present extensive numerical experiments to demonstrate the performance of the resulting tool and examine the effects of key algorithm choices. Finally, we briefly comment on the wider implications of this study and outline further future research directions for data-driven discoveries in the field of infinite-dimensional dynamical systems.

2. Distributed delay differential and renewal equations

In many delayed dynamical systems, the time evolution of the concerned quantity depends not only on its values on just one or few single distinct points in the past, but rather on a weighted contribution of its history (full or possibly partial in between given or unknown time instants), with such weighting described by a delay kernel. These systems are portrayed by models with distributed memory, in which past dependence is expressed via integral functionals. DIDEs and REs are primary classes of such models, together with relevant or even more complex formulations via PDEs (whose treatment we deliberately omit in this work, yet we refer the interested reader to [31,32] as far as models of structured populations are concerned).

In principle, both DIDEs and REs can make use of a common structure for the right-hand side to capture the influence of the history of the system. In this respect a general DIDE is formulated as a differential equation

$$x'(t) = G(x_t) \quad (1)$$

whose rate of change depends on the past, whereas a typical RE

$$x(t) = G(x_t) \quad (2)$$

directly relates the current value of the unknown to its history x_t defined as $x_t(\sigma) := x(t + \sigma)$, $\sigma \in [-\tau, 0]$, for a given maximum delay $\tau > 0$. In both formulations, the central component is a functional $G : X \rightarrow \mathbb{R}^n$ for n a positive integer, which processes the state history and operates in a Banach space X of functions $[-\tau, 0] \rightarrow \mathbb{R}^n$ of suitable regularity (typically continuous for DIDEs and measurable for REs [33]). G accounts for the aggregated influence at the current time of the history of the system on the time span $[-\tau, 0]$.¹ This functional is in general defined by an integral

$$G(\varphi) := \int_{-\tau}^0 g(\sigma, \varphi(\sigma)) d\sigma, \quad \varphi \in X, \quad (3)$$

over the delay interval, where $g : [-\tau, 0] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a smooth possibly nonlinear map termed *kernel* function throughout the paper. g determines the specific nature of the model by assigning a weight to past states. The integration variable σ represents the history time from the present moment t : it directly affects the weight selection (first argument of g) beyond the specific moment at which the past is considered (via the second argument of g). As we clarify in the following sections, this nonautonomous dependence of g from σ is a key aspect for its reconstruction from observed data. We instead conclude this section with a couple of prototype examples of DIDEs and REs.

Example 1. The logistic Daphnia model [34,35]

$$\begin{cases} b(t) = \beta S(t) \int_{a_*}^{a_+} b(t-a) da, \end{cases} \quad (4a)$$

$$\begin{cases} S'(t) = rS(t) \left(1 - \frac{S(t)}{\mathcal{K}} \right) - \gamma S(t) \int_{a_*}^{a_+} b(t-a) da, \end{cases} \quad (4b)$$

¹ We omit to consider formulations with multiple integrals as the following treatment can be generalized straightforwardly also thanks to the additivity of integration. Instead in the case of DIDEs we feel free to include next also current time and discrete delay terms.

offers a simplified framework of an RE coupled to a DIDE to describe a consumer-resource dynamics. Above b denotes the birth rate of a population feeding on a resource S that in the absence of the consumer grows logistically with the growth rate r and carrying capacity K . β and γ are the effective fertility and consumption rates of the adult individuals of the consumer population, which are assumed to have a constant survival probability between the maturation age a_* and the maximal age a_+ . In Section 5 we employ (4) as a representative example of models of coupled DIDEs/REs, making it an ideal starting point for demonstrating the effectiveness of the proposed sparse identification of distributed memory kernels. In the experiments, we will treat both the maturation and the maximum ages as unknown. In terms of (1) and (2) (and recall footnote ¹) the right-hand side of (4) can be rewritten as

$$f(b_t, S_t) + G(b_t, S_t)$$

with

$$f(\varphi, \psi) := \left(0, r\psi(0) \left(1 - \frac{\psi(0)}{K} \right) \right)^T, \quad G(\varphi, \psi) := \int_{-a_+}^{-a_*} g(\sigma, \varphi(\sigma), \psi(\sigma)) d\sigma$$

for

$$g(\sigma, \varphi(\sigma), \psi(\sigma)) := (\beta\psi(0)\varphi(\sigma), -\gamma\psi(0)\varphi(\sigma))^T.$$

In Section 5 we will identify both f and g . Note that in this case, the kernel g is autonomous.

Example 2. In the behavioural epidemiology framework considered, e.g., in [17], the classical SIR paradigm can be expanded to include vaccination conditioned on information decision processes. Consider a stable and equally interacting population structured into compartments comprising susceptible (S), infected (I), recovered (R) and vaccinated (V) individuals, with the vaccination rate governed by an information variable $M(t)$ tracking past disease levels. A plausible mathematical model reads

$$\begin{cases} S'(t) &= \mu[1 - p(M(t))] - \mu S(t) - \beta S(t)I(t), \\ I'(t) &= \beta S(t)I(t) - (\mu + \nu)I(t), \\ R'(t) &= \nu I(t) - \mu R(t), \\ V'(t) &= \mu p(M(t)) - \mu V(t), \end{cases}$$

where μ is the demographic turnover rate, β is the contact rate, and ν is the recovery rate. The coverage function $p(M)$ is assumed to be non-decreasing and is used to express the proportion of immunized newborns in terms of an information variable

$$M(t) := \int_{-\infty}^0 K(\sigma)h(S(t+\sigma), I(t+\sigma))d\sigma.$$

Here, $h : [0, +\infty) \times [0, +\infty) \rightarrow \mathbb{R}$ represents the prevalence of the disease weighted through an integrable, nonnegative memory kernel K normalized such that $\int_{-\infty}^0 K(\sigma)d\sigma = 1$. This ensures that $M(t)$ serves as a weighted average of past disease prevalence. Typically K is an Erlang distribution, but realistic data may require other choices asking for suitable identification tools. Note that in general the local stability at the endemic equilibrium is affected by the properties of K [13]: as the memory concentration increases, the system is likely to have an increased number of intervals with instabilities like Hopf bifurcations and oscillations. This model involves an infinite delay. Analytical tools can handle this case in particular situations (e.g., using the linear chain trick [36] as in [37]), as well as suitable quadrature rules on the half-line can be employed. Time truncation can also prove effective for identification, as we will show in Section 5 for another (Ricker-type) population model with infinite delay that we will consider first in Section 4.3.

3. Integral SINDy for ODEs

The SINDy framework of [1,2] for ODEs recovers parsimonious governing equations from time-series data for dynamical systems. When time derivatives are unavailable, the standard approach relies on numerical differentiation, which may lead to significant sensitivity to the measurement noise. The alternative application of SINDy to the integral form of the underlying ODE has been proposed to improve robustness² [26,27,38]. In this version, hereafter referred to as integral SINDy, the derivative estimation is replaced with numerical integration, which is an inherently more stable operation in noisy settings. This choice has inspired our treatment of systems with distributed memory, therefore we briefly describe its main features next.

Consider the IVP

$$\begin{cases} x'(t) &= f(x(t)), \quad t \in [0, T], \\ x(0) &= x_0, \end{cases} \quad (5)$$

for a general autonomous system of ODEs where $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is an unknown vector field, $x_0 \in \mathbb{R}^n$ and $T > 0$. Integrating (5) yields the equivalent integral form

$$x(t) - x_0 = \int_0^t f(x(\sigma))d\sigma, \quad t \in [0, T]. \quad (6)$$

² The integral formulation is naturally more robust to measurement noise compared to the conventional differential form, as integration acts as a smoothing low-pass filter. Adjusting the sparsity parameter λ in the differential approach cannot provide this level of robustness, since it would only compel the regression to accommodate noise-corrupted derivatives.

In practical scenarios, the state is observed at discrete times $t_1, \dots, t_m$ and the corresponding snapshots $X_j := (x_j(t_1), \dots, x_j(t_m))^T$ are organized in the data matrix

$$\mathbf{X} = (X_1, \dots, X_n) \in \mathbb{R}^{m \times n}. \quad (7)$$

Each component f_j of the vector field is approximated as a sparse linear combination of candidate functions from a predefined library $\Theta(x) \in \mathbb{R}^{m \times p}$. The library may include polynomial, trigonometric, or other nonlinear terms. Hence

$$f_j(x) \approx \Theta(x^T) \xi_j, \quad (8)$$

where $\xi_j \in \mathbb{R}^p$ is a sparse coefficient vector for the j th governing equation. Substituting the latter into (6) requires integrating the candidate functions of the library. To this aim we resort to a simple rectangles formula based on the same time instants of the data sampling, assumed to be uniformly distributed in $[0, T]$ for the sake of simplicity. By letting $h := t_k - t_{k-1}$ we get

$$x_j(t_i) - x_{0,j} \approx \left(\int_0^{t_i} \Theta(x^T(\sigma)) d\sigma \right) \xi_j \approx \left(h \sum_{k=0}^{i-1} \Theta(x^T(\sigma_k)) \right) \xi_j \quad (9)$$

for each state component $j = 1, \dots, n$ and every time instant $t_i, i = 1, \dots, m$. This establishes a relationship between the change in the state variable over the interval $[0, T]$ and a linear combination of the functions of Θ through the unknown coefficients ξ_j . According to the standard SINDy procedure, the regularized least squares regression

$$\xi_j = \arg \min_{\xi \in \mathbb{R}^p} \left\| \mathbf{X}_j - x_{0,j} - \left(h \sum_{k=0}^{i-1} \Theta(\mathbf{X}_{k,:}) \right) \xi \right\|_2 + \lambda \|\xi\|_1 \quad (10)$$

is solved independently for each $j = 1, \dots, n$ to find the sparse vector ξ_j that best describes the dynamics f_j . Above $\mathbf{X}_{k,:}$ denotes the k th row of $\mathbf{X}$ and λ is a regularization parameter that controls sparsity. (10) is a standard convex optimization problem that can be solved efficiently using established tools such as STLS [1] or LASSO [39]. In general one can resort to other quadrature formulas and the approach can be easily adapted to the case of non-equispaced samples by using (piecewise) linear interpolation. Next we adopt the underlying use of quadrature to tackle equations with distributed delays, where in general a remarkable difference from what described here is that the integral kernel is by nature *nonautonomous*.

4. Extending SINDy to identify distributed memory kernels

We first illustrate in Section 4.1 our general approach to identify the kernel g of a distributed memory dependence that characterises the right-hand side (3) of a DIDE (1) or an RE (2). Then in Section 4.2 we briefly discuss relevant algorithms including the use of advanced optimization tools to recover possible unknown delays or non-multiplicative parameters.³ In the sequel, we use the term “external” optimization to refer to this specific problem, which in this context is mainly devoted to recovering unknown integration extrema beyond model parameters characterising special nonlinearities that might be included in the SINDy library in a physics-informed fashion. Finally, in Section 4.3 we compare the new approach with a standard application of SINDy aimed at just reconstructing the unknown dynamics as a “black-box” model, which in general does not provide any information on the underlying integration kernel. This analysis is performed on a specific DIDE with a nonlinear and nonautonomous kernel of Ricker type.

4.1. Quadrature and weighted libraries

In order to extend SINDy to recover distributed memory right-hand sides, it is key to first apply a suitable quadrature formula to approximate the integral representing $G(x_t)$ in (3) as a sum of weighted kernel values:

$$\int_{-\tau}^0 g(\sigma, x(t+\sigma)) d\sigma \approx \sum_{k=1}^K w_k g(\sigma_k, x(t+\sigma_k)). \quad (11)$$

The above K represents the number of quadrature nodes $\sigma_k \in [-\tau, 0]$ and corresponding weights $w_k, k = 1, \dots, K$. According to the driving principle behind SINDy, we assume that the kernel g admits a sparse linear representation within a library of candidate functions Θ . Since at each time t the kernel g is a function of both the delay variable σ and the state value $x(t+\sigma)$, the library must take into account both arguments, thus allowing the recovery of *nonautonomous* kernels. Therefore, the delay variable σ is treated as an explicit candidate function, together with its possible nonlinear dependencies. Then, in view of the quadrature (11), for each $k = 1, \dots, K$ we first consider a library Θ_k of p candidate functions which depend on the quadrature node σ_k and time-shifted state values $x(t+\sigma_k)$.⁴ Following the usual SINDy description relying on the available time samples organized in the matrix $\mathbf{X}$ as in (7), such a k th library may typically include polynomial terms up to a given degree d as

$$\begin{aligned} \Theta_k := \Theta(\sigma_k, \mathbf{X}(\cdot + \sigma_k)) &= [1, \sigma_k, \mathbf{X}_1(\cdot + \sigma_k), \dots, \mathbf{X}_n(\cdot + \sigma_k), \\ &\quad \sigma_k^2, \sigma_k \mathbf{X}_1(\cdot + \sigma_k), \dots, \mathbf{X}_n^2(\cdot + \sigma_k), \\ &\quad \dots, \sigma_k^d, \sigma_k^{d-1} \mathbf{X}_1(\cdot + \sigma_k), \dots, \mathbf{X}_n^d(\cdot + \sigma_k)], \end{aligned}$$

³ Multiplicative ones are indeed contained in the regression vectors ξ_j .

⁴ Note that in principle p can depend on k even though we do not resort to such generality here.

plus any other nonlinear function of either σ_k or $\mathbf{X}_j(\cdot + \sigma_k)$, $j = 1, \dots, n$, that the user can add based on prior model knowledge or any other available information (e.g., e^{σ_k} , $\sin(\mathbf{X}_j(\cdot + \sigma_k))$, etc.). Above $\mathbf{X}_j(\cdot + \sigma_k)$ represents the j th state variable evaluated at shifted time instants $t_i + \sigma_k$, $i = 1, \dots, m$: such values can be recovered by (piecewise linear) interpolation if not already at disposal within $\mathbf{X}$.

A weighted sum of these libraries Θ_k , $k = 1, \dots, K$, through the chosen quadrature weights w_k represents the complete SINDy library to be used in the final regression problem. For an RE (2) the corresponding linear representation reads

$$\mathbf{X}_j \approx \left(\sum_{k=1}^K w_k \Theta(\sigma_k, \mathbf{X}(\cdot + \sigma_k)) \right) \xi_j, \quad (12)$$

for each j th component of the state, $j = 1, \dots, n$. For a DIDE (1) it is enough to replace $\mathbf{X}_j$ at the left-hand side with samples data $\mathbf{X}'_j$ for the time derivatives.⁵ In (12) the vector $\xi_j \in \mathbb{R}^p$ is the sought sparse vector of coefficients that finally selects the active library terms for the j th equation. To get each ξ_j we independently solve the regularized regression problem

$$\xi_j = \arg \min_{\xi \in \mathbb{R}^p} \left(\left\| \mathbf{X}_j - \left(\sum_{k=1}^K w_k \Theta(\sigma_k, \mathbf{X}(\cdot + \sigma_k)) \right) \xi \right\|_2 + \lambda \|\xi\|_1 \right)$$

for each $j = 1, \dots, n$. By collecting all the resulting sparse vectors in a matrix $\Xi := (\xi_1, \dots, \xi_n) \in \mathbb{R}^{p \times n}$, we finally identify (2) with

$$x(t) \approx \left(\int_{-\tau}^0 \Theta(\sigma, x_t^T) d\sigma \right) \Xi$$

(or (1) with $x'(t)$ replacing $x(t)$ at the left-hand side), thus revealing the underlying analytical structure of the kernel g in (3).

4.2. Algorithms and external optimization

Here, we provide a brief description of the algorithms included in Appendix A that are used in Section 5 for the numerical tests. Algorithm 1 outlines a systematic method for identifying DIDEs (1) and REs (2) using the proposed SINDy framework as illustrated in the previous section. The resulting tool relies on a priori knowledge of the involved delays, which give indeed the concerned integration window(s). In realistic situations this information is hardly available, and the same consideration holds for any other (non-multiplicative) parameter characterizing possible specific nonlinear dependencies (e.g., the exponent of an exponential function or the frequency or phase of a sinusoid), as well as for numerical parameters such as the number K of quadrature nodes or their distribution. For this reason, following [7], we integrate the new SINDy framework of Algorithm 1 into a second algorithm, viz. Algorithm 2, which performs an iterative external optimization loop over the parameters of interest to minimize the SINDy reconstruction error given by Algorithm 1. We introduce the latter for REs as

$$\varepsilon(\rho) := \|\mathbf{X} - \Theta(\rho)\Xi(\rho)\|_2, \quad (13)$$

where ρ denotes the vector of possible parameter values in the current step of Algorithm 2 (for DIDEs replace again $\mathbf{X}$ with $\mathbf{X}'$).⁶

External optimization methods, such as PS [28–30] or Bayesian optimization (BO) [41–43], are used to effectively search the parameter space to reduce the error function to a value such that both the recovered kernel and its associated parameters are properly aligned with the observed data. Therefore Algorithm 2 helps uncover not only the sparse structure of the distributed delay kernel (via the “internal” regression of SINDy) but also its bounds, other embedded parameters, and possible numerical settings (via the “external” optimization loop), thereby enhancing both the interpretability and the predictive accuracy of the identified model.

4.3. Comparing kernel reconstruction to a black-box SINDy approach

To illustrate the differences between applying the proposed specialized framework for distributed delays (hereinafter named DD-SINDy) and a standard SINDy “black-box” approach (hereinafter named BB-SINDy) we consider the DIDE

$$x'(t) = d_0 \left(\int_{-\infty}^0 F_{n/\omega}^{(n)}(\sigma) e^{\eta + d_1 \sigma - a x(t+\sigma)} x(t+\sigma) d\sigma - x(t) \right) \quad (14)$$

modeling the growth of a single-species population according to a gamma-distributed maturation delay and a Ricker-type nonlinear reproduction with positive parameters η and a (see [14, equation (4)] or [44,45]). $x(t)$ denotes adult population size at time $t \geq 0$, $d_0 > 0$ the constant death rate of the population, and $d_1 \geq 0$ the constant death rate during the maturation stage. The integral term represents the history-dependent birth rate of the population, with a delay continuously distributed according to

$$F_{\alpha}^{(n)}(\sigma) := \frac{\alpha^n (-\sigma)^{n-1} e^{\alpha \sigma}}{(n-1)!}, \quad \sigma \in (-\infty, 0], \quad (15)$$

⁵ Derivative values can be recovered from $\mathbf{X}$ via suitable techniques for noise reduction [40] if not available from measurements. Alternatively, one can resort to the integral reformulation of the DIDE as explained in Section 3 for ODEs.

⁶ Note that in the experiments part of the data are used to train SINDy and part to validate the recovered model; In this respect (13) is extended to account for both contributions.

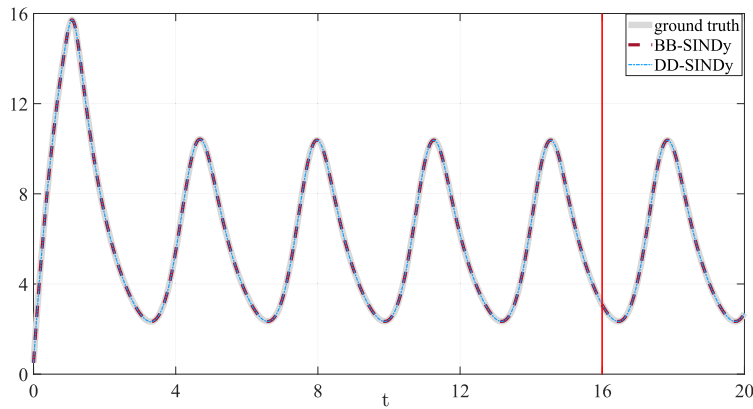


Fig. 1. Comparison of trajectories of (14) reconstructed by DD-SINDy and BB-SINDy. $m = 100$ uniform samples were collected in the time window $[0, 20]$ with $K = 50$ and $\lambda = 10^{-2}$, using 80% for training and 20% for validation.

Table 1

comparison of DD-SINDy with BB-SINDy in terms of the absolute error on the reconstructed coefficients and RMSE (13) on derivatives for (14) in the time window $[0, 20]$ using $m = 100$ samples and $\lambda = 10^{-2}$, with DD-SINDy using $K = 50$ quadrature nodes.

	$ d_0 - \hat{d}_0 $	$ \gamma - \hat{\gamma} $	RMSE $_{x'}$ training	RMSE $_{x'}$ validation
DD-SINDy	3.29×10^{-3}	4.92×10^{-3}	4.36×10^{-2}	3.46×10^{-2}
BB-SINDy	6.81×10^{-2}	–	3.51×10^{-2}	9.96×10^{-2}

for n a positive integer and $\alpha := n/\omega$, with mean $\omega > 0$ and variance ω^2/n . Our objective is to recover the nonautonomous kernel

$$g(\sigma, \varphi(\sigma)) := d_0 F_{\alpha}^{(n)}(\sigma) e^{\eta + d_1 \sigma - a \varphi(\sigma)} \varphi(\sigma) \quad (16)$$

using the time-series data of $x(t)$. For the sake of focusing on the comparison of DD-SINDy with BB-SINDy, in this section we deliberately tackle a simple instance of (14) by choosing $d_1 = 0$, $d_0 = a = 1$, $\eta = \log 80$, $n = 4$ and $\omega = 1$, in which case $\alpha = 4$ and the explicit kernel (16) reduces to

$$g(\sigma, \varphi(\sigma)) := \gamma \cdot \sigma^3 e^{4\sigma} e^{-\varphi(\sigma)} \varphi(\sigma) \quad (17)$$

for $\gamma := -10240/3$. The above choices return $g(\sigma, x(t + \sigma))$ as a product of integer powers of the basis functions σ , e^{σ} , $e^{-x(t+\sigma)}$, $x(t)$ and $x(t + \sigma)$, which we therefore use as candidates to construct a polynomial SINDy library Θ_{DD} of degree (at least) 4 for DD-SINDy. Moreover, we further simplify model (14) by truncating the influence of the history on the integration window $[-10\omega, 0]$ (in Section 5 we consider a more general case with non-integer powers and irrational exponents to challenge the external optimization Algorithm 2 in combination with kernel recovery via DD-SINDy).

As far as BB-SINDy is concerned, (14) is seen just as an instance of the general form (1), so that we seek for a sparse representation of the right-hand side as $G(x_t) \approx \Theta_{BB}(x_t^T) \Xi$. By resorting to a general polynomial library it is clear that one expects BB-SINDy to correctly recover the coefficient $-d_0$ of the linear term in (14), but in general all the other coefficients will not serve to identify interpretable structures for the right-hand side, although the returned model can well serve for making predictions via simulation beyond the time horizon of available data.

For the sake of comparison synthetic time series data for the population x were generated in the time window $[0, 20]$ by numerically solving the IVP for (14) with initial function $\varphi(s) = 0.5$, $s \in [-10\omega, 0]$. Actually MATLAB's `dde23` was used to integrate a version of (14) with the integral replaced by a uniform rectangles-based quadrature with $K = 50$ uniform nodes. Then 100 samples were selected, with 80% reserved for training both DD-SINDy and BB-SINDy, while the remaining data were used for validation to assess the accuracy of the prediction. The same number and distribution of quadrature nodes was used for DD-SINDy, while the sparsity-promoting hyperparameter was set to $\lambda = 10^{-2}$ for both DD-SINDy and BB-SINDy. STLS was used to solve the relevant sparse regression problem. The trajectories of the two recovered models are compared to the ground truth in Fig. 1, showing a remarkable reconstruction capacity for both training and validation.

The root mean square error (RMSE) (13) measured on the derivatives was also used to evaluate the performance of both approaches, see Table 1. The latter shows also the accuracy of the coefficients learnt during sparse regression. DD-SINDy is able to recover both d_0 in (14) and γ in (17), while BB-SINDy returns only the former as already explained.

5. Numerical experiments

In this section we present the results of quantitatively assessing the proposed methodology on three different models by using Algorithms 1 and 2 discussed in Section 4.2 and implemented in Appendix A.

Table 2

absolute errors on the coefficients of the polynomial library terms recovered by DD-SINDy on (18) with degree $d = 2, 3$ and 4 (top); relevant RMSE on the state trajectories (bottom). $m = 100$ uniform samples were collected in the time window $[0, 20]$ with $K = 128$ and $\lambda = 10^{-5}$, using 50% for training and 50% for validation.

Polynomial degree d	2	3	4
$x(t + \sigma)$	6.11×10^{-1}	1.82×10^{-3}	2.69×10^{-1}
$\sigma \cdot x(t + \sigma)$	6.90×10^{-1}	3.06×10^{-3}	4.62×10^{-1}
$x(t + \sigma)^2$	4.19×10^{-2}	2.74×10^{-3}	1.30×10^{-2}
σ^2	9.89×10^{-1}	—	—
$\sigma^2 \cdot x(t + \sigma)$	—	9.00×10^{-3}	2.55×10^{-1}
$\sigma \cdot x(t + \sigma)^2$	—	3.55×10^{-4}	9.93×10^{-3}
$x(t + \sigma)^3$	—	—	5.13×10^{-3}
$\sigma^3 \cdot x(t + \sigma)$	—	—	4.19×10^{-2}
$\sigma^2 \cdot x(t + \sigma)^2$	—	—	1.64×10^{-3}
$\sigma \cdot x(t + \sigma)^3$	—	—	1.90×10^{-3}
$x(t + \sigma)^4$	—	—	8.88×10^{-4}
RMSE training	2.01×10^{-1}	1.40×10^{-3}	1.25×10^{-3}
RMSE validation	1.44×10^{-1}	2.18×10^{-4}	3.70×10^{-4}

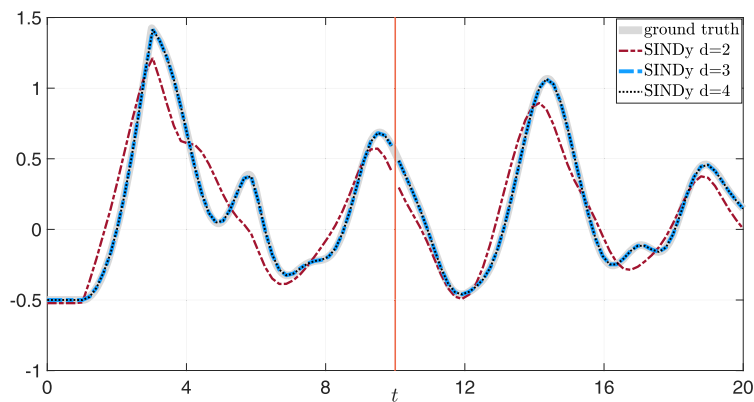


Fig. 2. Comparison of trajectories of (18) reconstructed by DD-SINDy with a polynomial library of degree $d = 2, 3$ and 4; corresponding data in Table 2.

5.1. A logistic renewal equation

We first consider the prototype RE

$$x(t) = \int_{-3}^{-1} (\sigma + 1)x(t + \sigma)[1 - x(t + \sigma)] d\sigma \quad (18)$$

as a toy modification of a classical logistic integral model (see, e.g., [46–48]). Data were generated using a time integration scheme from [49] on the relevant IVP with constant history function $\varphi(s) = 0.5$, $s \in [-3, 0]$. m equidistant samples were collected on $[0, 20]$, using the first 50% for training and the rest for validation. Polynomial SINDy libraries of degree $d = 2, 3$ and 4 were constructed in combination with trapezoidal quadrature. Table 2 presents the active library terms recovered by DD-SINDy with $m = 100$, $K = 128$ and $\lambda = 10^{-5}$, together with relevant RMSE on the state trajectories for both training and validation. Correspondingly, Fig. 2 illustrates the reconstructed trajectories. The quadratic library proves insufficient to capture the system dynamics, while extending the library to $d = 4$ slightly degrades parameter accuracy due to the inclusion of redundant higher order terms. Conversely, the cubic library achieves the best performance, reducing the coefficient error to at least 10^{-3} .

Further tests were carried out by varying the number K of quadrature nodes, the hyperparameter λ promoting sparsity and the number m of samples, employing also rectangles and Clenshaw-Curtis quadrature [50–52] for the sake of performance comparison, see Figs. 3–6. In general and unsurprisingly the accuracy improves by increasing the number K of quadrature nodes independently of the fixed value of λ and of the fixed number of samples m , see Fig. 3 for $m = 50$, Fig. 4 for $m = 200$ and Fig. 5 for $m = 1000$. The behavior of different quadrature formulas is similar for low or moderate values of m , while for larger values differences emerge, with Clenshaw-Curtis prevailing on the other choices. At fixed number of quadrature nodes K the accuracy increases with the number of samples m , apparently reaching a barrier already with moderate values on the order of hundreds. Clenshaw-Curtis and trapezoidal rules perform a little better than rectangles, even though no clear trend is visible along the barrier.

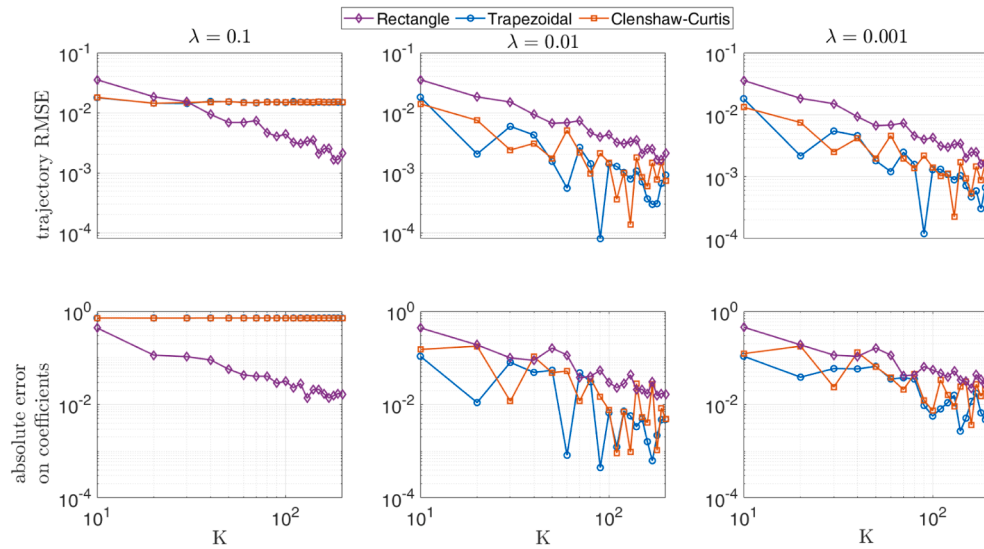


Fig. 3. Comparison of RMSE on trajectories (top row) and absolute error on coefficients (bottom row) for (18), varying K and $\lambda = 0.1$ (left), $\lambda = 0.01$ (center) and $\lambda = 0.001$ (right). $m = 50$ uniform samples were collected in the time window $[0, 20]$ using 50% for training and 50% for validation. A polynomial library of degree $d = 3$ was used.

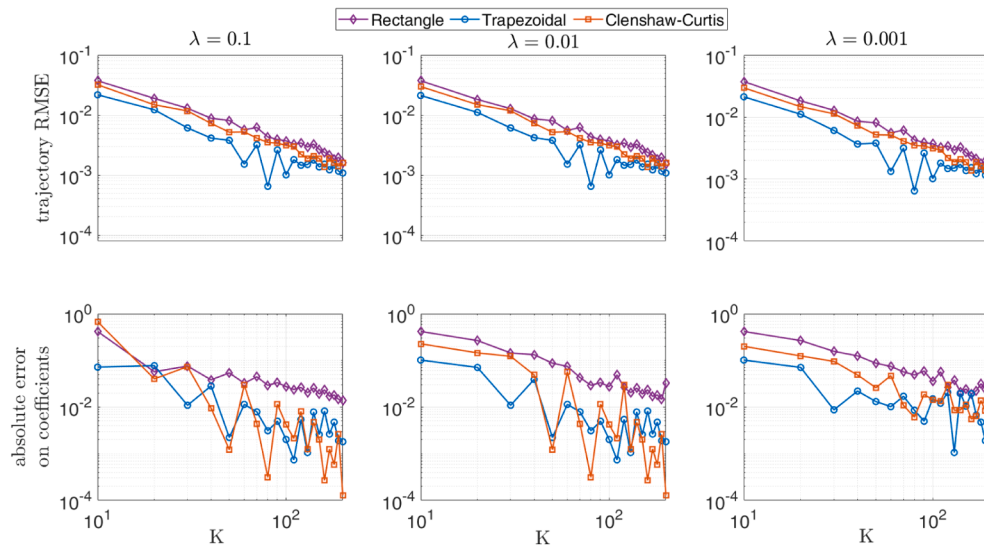


Fig. 4. Comparison of RMSE on trajectories (top row) and absolute error on coefficients (bottom row), varying K and $\lambda = 0.1$ (left), $\lambda = 0.01$ (center) and $\lambda = 0.001$ (right). $m = 200$ uniform samples were collected in the time window $[0, 20]$ using 50% for training and 50% for validation. A polynomial library of degree $d = 3$ was used.

The accurate identification of distributed delay kernels via DD-SINDy turns out to be a delicate balance among the adopted quadrature, an appropriate choice of λ and an adequate sampling (i.e., m). A thorough analysis is out of the scope of the present work, even though the linear decrease of the errors w.r.t. K for all type of quadratures certainly deserves a focused investigation in the future. Let us just notice indeed that while the classical quadrature error is the result of replacing the integrand with a suitable finite-dimensional approximation, typically (piecewise) polynomial on given sets of nodes, in the case of sparse regression one has to take into account also (and mainly) for the linear approximation through the library candidate functions.

Finally, as far as the CPU time is concerned, Fig. 7 illustrates that the latter scales linearly w.r.t. the number of quadrature nodes K , while the number of samples m or the library degree d basically do not affect it. In any case, the overall computational time is quite low when external optimization is not required.

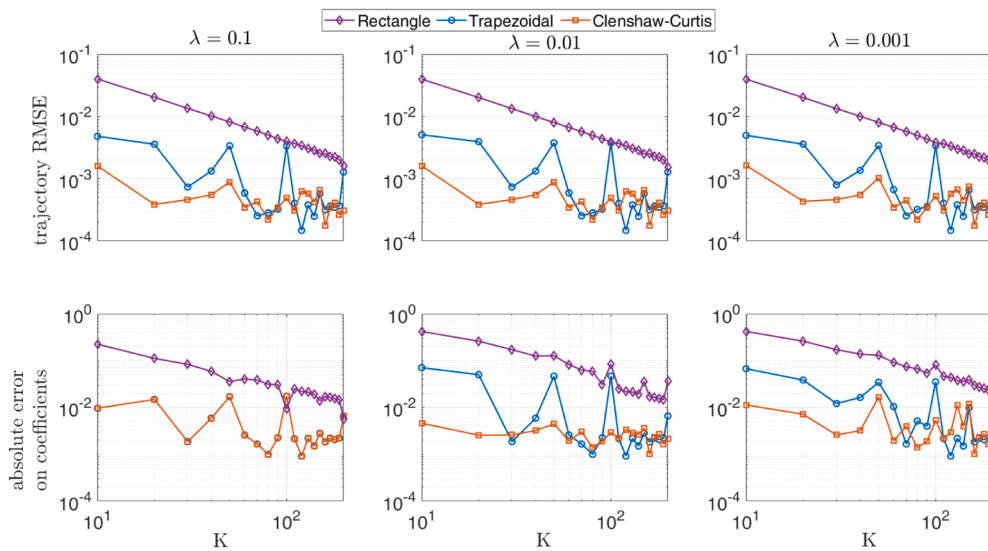


Fig. 5. Comparison of RMSE on trajectories (top row) and absolute error on coefficients (bottom row), varying K and $\lambda = 0.1$ (left), $\lambda = 0.01$ (center) and $\lambda = 0.001$ (right). $m = 1000$ uniform samples were collected in the time window $[0, 20]$ using 50% for training and 50% for validation. A polynomial library of degree $d = 3$ was used.

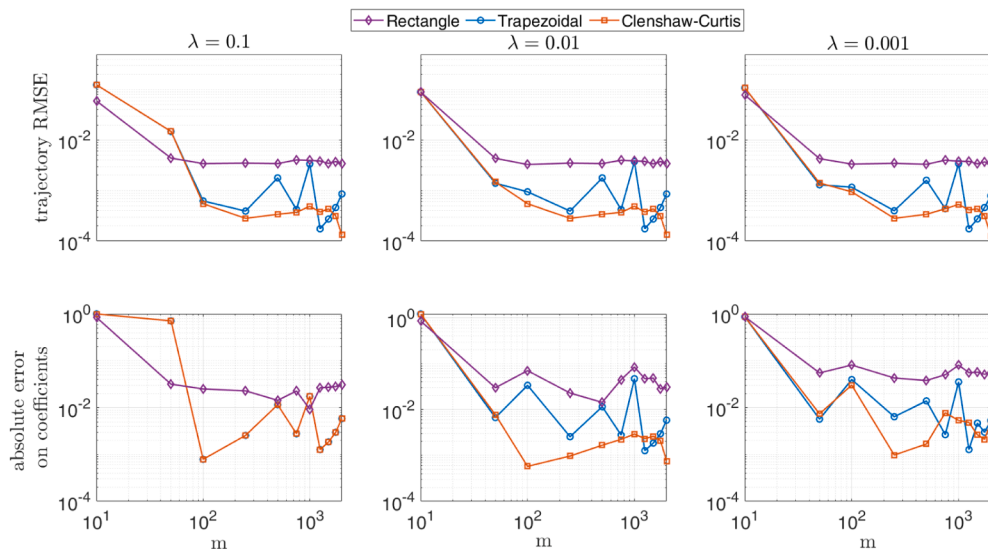


Fig. 6. Comparison of RMSE on trajectories (top row) and absolute error on coefficients (bottom row), varying m and $\lambda = 0.1$ (left), $\lambda = 0.01$ (center) and $\lambda = 0.001$ (right). $K = 100$ quadrature nodes were used and m uniform samples were collected in the time window $[0, 20]$ using 50% for training and 50% for validation. A polynomial library of degree $d = 3$ was used.

5.2. An advanced Ricker-type model

We consider now a version of (14) more sophisticated than that discussed in Section 4.3. Indeed, by keeping the same choices of parameters that led therein to (17), let us change d_1 and a respectively to $d_1 = 0.5$ and $a = \pi/10$, yielding the kernel (16)

$$g(\sigma, \varphi(\sigma)) := \gamma \cdot \sigma^3 e^{4.5\sigma} e^{-\pi\varphi(\sigma)/10} \varphi(\sigma). \quad (19)$$

Therefore while the factor σ^3 can be recovered via a polynomial library that includes the candidate function σ , the two exponential factors need for external optimization of both non-multiplicative parameters d_1 and a when treating them as unknown. In particular Algorithm 2 was used to externally optimize all the unknown values $\omega = 1$, $n = 4$, $\alpha + d_1 = 4.5$ and $a = \pi/10$. The recovered value for n was rounded to the nearest integer and combined with the optimized ω to get first $\alpha = n/\omega$ and then d_1 from the optimized value of $\alpha + d_1$, while a was optimized independently.

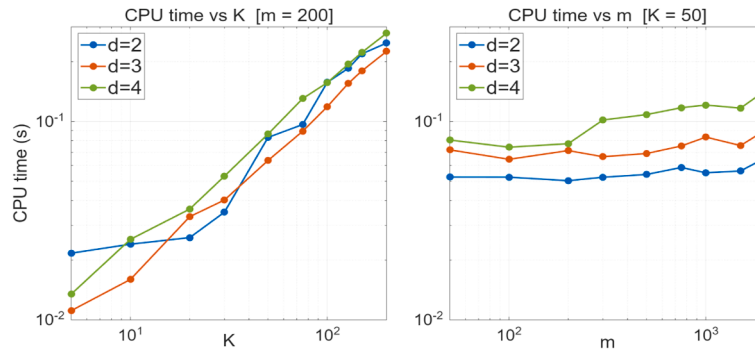


Fig. 7. Comparison of CPU time (s) varying K and $m = 200$ (left), varying m and $K = 50$ (right). Uniform samples were collected in the time window $[0, 20]$ using 50% for training and 50% for validation. Polynomial libraries of degree $d = 2, 3, 4$ were used.

Table 3

absolute errors of optimized values (top), absolute errors on library coefficients (middle) and RMSE on the derivative (bottom) obtained by DD-SINDy on (14) with kernel (19). $m = 500$ uniform samples were collected over $[0, 20]$ with $K = 100$, a polynomial library of degree $d = 5$ and $\lambda = 10^{-2}$, using 80% for training and 20% for validation.

error on externally optimized values	$ n - \hat{n} $ 0	$ d_1 - \hat{d}_1 $ 1.43×10^{-3}	$ \omega - \hat{\omega} $ 1.85×10^{-5}	$ a - \hat{a} $ 3.64×10^{-4}
error on library coefficients	$ d_0 - \hat{d}_0 $ 5.26×10^{-3}	$ \gamma - \hat{\gamma} $ 1.68×10^{-2}		
RMSE _{x'}	training 1.15×10^{-3}	validation 2.74×10^{-4}		

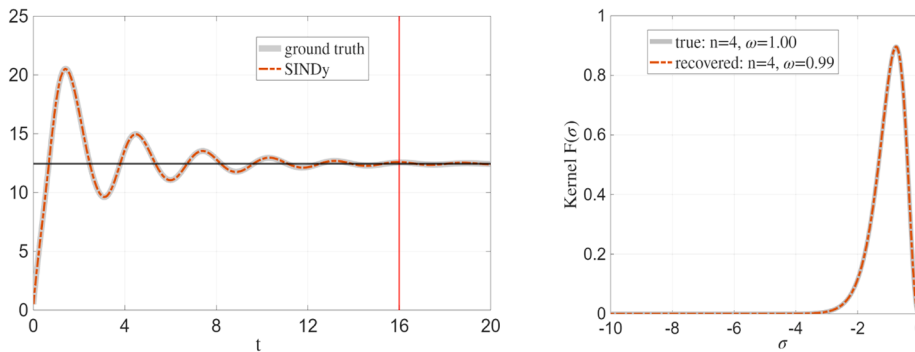


Fig. 8. Comparison of trajectories (left) and kernel functions over the delay interval $[-10\omega, 0]$ (right) of (14) with (19) by DD-SINDy; corresponding data in Table 3, horizontal line in the left panel indicates the equilibrium.⁷

Data for the tests were generated similarly as illustrated in Section 4.3, collecting now $m = 500$ equidistant samples on $[0, 20]$, using the first 80% for training and the rest for validation. A polynomial SINDy library of degree $d = 5$ was employed in combination with trapezoidal quadrature with $K = 100$ nodes and $\lambda = 10^{-2}$. Table 3 collects the results concerning the externally optimized values (top part), the absolute error on the recovered sparse coefficients (central part) and the RMSE on the derivative both for training and validation (bottom part). Correspondingly, Fig. 8 shows the recovered trajectory and kernel, while Fig. 9 illustrates the iterations of the external optimization by Algorithm 2 as explained above.

The obtained results show the excellent performance of DD-SINDy, even when coupled to external optimization to recover unknown delay(s) and non-multiplicative parameters in general. The reconstructed trajectory closely resembles the ground truth during both training and validation. Also the extracted kernel agrees strongly with the actual one. The iterations of Algorithm 2 demonstrate effective convergence and alignment with the true values. Optimization took approximately 1.50×10^3 s for 971 internal calls to DD-SINDy to reach a given tolerance of 10^{-4} , ensuring that PS efficiently identified high-dimensional dynamical structures.

⁷ equilibrium x^* satisfies $x'(t) = 0$, reducing the RHS of (14) to zero, the horizontal line corresponds to the resulting non-trivial positive solution $x^* > 0$.

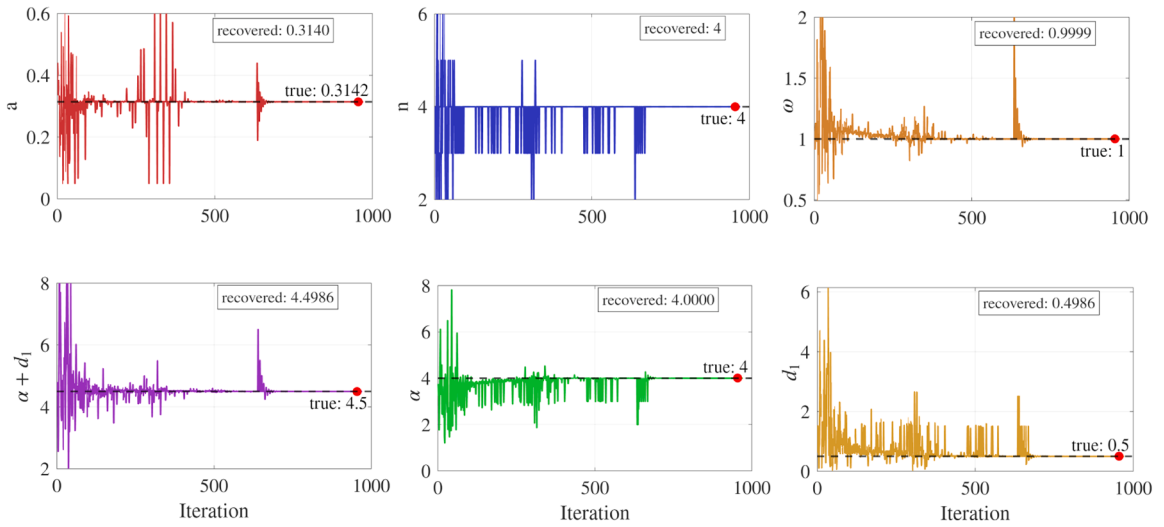


Fig. 9. Iterations of the external optimization of the unknown parameters $\hat{\alpha}$, $\hat{n}$, $\hat{\omega}$, $\hat{\alpha} + \hat{d}_1$, $\hat{\alpha}$ and $\hat{d}_1$ using particle swarm (PS) for the model (14) with (19); red circles mark final optimized values, corresponding data in Table 3.

Table 4

absolute errors of optimized values (top), absolute errors on library coefficients (middle) and RMSE on the derivative (bottom) obtained by DD-SINDy on (14) with kernel (19). A 20% white noise was added to the $m = 500$ uniform samples collected in the time window $[0, 20]$ with $K = 100$, library degree $d = 5$ and $\lambda = 10^{-2}$, using 80% for training and 20% for validation.

error on externally optimized values	$ n - \hat{n} $ 0	$ d_1 - \hat{d}_1 $ 1.48×10^{-3}	$ \omega - \hat{\omega} $ 1.10×10^{-4}	$ \alpha - \hat{\alpha} $ 3.83×10^{-4}
error on library coefficients	$ d_0 - \hat{d}_0 $ 4.02×10^{-3}	$ \gamma - \hat{\gamma} $ 6.75×10^{-2}		
RMSE _{α'}	training 1.19×10^{-3}	validation 2.83×10^{-4}		

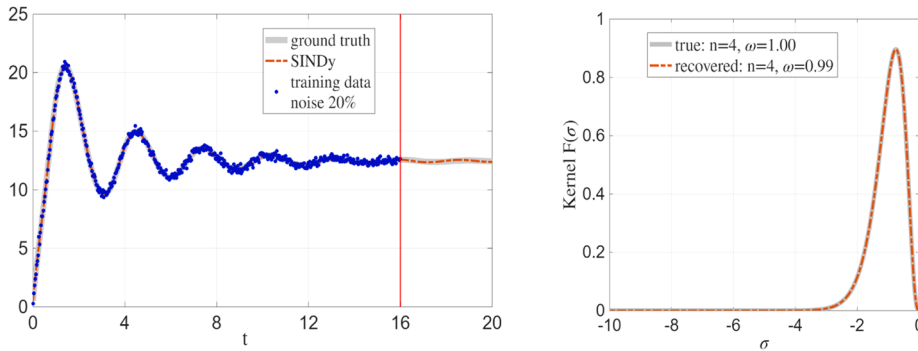


Fig. 10. Comparison of trajectories (left) and kernel functions over the delay interval $[-10\omega, 0]$ (right) of (14) with (19) by DD-SINDy (noisy data); see Table 4.

The results just described are related to a noise-free dataset. Similar trends were observed by adding 20% white noise to the original dataset, see Table 4 and Figs. 10 and 11. The results based on noisy data demonstrate that DD-SINDy retains high generalization and parameter estimation performance despite noisy conditions (see Section 5.4).

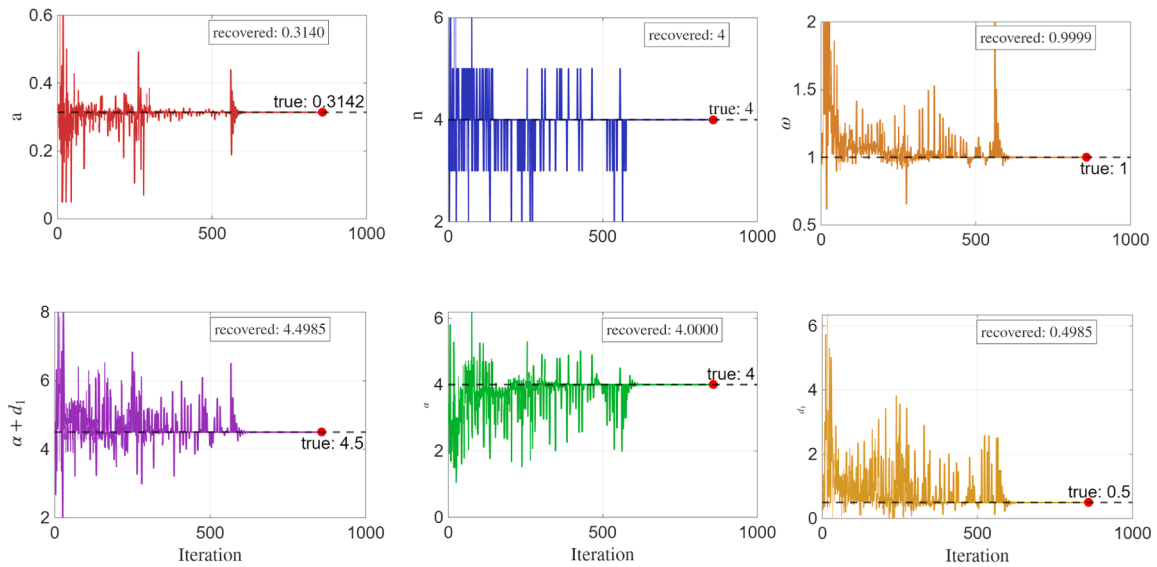


Fig. 11. Iterations of the external optimization of the unknown parameters $\hat{a}$, $\hat{n}$, $\hat{\omega}$, $\hat{\alpha} + \hat{d}_1$, $\hat{\alpha}$ and $\hat{d}_1$ using particle swarm (PS) for the model (14) with (19) on noisy data; red circles mark final optimized values, corresponding data in Table 4.

Table 5

absolute errors of optimized values (top), absolute errors on library coefficients (middle) and RMSE on the derivative (bottom) obtained by DD-SINDy on (4). $m = 1733$ non-uniform samples were collected in the time window $[0, 50]$ with $K = 8$, library degree $d = 2$ and $\lambda = 10^{-3}$, using 80% for training and 20% for validation.

error on externally optimized values	$ a_* - \hat{a}_* $	$ a_{\dagger} - \hat{a}_{\dagger} $		
	1.00×10^{-2}	1.70×10^{-2}		
	(4a)	(4b)	(4b)	(4b)
error on library coefficients	$S(t)b(t-a)$	$S(t)$	$S(t)^2$	$S(t)b(t-a)$
	6.05×10^{-2}	3.01×10^{-3}	4.50×10^{-3}	4.44×10^{-2}
RMSE _{x'}	training	validation		
	7.30×10^{-4}	1.40×10^{-4}		

5.3. The simplified logistic daphnia model

As a last case study we consider the coupled RE/DIDE (4). Fixing $a_* = 3$, $a_{\dagger} = 4$, $r = 1$ and $\gamma = 1$ while varying β , the consumer free equilibrium $(0, \mathcal{K})$ with $\mathcal{K} = 1$ undergoes a transcritical bifurcation at $\beta = (\mathcal{K}(a_{\dagger} - a_*))^{-1} = 1$. This yields a stable nontrivial equilibrium that undergoes a Hopf bifurcation at $\beta \approx 3.0161$ [35].

$m = 1733$ non-uniform time-step samples for both variables b and S were generated using the MATCONT package [53] on the time window $[0, 50]$, using 80% for training and 20% for validation. A polynomial SINDy library of degree $d = 2$ was employed in combination with trapezoidal quadrature with only $K = 8$ nodes and $\lambda = 10^{-3}$. Table 5 collects the results concerning the externally optimized values (top part), the absolute error on the recovered sparse coefficients (central part) and the RMSE on the derivative both for training and validation (bottom part). Correspondingly, Fig. 12 shows the recovered trajectories. The integral bounds a_* and $a_{\dagger}$ were assumed to be unknown, thus identified externally via Algorithm 2. Again DD-SINDy demonstrates excellent performance also when dealing with coupled REs/DIDEs, even in the case of unknown integration window.

The findings from the logistic Daphnia model (4) underscore two key benefits of the proposed DD-SINDy framework. Firstly, the approach is effective in identifying coupled systems where the dynamics of one state variable (the resource S) influence the delayed feedback of another (the consumer birth rate b). Secondly, and more importantly, the algorithm adeptly determines the finite integration window $[a_*, a_{\dagger}]$. The accurate reconstruction of the maturation age a_* and the maximum age $a_{\dagger}$ (refer to Table 5) illustrates that the external optimization loop (Algorithm 2) can identify the “window of influence” in historical data, providing both a predictive model and biologically meaningful insights into the characteristics of life-history of the population.

5.4. Effect of measurement noise: a Monte Carlo study

A detailed statistical analysis of the estimator properties of DD-SINDy is beyond the scope of the present study. However, to give readers a clear sense of the reliability of the proposed method under realistic data conditions, we consider a simple Ricker-type model

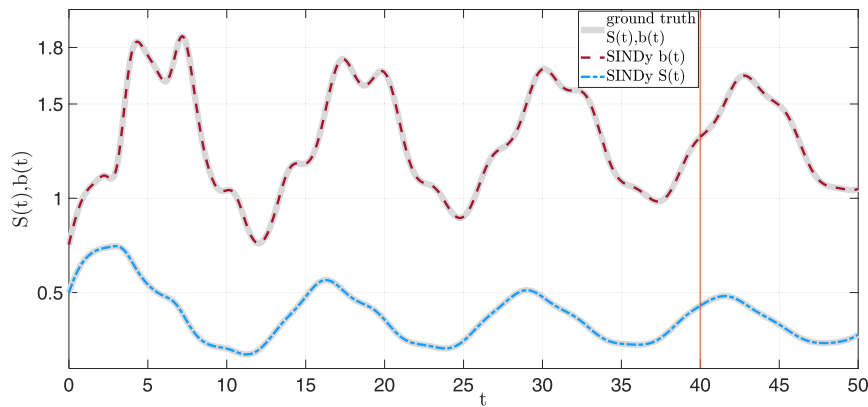


Fig. 12. Comparison of trajectories of (4) reconstructed by DD-SINDy with a polynomial library of degree $d = 2$; corresponding data in Table 5.

Table 6

Monte Carlo analysis of DD-SINDy on (14) over $M = 50$ realizations per noise level. The mean bias and variance of the coefficients γ and d_0 are reported alongside the RMSE on derivatives for both training and validation and the mean CPU time per realization in the time window $[0, 20]$ is roughly 2.4s using $m = 100$ samples and $\lambda = 10^{-2}$, using $K = 50$ quadrature nodes.

Noise (%)	γ		d_0		RMSE _{x'}	
	bias	variance	bias	variance	training	validation
5	8.58×10^{-3}	3.44×10^{-6}	9.94×10^{-3}	4.85×10^{-9}	6.88×10^{-2}	4.49×10^{-2}
10	7.25×10^{-3}	1.74×10^{-5}	9.91×10^{-3}	2.28×10^{-8}	6.88×10^{-2}	4.49×10^{-2}
25	2.63×10^{-3}	1.01×10^{-4}	9.73×10^{-3}	1.43×10^{-7}	6.90×10^{-2}	4.45×10^{-2}
50	7.01×10^{-2}	3.23×10^{-4}	9.32×10^{-3}	4.48×10^{-7}	6.94×10^{-2}	4.47×10^{-2}
75	3.58×10^{-2}	5.28×10^{-4}	8.32×10^{-3}	6.74×10^{-7}	7.01×10^{-2}	4.40×10^{-2}
100	5.32×10^{-2}	3.23×10^{-4}	7.56×10^{-3}	1.18×10^{-6}	7.10×10^{-2}	4.38×10^{-2}

from Section 4.3 with measurement noise. Specifically, following the Monte Carlo approach [54], we generate $M = 50$ independent noise realizations for each of six noise levels $\{0.05, 0.10, 0.25, 0.50, 0.75, 1.00\}$. All other settings remain unchanged ($m = 100$, $K = 50$, $\lambda = 10^{-2}$, 80% training). The bias and variance of the recovered coefficients γ , along with the mean RMSE on derivatives on the training and validation dataset, are shown in Table 6.

The results clearly support the method. The bias of γ stays below 0.2% of the true value for noise levels up to 50%. This confirms that DD-SINDy does not create a systematic error in kernel recovery under moderate noise conditions. However, at higher noise levels of 75% and 100%, the bias of γ rises to 3.58×10^{-2} and 5.32×10^{-2} , respectively. The variance of γ increases steadily from 3.44×10^{-6} at 5% noise to 5.28×10^{-4} at 75% noise. It then levels off at 3.23×10^{-4} at 100% noise. This shows that the sparsity-promoting regression saturates at very high noise levels. The parameter d_0 is recovered with even greater accuracy across the entire spectrum. Its bias decreases steadily, and its variance remains two to three orders of magnitude smaller than that of γ at every noise level. The training RMSE remains notably consistent throughout the entire range of noise levels examined. At every noise level, the validation RMSE is consistently lower than the training RMSE and even shows a slight decrease at higher noise levels. The average CPU time per realization is roughly 2.4 s, indicating that the computational cost of DD-SINDy is completely unaffected by the noise amplitude.

These findings provide solid proof of the reliability and robustness of DD-SINDy in noisy conditions. They also directly address the issue of the sensitivity of the method to measurement noise.

6. Conclusions

This study significantly extends the SINDy framework by expanding its applicability to identify the nonautonomous kernel of the governing equations for systems with distributed delays and REs. By combining a quadrature-based numerical approximation with sparse regression techniques, the proposed method directly reconstructs distributed delay kernels from time-series data. This captures the complex effects of distributed memory inherent in infinite-dimensional dynamical systems.

The proposed methodology introduces several key innovations. First, we develop candidate function libraries that explicitly depend on both delay and state variables, enabling recovery of interpretable kernel structures instead of producing black-box high-dimensional models. Second, the framework applies sparsity-promoting optimization to discover parsimonious representations of distributed memory kernels through weighted sums of quadrature terms. Third, external optimization algorithms jointly identify unknown integration bounds alongside kernel structures, maintaining interpretability throughout. Fourth, comprehensive comparative analysis of various quadrature rules provides practical guidance for practitioners.

This work presents a novel framework marking the first of its kind for distributed delay. Unlike previous research that relied on predefined functional forms to study distributed delays, our method systematically discovers kernel structures and integration limits directly from time-series data. This approach not only enhances interpretability but also surpasses traditional SINDy by revealing complex dynamics within structured population models. This supports future research on more complex functional equations, including systems with features like threshold delays and defined maturation structures [55]. This further broadens the application and scope of data-driven modeling for real-world systems that involve distributed memories.

Declaration of competing interest

None.

Acknowledgements

DB and MT are members of INdAM research group GNCS; DB is a member of UMI research group “Modellistica socio-epidemiologica”. JW is a member of the “Centre of Excellence in Artificial Intelligence for Public Health Advancement” and a member of the “Laboratory for Industrial and Applied Mathematics (LIAM)”. The work of DB was partially supported by the Italian Ministry of University and Research (MUR) through the PRIN 2022 project (No. 20229P2HEA) “Stochastic numerical modelling for sustainable innovation”, Unit of Udine (CUP G53C24000710006). The work of MT was supported by the Italian Ministry of University and Research (MUR) through a PhD grant PNRR DM351/22 (CUP: G23C22001320003). The work of JW was supported in part by the York Research Chair program (grant number: 492108) and by the NSERC-Sanofi Alliance program in Vaccine Mathematics, Modelling, and Manufacturing (517504).

Data availability

Data will be made available on request.

Appendix A. Algorithms

Pseudocodes for the proposed SINDy framework are given below. Relevant MATLAB demo codes are freely available at <https://cdlab.uniud.it/software>.

Algorithm 1 SINDy for distributed delay and renewal equations.

- 1: **Input:** Data $\mathbf{X}, \mathbf{X}'$, function library Θ , delay interval $[-\tau, 0]$, quadrature rule parameters, regularization parameter λ
 - 2: **Output:** Sparse coefficient matrix Ξ representing kernel $g(\sigma, x(t + \sigma))$
 - 3: **Step 1: Quadrature Setup**
 - 4: Discretize the delay interval $[-\tau, 0]$ into K nodes σ_k with weights w_k (e.g., via trapezoidal, Clenshaw-Curtis, etc.)
 - 5: **Step 2: Data Preparation**
 - 6: **for** $k = 1$ to K **do**
 - 7: Compute delay σ_k from integration nodes
 - 8: Obtain time shifted data $\mathbf{X}_{\sigma_k}$ by interpolating $\mathbf{X}$ at times $t_i + \sigma_k$
 - 9: **Step 3: Library Construction** $\Theta_k = \sum_{k=1}^K w_k \cdot \Theta(\sigma_k, \mathbf{X}(t + \sigma_k))$
 - 10: **Step 4: Sparse Regression**
 - 11: **for** $j = 1$ to n (each state component) **do**
 - 12: **if** DIDE: **then** $Y_j = \mathbf{X}'_j$
 - 13: **else (RE:)** $Y_j = \mathbf{X}_j$
 - 14: Solve sparse regression using STLS:
 - 15: $\xi_j = \arg \min_{\xi} \left(\left\| Y_j - \left(\sum_{k=1}^K w_k \Theta_k \right) \xi \right\|_2 + \lambda \|\xi\|_1 \right)$
 - 16: Assemble coefficient matrix: $\Xi := (\xi_1, \xi_2, \dots, \xi_n)$
 - 17: **Step 5: Model Validation**
 - 18: **return** Ξ , validation metrics
-

Algorithm 2 SINDy with parameter optimization for distributed delays.

```

1: Input: Data  $\mathbf{X}, \mathbf{X}'$ , function library  $\Theta$ , parameter search spaces
2: Output: Optimal parameters  $\rho^*$  and sparse coefficient matrix  $\Xi^*$ 
3: function  $\varepsilon(\rho)$ 
4:   Parse parameters: delay bounds  $[\tau_1, \tau_2]$ , quadrature settings from  $\rho$ 
5:   Generate nodes  $\{\sigma_k\}_{k=1}^K$  with weights  $\{w_k\}_{k=1}^K$  in  $[\tau_1, \tau_2]$  and obtain  $\{\mathbf{X}_{\sigma_k}\}_{k=1}^K$  by interpolation
6:   Construct weighted library matrix  $\Theta$  as in Algorithm 1
7:   for  $j = 1$  to  $n$  do Solve sparse regression to get  $\xi_j(\rho)$ 
8:   if DIDE: then  $\varepsilon(\rho) = \|\mathbf{X}' - \Theta(\rho)\Xi(\rho)\|_2$ 
9:   else(RE):  $\varepsilon(\rho) = \|\mathbf{X} - \Theta(\rho)\Xi(\rho)\|_2$ 
10:  return  $\varepsilon(\rho)$ 
11: External Parameter Optimization via PS:  $\rho^* = \arg \min_{\rho} \varepsilon(\rho)$  return  $\rho^*, \Xi^*$ 

```

References

- [1] S.L. Brunton, J.N. Kutz, Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press, 2022.
- [2] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, *Proc. Natl. Acad. Sci.* 113 (15) (2016) 3932–3937.
- [3] N. Demo, M. Tezzele, G. Rozza, PyDMD: python dynamic mode decomposition, *J. Open Source Softw.* 3 (22) (2018) 530.
- [4] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, *Sci. Adv.* 3 (4) (2017) e1602614.
- [5] L. Boninsegna, F. Nüske, C. Clementi, Sparse learning of stochastic dynamical equations, *J. Chem. Phys.* 148 (24) (2018).
- [6] M. Wanner, I. Mezić, On higher order drift and diffusion estimates for stochastic SINDy, *SIAM J. Appl. Dyn. Syst.* 23 (2) (2024) 1504–1539.
- [7] E. Bozzo, D. Breda, M. Tanveer, Sparse identification of time delay systems via pseudospectral collocation, *IFAC-Pap.* 58 (27) (2024) 108–113.
- [8] A.T. Köpeczi-Bócz, H. Sykora, D. Takács, Data-driven delay identification with sindy, in: *International Conference on Nonlinear Dynamics and Applications*, Springer Nature Switzerland, Cham, 2023.
- [9] A. Pecile, N. Demo, M. Tezzele, G. Rozza, D. Breda, Data-driven discovery of delay differential equations with discrete delays, *J. Comput. Appl. Math.* 461 (2025) 116439.
- [10] A. Sandoz, V. Ducret, G.A. Gottwald, G. Vilmart, K. Perron, SINDy for delay-differential equations: application to model bacterial zinc response, *Proc. R. Soc. A Math. Phys. Eng. Sci.* 479 (2269) (2023) 20220556.
- [11] D. Breda, D. Conte, R. D'Ambrosio, I. Santaniello, M. Tanveer, Sparse identification of nonlinear dynamics for stochastic delay differential equations, *J. Comput. Appl. Math.* 453 (2025) 117247.
- [12] P. Han, Q. Guo, H. Zhang, L. Wang, Approximation identification for the stochastic time-delayed dynamical system, *Phys. A Stat. Mech. Appl.* 654 (2024) 130135.
- [13] A. Andø, D. Breda, G. Gava, How fast is the linear chain trick? a rigorous analysis in the context of behavioral epidemiology, *Math. Biosci. Eng.* 17 (5) (2020) 5059–5084.
- [14] E. Beretta, D. Breda, Discrete or distributed delay? effects on stability of population growth, *Math. Biosci. Eng.* 13 (1) (2016) 19–41.
- [15] D. Breda, O. Diekmann, W.F.D. Graaf, A. Pugliese, R. Vermiglio, On the formulation of epidemic models (an appraisal of Kermack and McKendrick), *J. Biol. Dyn.* 6 (Suppl 2) (2012) 103–117.
- [16] J.A. Metz, O. Diekmann (Eds.), *The Dynamics of Physiologically Structured Populations*, Springer, 2014.
- [17] C. Bauch, A. d'Onofrio, P. Manfredi, Behavioral epidemiology of infectious diseases: an overview, in: *Modeling the Interplay Between Human Behavior and the Spread of Infectious Diseases*, 2012, pp. 1–19.
- [18] B. Buonomo, E. Messina, C. Panico, Minimal epidemic models with information index: from compartmental to integral formulation, *Boll. Unione Mat. Ital.* 18 (4) (2025) 1181–1202.
- [19] B. Buonomo, E. Messina, C. Panico, A. Vecchio, An integral renewal equation approach to behavioural epidemic models with information index, *J. Math. Biol.* 90 (2025) 8.
- [20] D. Breda, M. Tanveer, J. Wu, X. Zhang, Exploring memory effects: sparse identification in vector-borne diseases, *PNAS* 123 (26) (2026) e2601074123.
- [21] X. Zhang, H. Cui, Y. Zhou, J. Ding, K. Nah, J. Wu, Patterns of transmission dynamics of severe fever with thrombocytopenia syndrome virus in dalian: roles of systemic, co-feeding, and transovarial routes, *SIAM Trans. Life Sci.* 1 (2) (2025) 299–319.
- [22] L. Guerrini, M. Anokye, A.L. Sackitey, J. Amoah-Mensah, Dynamics of a price adjustment model with distributed delay, *Mathematics* 12 (2024) 3220.
- [23] M. Sarfraz, Leveraging time lag-based diffusion models to predict innovation adoption for optimized product development, *Am. J. Bus. Oper. Res.* 12 (2) (2025) 15–31.
- [24] R. Sipahi, I.I. Delice, Supply network dynamics and delays; performance, synchronization, stability, *IFAC Proc. Vol.* 41 (2) (2008) 6330–6335.
- [25] L. Zhan, H. Shu, X. Zhou, X. Lin, A quality decision model considering the delay effects in a dual-channel supply chain, *Sustainability* 14 (10) (2022) 6240.
- [26] D.A. Messenger, D.M. Bortz, Weak SINDy: Galerkin-based data-driven model selection, *Multiscale Model. Simul.* 19 (3) (2021) 1474–1497.
- [27] B. Wei, Sparse dynamical system identification with simultaneous structural parameters and initial condition estimation, *Chaos Solit. Fractals* 165 (2022) 112866.
- [28] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, *Evol. Comput.* 25 (1) (2017) 1–54.
- [29] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *IEEE International Conference on Neural Networks*, 4, IEEE, 1995, pp. 1942–1948.
- [30] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: *Evolutionary Computation Proceedings*, 890, 1998.
- [31] B. Boldin, O. Diekmann, J.A.J. Metz, Population growth in discrete time: a renewal equation oriented survey, *J. Differ. Eqs. Appl.* 30 (8) (2024) 1062–1090.
- [32] E. Franco, O. Diekmann, M. Gyllenberg, Modelling physiologically structured populations: renewal equations and partial differential equations, *J. Evol. Eqs.* 23 (2023) 46.
- [33] O. Diekmann, P. Getto, M. Gyllenberg, Stability and bifurcation analysis of Volterra functional equations in the light of suns and stars, *SIAM J. Math. Anal.* 39 (4) (2008) 1023–1069.
- [34] D. Breda, O. Diekmann, M. Gyllenberg, F. Scarabel, R. Vermiglio, Pseudospectral discretization of nonlinear delay equations: new prospects for numerical bifurcation analysis, *SIAM J. Appl. Dyn. Syst.* 15 (1) (2016) 1–23.
- [35] D. Breda, D. Liessi, Approximation of eigenvalues of evolution operators for linear coupled renewal and retarded functional differential equations, *Ric. Mat.* 69 (2) (2020) 457–481.
- [36] N. MacDonald, *Time Lags in Biological Models*, Springer Science & Business Media, 2013.
- [37] A. d'Onofrio, P. Manfredi, E. Salinelli, Vaccinating behaviour, information, and the dynamics of SIR vaccine preventable diseases, *Theor. Popul. Biol.* 71 (3) (2007) 301–317.
- [38] H. Schaeffer, S.G. McCalla, Sparse model selection via integral terms, *Phys. Rev. E* 96 (2) (2017) 023302.
- [39] R. Tibshirani, Regression shrinkage and selection via the lasso, *J. R. Stat. Soc. B Stat. Methodol.* 58 (1) (1996) 267–288.
- [40] R. Chartrand, Numerical differentiation of noisy, nonsmooth data, *Int. Sch. Res. Not.* 2011 (1) (2011) 164564.

- [41] R. Garnett, *Bayesian Optimization*, Cambridge University Press, 2023.
- [42] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, N.D. Freitas, Taking the human out of the loop: a review of Bayesian optimization, *Proc. IEEE* 104 (1) (2015) 148–175.
- [43] C.K. Williams, C.E. Rasmussen, *Gaussian Processes for Machine Learning*, 2, MIT Press, Cambridge, MA, Cambridge, MA, 2006.
- [44] Z. Jiang, W. Zhang, Bifurcation analysis in single-species population model with delay, *Sci. China Math.* 53 (6) (2010) 1475–1481.
- [45] W.E. Ricker, Stock and recruitment, *J. Fish. Board Can.* 11 (5) (1954) 559–623.
- [46] D. Breda, O. Diekmann, D. Liessi, F. Scarabel, Numerical bifurcation analysis of a class of nonlinear renewal equations, *Electron. J. Qual. Theory Differ. Eqs.* 2016 (65) (2016) 1–24.
- [47] D. Breda, D. Liessi, Approximation of eigenvalues of evolution operators for linear renewal equations, *SIAM J. Numer. Anal.* 56 (3) (2018) 1456–1481.
- [48] D. Breda, D. Liessi, A practical approach to computing Lyapunov exponents of renewal and delay equations, *Math. Biosci. Eng.* 21 (2024) 1249–1269.
- [49] E. Messina, E. Russo, A. Vecchio, A stable numerical method for Volterra integral equations with discontinuous kernel, *J. Math. Anal. Appl.* 337 (2) (2008) 1383–1393.
- [50] C.W. Clenshaw, A.R. Curtis, A method for numerical integration on an automatic computer, *Numer. Math.* 2 (1) (1960) 197–205.
- [51] W.M. Gentleman, Implementing Clenshaw-Curtis quadrature, I methodology and experience, *Commun. ACM* 15 (5) (1972) 337–342.
- [52] L.N. Trefethen, *Approximation theory and approximation practice*, 2009, (In Book of abstracts).
- [53] D. Liessi, E. Santi, R. Vermiglio, M. Thakur, H.G. Meijer, F. Scarabel, New developments in MatCont: delay equation importer and Lyapunov exponents, (2025). arXiv e-prints arXiv:2504.12785v2
- [54] C.P. Robert, G. Casella, *Monte Carlo Statistical Methods*, 2, Springer, New York, New York, 2004.
- [55] Z. Wang, A.R. Humphries, Numerical methods for delay differential equations with threshold state-dependent delay, *IFAC-Pap.* 55 (36) (2022) 163–168.