Data-Centric Engineering (2026), 7: e22
doi:10.1017/dce.2026.10053

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Neural delay differential equations: learning non-Markovian closures for partially known dynamical systems

Thibault Monsel[1], Onofrio Semeraro[1,2], Guillaume Charpiat[1,3] and Lionel Mathelin[1,2]

[1]Laboratoire Interdisciplinaire des Sciences du Numérique, Université Paris-Saclay, France
[2]CNRS, France
[3]INRIA, France
Corresponding author: Onofrio Semeraro; Email: onofrio.semeraro@cnrs.fr

Received: 12 November 2025 UTC; **Revised:** 03 March 2026 UTC; **Accepted:** 06 May 2026 UTC

Keywords: neural delay differential equation; partial observability; reduced order models; Mori–Zwanzig formalism; scientific machine learning

Abstract

Recent advances in learning dynamical systems from data have shown significant promise. However, many existing methods assume access to the full state of the system—an assumption that is rarely satisfied in practice, where systems are typically monitored through a limited number of sensors, leading to partial observability. To address this challenge, we draw inspiration from the Mori–Zwanzig formalism, which provides a theoretical connection between hidden variables and memory terms. Motivated by this perspective, we introduce a constant-lag neural delay differential equations (NDDEs) framework, providing a continuous-time approach for learning non-Markovian dynamics directly from data. These memory effects are captured using a finite set of time delays, which are identified via the adjoint method. We validate the proposed approach on a range of datasets, including synthetic systems, chaotic dynamics, and experimental measurements, such as the Kuramoto–Sivashinsky equation and cavity-flow experiments. Results demonstrate that NDDEs compare favorably with existing approaches for partially observed systems, including long short-term memory (LSTM) networks and augmented neural ordinary differential equations (ANODEs). Overall, NDDEs offer a principled and data-efficient framework for modeling non-Markovian dynamics under partial observability. An open-source implementation accompanies this article.

Impact statement

This work grounds neural delay differential equations (NDDEs), drawing inspiration from the Mori–Zwanzig formalism to provide a principled framework for learning non-Markovian dynamics in partially observed systems. NDDEs provide a data-efficient and physically interpretable alternative to traditional recurrent and latent-state neural models for complex dynamical systems. An open-source implementation, torchdde, is released alongside this work.

1. Introduction

Describing the dynamics of a system is instrumental in many domains, such as biology (Epstein, 1990; Roussel, 1996), climate research (Ghil et al., 2008; Keane et al., 2019), or finance (Achdou et al., 2012).

This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

© The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

Indeed, an accurate description of the evolution of a system not only improves knowledge and understanding but also allows forecasting, a critical requirement in many situations where decisions must be taken based on predictions, or when devising a suitable sequence of actions to achieve some goal requires a good knowledge of the effect of these actions on the system under consideration. The system is typically supposed to be governed by a model of the form

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}(t)), \qquad \mathbf{u}(t_0) = \mathbf{u}_0, \tag{1.1}$$

with $\mathbf{u}(t)\colon \mathbb{R} \to \mathbb{R}^m$ the time-dependent solution vector representing the state of the system, approximated in a suitable finite-dimensional representation basis, and $\mathbf{f}\colon \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ the vector field. A reliable model may, however, not be available, and one then has to infer it from monitoring the system, here in the form of L distinct trajectories. This model-free approach solely relies on the history of observational data to learn a model for predicting their dynamics. One can use the dataset to fit the ODE dynamics model Eq. (1.1), and many previous efforts have been reported toward this aim. For instance, neural ordinary differential equations (NODEs), introduced in Chen et al. (2018), follow this exact formulation. NODEs gave rise to the class of continuous-depth models and can be viewed as a continuous extension of residual networks (He et al., 2016). An immediate extension of NODEs, referred to as Augmented NODEs (Dupont et al., 2019), explores the existence of certain functions that NODEs are unable to represent. It tackles the expressivity limitation of NODEs, where expressivity is here informally accepted as the richness of the class of functional representations, by expanding the dimension of the solution space through the incorporation of additional variables to learn more complex functions using simpler flows.

However, the state $\mathbf{u}$ is not necessarily available and, in most situations of practical interest, the sole source of information about a system is via a small set of sensors providing a few observations. In addition, the system may not be Markovian and may adopt a behavior that depends not only on its current state but also on its history.

Relying on the past to compensate for the lack of information from the current state is a common approach in partially observed systems where the available observations are not sufficient statistics to predict the future evolution deterministically. One then reverts to predicting an uncertain future, in the form of a probability distribution, as is done with Kalman filters, or to accounting for past measurements to narrow down the possible futures consistent with the observations to a unique one. The problem thus formulates as time-series prediction. State-of-the-art methods for time-series prediction involve autoregressive models like ARMAX (Guidorzi, 2003) and related algorithms; recurrent neural networks such as LSTM, whose distinctive feature is their ability to incorporate a "memory" as a latent variable (Jordan, 1986; Rumelhart et al., 1986; Hochreiter and Schmidhuber, 1997); and echo state networks and reservoir computing (Maass et al., 2002; Jaeger and Haas, 2004), often regarded as an alternative to RNNs for their efficient training and strong performance in capturing long-term statistics when full state dynamics are accessible (Vlachas et al., 2020). Other techniques encompass Latent ODE, which combines NODEs and RNNs together (Rubanova et al., 2019)—a variational autoencoder model using an ODE-RNN encoder and ODE decoder architecture to construct a continuous-time model with a latent state defined at all times—the already mentioned Augmented Neural ODEs (Dupont et al., 2019), or the recently introduced Transformers (Vaswani et al., 2017). While often effective, these techniques lack expressivity (ARMAX, limited in its ability to account for the complex dynamics) or interpretability (recurrent networks, Transformers), in the sense that their resulting architecture cannot be clearly justified by, or related to, specific considerations on the system, in contrast with time delays, which can be linked with physical feed-back mechanisms and time constants, for instance. Further, the discrete nature of some of these approaches conflicts with the continuous formulation of the problem considered in this work, where the focus is on learning a model for a dynamical system from observations, possibly complementing an already available imperfect or coarse model, then formulating it as a closure problem.

A classical framework that formalizes these ideas is the Mori–Zwanzig (MZ) projection formalism, originally developed in statistical physics to derive macroscopic equations for systems with many degrees of freedom. The MZ formalism decomposes the full dynamics into three terms: a Markovian contribution

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

from the directly resolved variables, a memory term that encodes the cumulative influence of the unresolved modes, and a fluctuating noise term. The resulting Generalized Langevin Equation (GLE) provides a mathematically exact description of the reduced system, where the effect of missing physics appears as a convolution integral over the past. In practice, however, this integro-differential form is rarely tractable, motivating the search for parsimonious representations of the memory kernel. Within fluid mechanics, the MZ formalism has increasingly been considered as a theoretical foundation for closure modeling and reduced-order modeling. In Parish and Duraisamy (2017a, b, c), it is shown that finite-memory approximations of the MZ kernel can yield dynamic subgrid-scale models, connecting the MZ projection to large-eddy simulation (LES) closures. These studies also extended to variational multiscale formulations, emphasizing that MZ closures naturally align with hierarchical scale separation and energy transfer mechanisms. Lin et al. (2021) developed algorithms to estimate the Markov, memory, and noise terms directly from simulation data, demonstrating that such operator learning can capture long-term correlations absent in purely Markovian models. In the context of neural models, a recent work by Gupta et al. (2025) explored MZ-based closures in latent spaces obtained from nonlinear autoencoders, showing that embedding MZ dynamics into reduced coordinates can enhance stability and interpretability. The interested reader can refer to the recent overview by Sanderse et al. (2025).

Another approach is to explicitly incorporate non-Markovianity into the formulation by including the historical past state in Equation (1.1). This leads to the domain of neural delay differential equations (NDDEs), which constitutes another subset of models falling under the umbrella of continuous-depth models, alongside NODEs. A generic delay differential equation (DDE) is described by:

$$\begin{aligned} &\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}(t), \mathbf{u}(\alpha_1(t)), \ldots, \mathbf{u}(\alpha_n(t))), \quad \mathbf{u}(t \le 0) = \boldsymbol{\psi}(t), \\ &\alpha_i(t) := t - \tau_i(t, \mathbf{u}(t)), \quad i \in \{1, 2, \ldots, n\}, \end{aligned} \tag{1.2}$$

where $\boldsymbol{\psi}(t): \mathbb{R}^- \to \mathbb{R}^m$ is the history function, $\tau_i: \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^+$ is a delay function, and $\mathbf{f}$ is the vector field. The history function $\boldsymbol{\psi}$ serves as the initial condition for DDEs, analogous to $\mathbf{u}_0$ in ODEs. The modeling capabilities of NDDEs vary based on the chosen type of delay. Inherently, NDDEs incorporate and leverage information from preceding time points, effectively converting the delay term into a dynamic memory mechanism. Initially proposed by Zhu et al. (2021) to learn NDDEs with a single constant delay, subsequent work by Zhu et al. (2023) and Schlaginhaufen et al. (2021) explored constant delays piecewise and developed a stabilizing loss for NDDEs, respectively. Additionally, Oprea et al. (2023) focused on learning a single delay within a small network (fewer than 10 parameters). In Monsel et al. (2024), we revisit the Neural DDE by introducing neural state-dependent DDE (SDDDE), a general and flexible framework that can model multiple and state- and time-dependent delays.

Coming back to the MZ formalism, Gupta and Lermusiaux (2021) demonstrated that neural delay differential equations (NDDEs) could represent non-Markovian effects in a tractable way, motivating the idea of combining neural networks with delay-based memory embeddings within the MZ framework. In a similar spirit, Menier et al. (2023) introduced an exponentially decaying memory, allowing for data-efficient training and interpretability of the decay rate in terms of the time scales of the underlying system. In this work, we extend our previous contributions (Monsel et al., 2024) by embedding NDDEs in the general framework of the MZ and Takens formalisms along similar lines as Gupta and Lermusiaux (2021). Specific to the present work is that the multiple delays are here *learned* rather than being fixed a priori. To this aim, an adjoint formulation is derived and allows efficient learning. The impact of relevant delays on the information extracted from the past is illustrated in Section 3.2. We explore the possibility of learning the values of the delays by deriving a practical training procedure with learnable delays using adjoint differentiation. The adjoint procedure makes NDDEs computationally efficient for large datasets; we illustrate the approach on both synthetic and experimental systems, including chaotic dynamics, partially observed fluid flows, and reduced-order closure models, highlighting how the learned delays may correspond to physical time scales.

The remainder of the article is organized as follows. Section 2 revisits the MZ formalism and discusses how time delays can be used for representing the memory kernel. Section 3 details the proposed NDDE formulation, presents the adjoint training method with learnable delays, and outlines its implementation.

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

We report numerical and experimental validations in Section 4.2. In Section 4.3, NDDEs are used as closure models within a reduced-order framework. Section 5 summarizes the findings, discusses limitations, and outlines future research directions.

2. Modeling partially observed dynamical systems

Let us consider the nonlinear system stated in Eq. (1.1):

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}(t)), \quad \mathbf{u}(t_0) = \mathbf{u}_0, \tag{2.1}$$

with $\mathbf{u}(t)$ evolving on a smooth manifold $\mathcal{S} \subset \mathbb{R}^m$. We assume the system is only observed through a small set of sensor functions belonging to a Hilbert subspace of $L^2(\mathcal{S}, \mathbb{R})$ providing measurements $\{y_i\}_{i=1}^p$, $y_i(\mathbf{u}(t)) : \mathcal{S} \to \mathbb{R}$ and let $\mathbf{y}(\mathbf{u}(t)) = \left(y_1, \ldots, y_p\right)$. The dynamics of these observables can be derived from the model of the system:

$$\frac{\partial \mathbf{y}(\mathbf{u}(t))}{\partial t} = \nabla_{\mathbf{u}} \mathbf{y}\ \mathbf{f}(\mathbf{u}(t)) \equiv \mathcal{L}\mathbf{y}(\mathbf{u}(t)), \tag{2.2}$$

where $\mathcal{L}$ is linear and termed a Liouville operator. Defining the Liouville operator at $t = t_0$ yields $\mathcal{L}\mathbf{y} = \nabla_{\mathbf{u}_0}\mathbf{y}(\mathbf{u}_0)\,\mathbf{f}(\mathbf{u}_0)$ and the observables are then expressed as $\mathbf{y}(\mathbf{u}(t)) = e^{(t-t_0)\mathcal{L}}\mathbf{y}(\mathbf{u}_0)$.

Because the measurements are limited in number, the span $\mathcal{G} \coloneqq \mathrm{span}\left(\{y_i\}_i : \mathcal{S} \to \mathbb{R}\right)$ is finite, low-dimensional, and not invariant under the time-evolution operator governing the dynamics of $\mathbf{y}(\mathbf{u}(t > t_0))$. The dynamics of $\mathbf{y}$ is then not closed in $\mathcal{G}$, reflecting the fact that some information leaks in an orthogonal subspace $\mathcal{G}^{\perp}$. To recover the missing information, it is necessary to describe how to express the dynamics of $\mathbf{y}$ not just as a function of itself, but also accounting for its history. The rationale is that the past values of $\mathbf{y}$ are influenced by unobserved quantities, or unresolved observables, of the system, not contained in $\mathcal{G}$. Carefully processing the history of these observables then allows us to account for the impact of the unresolved observables on the observables $\mathbf{y}$. A visual representation is presented in Figure 1.

2.1. The Mori–Zwanzig formalism

Different approaches leverage this past information. The Mori–Zwanzig (MZ) formalism, rooted in statistical physics, provides a framework to derive an evolution equation for a set of variables, such as macroscopic observables, associated with a high-dimensional dynamical system (Mori, 1965; Zwanzig, 1966; Zwanzig et al., 1972). This framework is instrumental, for instance, in situations where the full state

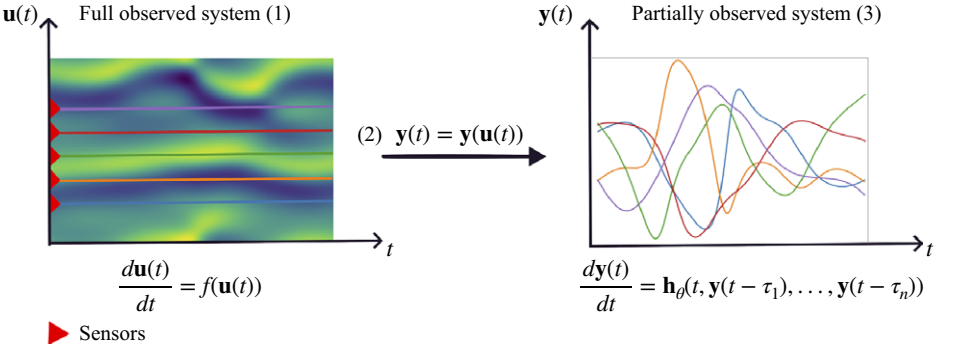


Figure 1. *Measuring the fully observed state of a system (left [1]) is often impossible due to its high dimensionality. Ultimately, the user only has, at their disposal, sparse observations of the full state that can be seen as a low-dimensional observable vector* $\mathbf{y}$ *(middle [2]). The MZ equation DDE approximation (Proposition 2.1) can then be used to model partially observed systems (right [3]).*

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

$\mathbf{u}(t)$ is unavailable, and where one can only access low-dimensional observations. Similarly, the MZ formalism is also relevant for addressing dimension reduction problems (Zhu, 2019) where the reduced state is a set of variables whose dynamics are affected by the projected-out and nonobserved variables or state components.

In the Mori–Zwanzig framework, the span of $\mathcal{G}$ of the measurements is considered as resulting from a projection through a projector $\mathcal{P}$ of the space of $L^2(\mathcal{S},\mathbb{R})$ functions. Similarly, the unobserved functions span a subspace resulting from a projection $\mathcal{Q}$, with $\mathcal{Q}=\mathcal{I}-\mathcal{P}$. The dynamics of $\mathbf{y}$ can then be expressed as

$$\frac{\partial}{\partial t}\mathbf{y}(\mathbf{u}(t))=\frac{\partial}{\partial t}e^{(t-t_0)\mathcal{L}}\mathbf{y}(\mathbf{u}_0)=e^{(t-t_0)\mathcal{L}}(\mathcal{P}+\mathcal{Q})\mathcal{L}\,\mathbf{y}(\mathbf{u}_0). \tag{2.3}$$

The influence of past unresolved dynamics on the current observations can be explicitly expressed using the Dyson identity, which states:

$$e^{(t-t_0)\mathcal{L}}=\int_0^{t-t_0}e^{(t-t_0-s)\mathcal{L}}\mathcal{P}\mathcal{L}\,e^{s\mathcal{Q}\mathcal{L}}\,\mathrm{d}s+e^{(t-t_0)\mathcal{Q}\mathcal{L}}, \tag{2.4}$$

so that one can write the dynamics of the current observables $\mathbf{y}(\mathbf{u}(t))$ as:

$$\frac{\partial}{\partial t}\mathbf{y}(\mathbf{u}(t))=\mathcal{P}\mathcal{L}\,\mathbf{y}(\mathbf{u}(t))+\int_0^{t-t_0}\mathcal{P}\mathcal{L}\,e^{s\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\,\mathbf{y}(\mathbf{u}(t-s))\mathrm{d}s+e^{(t-t_0)\mathcal{Q}\mathcal{L}}\mathcal{Q}\mathcal{L}\,\mathbf{y}(\mathbf{u}_0). \tag{2.5}$$

Rewriting Eq. (2.5) in a more compact form, and denoting $\mathbf{y}(\mathbf{u}(t))=\mathbf{y}(t)$, the dynamics of the vector of observables $\mathbf{y}=(y_1,\ldots,y_p)$ are shown to follow an integro-differential equation (IDE):

$$\frac{\partial\mathbf{y}(t)}{\partial t}=M\,\mathbf{y}(t)+\int_0^{t-t_0}K(\mathbf{y}(t-s),s)\,\mathrm{d}s+F(t,\mathbf{u}_0), \tag{2.6}$$

with M a Markovian operator and K an operator applied to past observables and integrated over the whole time span since the initial condition. The so-called "noise" term F accounts for the impact of the unobserved variables at the initial condition. It can be eliminated from the equations predicting future observations by applying the projector $\mathcal{P}$ and restricting their dynamics to the linear span of $\mathcal{P}\mathbf{y}$. Learning the dynamics of partially observed or known systems hence boils down to estimating each term of the differential equation above, sometimes referred to as the generalized Langevin equation (GLE).

Accounting for the past essentially allows us to isolate the dynamics of the observables. The Mori–Zwanzig framework is general and applies widely, allowing to describe the dynamics of the observables with a non-Markovian model. It similarly provides a principled closure for coarse models, hence missing information, which can be effectively complemented with a history-based term.

2.2. *Approximations of integro-differential equations (IDE)*

The statement in Eq. (2.6) outlines the structure of the dynamics. However, the operators involved in this dynamical equation are usually poorly known, if any, in particular the integral term. We now discuss several approaches from the literature to estimate these terms.

An approach to approximate the integral consists of studying particular asymptotic regimes. Among the many different models proposed in the literature (Stinis, 2004; Chorin and Stinis, 2007; Stinis, 2007), one of the most popular consists in approximating the integral under assumptions of very short memory or very long memory regimes. For example, the *t-model* (Chorin et al., 2002), also commonly called *slowly decaying memory approximation*, leads to Markovian equations with time-dependent coefficients, which can subsequently be modeled using a NODE. However, these remain asymptotic approximations and cannot be extended to intermediate-range memory, in general. Another approach consists of performing Monte Carlo integration (Robert and Casella, 1999). This approach has been extended to a neural network-based formulation (Neural IDE) (Zappala et al., 2023), where the memory integrand is decomposed as a product $K_1(t,s)K_2(\mathbf{y}(s))$. However, the number of function evaluations required to accurately integrate Equation (2.6) can be large, making the process computationally intensive (cf. Appendix A). In a

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

similar spirit, under the assumption of short memory, one can restrict the integral to a short past and discretize it in time, leading to an equation of the form (Gallage, 2017):

$$\frac{\partial \mathbf{y}(t)}{\partial t} \approx M(\mathbf{y}(t)) + \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{y}(t-\tau_i), \tau_i), \tag{2.7}$$

using n delays $\{\tau_i\}_i$ uniformly spaced over a past horizon instead of sampling them randomly. Such approximations can be improved using high-order discretization schemes, yet, as for Neural IDE, they require an unaffordable number of delays if the integrand varies quickly or if the time interval is too large.

2.3. Exact representation with neural DDE

While Equation (2.7) only provides an approximation of the true dynamics and requires many delays, we show that using a more complex function of a small number of delays, it is actually possible to represent the dynamics *exactly*. For this, let us consider diffeomorphic dynamical systems, that is, ODEs whose flow is invertible (in the full state space) and smooth in both time directions.

Proposition 2.1 (Exact representation with delays). For any smooth dynamical system (C^2 is enough), and differentiable observables $\mathbf{y}$, there exists almost surely an operator M of the current observables, a finite number n of delays $\tau_1, \ldots, \tau_n > 0$ and a function $\mathbf{h}$ such that the observables exactly follow the dynamics:

$$\frac{\partial \mathbf{y}(t)}{\partial t} = M(\mathbf{y}(t)) + \mathbf{h}(t, \mathbf{y}(t), \mathbf{y}(t-\tau_1), \mathbf{y}(t-\tau_2), \ldots, \mathbf{y}(t-\tau_n)). \tag{2.8}$$

The derivation of the above, deferred to Appendix C, is based on the application of Takens' theorem (hence the diffeomorphism requirement), which also provides a bound on the required number of delays: at least twice the intrinsic dimension of the manifold $\mathcal{S}$ in which the full state $\mathbf{u}$ lives. Note that this evolution equation is exact: approximations may arise from the optimization or the expressivity of the neural networks estimating M and $\mathbf{h}$, but not from the number of delays, provided they reach the Takens' lower bound. This is in contrast with the discretization of the IDE integral as in Equation (2.7), which becomes asymptotically precise only when the number of delays is large compared to the complexity of the integrand.

3. Neural delay differential equations with learnable delays

The proposition 2.1 above motivates our approach. We consider learning the dynamics of the observables by accounting for their past in the form of a set of time-continuous coupled delayed differential equations (DDEs) involving a number of past observations $\{\mathbf{y}(t-\tau_1), \ldots, \mathbf{y}(t-\tau_n)\}$. The vector field is approximated by a neural network and the resulting model is solved by NDDE, a neural differential equation solver dedicated to delayed equations (Monsel et al., 2024). The Mori–Zwanzig framework provides a rigorous rationale, grounded in statistical physics, on how the dynamics of the observables are related to their past, involving a continuous integral over a past horizon. In contrast, the Takens' theorem provides a geometrical view of the dynamics of a set of observables as a function of a sufficiently large number of delayed versions of themselves. These two frameworks motivate our approach of modeling the dynamics of the observables from a *finite* set of past observations via a learnable function. In addition, while the Takens' theorem is not constructive when it comes to the value of the time lag, we here learn every individual delay $\{\tau_i\}$, $1 \leq i \leq n$. The minimum number of delays n is determined from the D_2 dimension of the underlying system, estimated from the dataset of observations.

3.1. Learning the delays

A constant lag NDDE is part of the larger family of continuous depth models that emerged with NODE (Chen et al., 2018). It is defined by:

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

$$\frac{\partial \mathbf{y}(t)}{\partial t} = \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}(t-\tau_1), \ldots, \mathbf{y}(t-\tau_n)), \quad \mathbf{y}(t \leq 0) = \boldsymbol{\psi}(t), \tag{3.1}$$

with $\boldsymbol{\psi}$: $\mathbb{R} \to \mathbb{R}^m$ the history function, $\tau_i \in \mathbb{R}^+$ a delay constant and $\mathbf{h}_{\boldsymbol{\theta}} : [0, T] \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m \to \mathbb{R}^m$ a neural network.

There are two possible ways of training continuous-depth models: discretize-then-optimize or optimize-then-discretize (Kidger, 2021). In the former, the numerical simulation library's inherent auto-differentiation capabilities are leveraged. In the latter, the adjoint dynamics are employed to compute the gradient's loss. The following proposition 3.1 provides the adjoint method for constant lag NDDEs.

Proposition 3.1. Let us consider the Neural DDE model below, where τ is a learnable vector parameterized by some components of $\boldsymbol{\theta}$:

$$\frac{\partial \mathbf{y}(t)}{\partial t} = \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}(t-\tau)), \quad \mathbf{y}(t \leq 0) = \boldsymbol{\psi}(t), \quad 0 \leq t \leq T, \tag{3.2}$$

and the following loss function:

$$L(\mathbf{y}) = \int_0^T l(\mathbf{y}(t))\, \mathrm{d}t.$$

The gradient of the loss with respect to the parameters is given by:

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\int_0^T \boldsymbol{\lambda}(t) \left(\frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}(t-\tau))}{\partial \boldsymbol{\theta}} - \frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau)}(t))}{\partial \mathbf{y}_{(\tau)}} \frac{\partial \mathbf{y}(t-\tau)}{\partial \boldsymbol{\theta}} \right) \mathrm{d}t, \tag{3.3}$$

where $\mathbf{y}_{(\tau)}(t) \equiv \mathbf{y}(t-\tau)$ is the third set of variables $\mathbf{h}_{\boldsymbol{\theta}}\left(t, \mathbf{y}, \mathbf{y}_{(\tau)}\right)$ depends on. The adjoint dynamics $\boldsymbol{\lambda}(t)$ are given by another DDE:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{\lambda}(t)}{\mathrm{d}t} &= \frac{\partial l(\mathbf{y}(t))}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t) \frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau)}(t))}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t+\tau) \frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t+\tau, \mathbf{y}(t+\tau), \mathbf{y}_{(\tau)}(t+\tau))}{\partial \mathbf{y}_{(\tau)}}, \\ \boldsymbol{\lambda}(t \geq T) &= \mathbf{0}. \end{aligned} \tag{3.4}$$

The derivation of Proposition 3.1 is in Appendix B. Algorithm 1 below outlines the training procedure for a Neural DDE with multiple learnable constant delays, defined for convenience as $\mathbf{y}_{(\tau_i)}(t) = \mathbf{y}(t-\tau_i)$.

The developments are packaged in a user-friendly API, developing a numerically robust DDE solver, and implementing the adjoint method in the `torchdde` package. These advancements allow a seamless integration of DDEs for future users, enhancing reproducibility. Computational details and benchmarks are reported in Appendix A.2.

Algorithm 1 Training a Neural DDE with learnable delays with the adjoint method.

Require: Dataset of one trajectory $\mathcal{D} = \{(t_0, \mathbf{y}_0), \ldots, (t_T, \mathbf{y}_T)\}$.
Require: Initialized model $\mathbf{h}_{\boldsymbol{\theta}}$.
Require: Initialized positive delays $\tau_1, \ldots, \tau_n$, handled as additional entries in the parameters vector $\boldsymbol{\theta}$.
1: **for** N_{epochs} **do**
2: Set $\tau_{\max} = \max\{\tau_1, \ldots, \tau_n\}$
3: Create history function interpolation ψ with data from $\mathcal{D}$ such that $t \leq t_0 + \tau_{\max}$.
4: Solve DDE dynamics:
5: $$\begin{cases} \dfrac{\mathrm{d}\mathbf{y}(t)}{\mathrm{d}t} = \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau_1)}(t), \ldots, \mathbf{y}_{(\tau_n)}(t)) \\ \mathbf{y}(t \leq t_0 + \tau_{\max}) = \boldsymbol{\psi}(t) \end{cases}$$

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

6: Compute loss $L(\mathbf{y}) = \int_{t_0+\tau_{\max}}^{t_T} l(\mathbf{y}(t))\,\mathrm{d}t$

7: Solve Adjoint dynamics:

8:
$$\begin{cases} \dot{\boldsymbol{\lambda}}(t) = \dfrac{\partial l(\mathbf{y}(t))}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t)\dfrac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau_1)}(t), \ldots, \mathbf{y}_{(\tau_n)}(t))}{\partial \mathbf{y}} \\ \qquad\qquad - \displaystyle\sum_{i=1}^{n} \boldsymbol{\lambda}(t+\tau_i)\dfrac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t+\tau_i, \mathbf{y}(t+\tau_i), \mathbf{y}_{(\tau_1)}(t+\tau_i), \ldots, \mathbf{y}_{(\tau_n)}(t+\tau_i))}{\partial \mathbf{y}_{(\tau_i)}} \\ \boldsymbol{\lambda}(t \geq t_T) = \mathbf{0}. \end{cases}$$

9: Compute $\frac{\partial L}{\partial \boldsymbol{\theta}}$:

10:
$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\int_{t_0}^{T} \boldsymbol{\lambda}(t) \left(\frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau_1)}(t), \ldots, \mathbf{y}_{(\tau_n)}(t))}{\partial \boldsymbol{\theta}} - \sum_{i=1}^{n} \frac{\partial \mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}_{(\tau_1)}(t), \ldots, \mathbf{y}_{(\tau_n)}(t))}{\partial \mathbf{y}_{(\tau_i)}} \frac{\partial \mathbf{y}(t-\tau_i)}{\partial \boldsymbol{\theta}} \right) \mathrm{d}t$$

11: Update $\boldsymbol{\theta}$

12: **end for**

3.2. On the importance of relevant delays

Learning the delays τ_i within NDDEs is crucial for accurately modeling partially observed dynamics, and therefore, delays need to be adapted during training. To illustrate the impact of the relevant delays, we evaluate the information that past observations share with the current dynamics of the observable, and how this varies with the delay. To this end, let us consider a dynamical system evolving on a compact smooth manifold $\mathcal{S} \subset \mathbb{R}^m$ described by a simple 2-delay dynamical system, assumed to be an attractor, and let us consider a C^2 observable function $y : \mathcal{S} \to \mathbb{R}$

$$\begin{aligned} &y(t+\Delta t) = \cos(y(t-\tau_1))\sin(y(t-\tau_2)) - \alpha\,\mathrm{sinc}(3y(t-\tau_1)) + \alpha\cos(y(t-\tau_2)), \\ &y(t<0) = \psi(t), \end{aligned} \tag{3.5}$$

with $\alpha = 0.2$, $\tau_1 = p_1^\star \Delta t$, $\tau_2 = p_2^\star \Delta t$, $p_1^\star = 250$, $p_2^\star = 400$ and $\mathrm{sinc}(x) := \sin(x)/x$ for $x \neq 0$, $\mathrm{sinc}(0) := 1$.

The Takens' theorem (Takens, 1981) rigorously discusses the conditions under which a delayed vector of real-valued observables $(y(\mathbf{u}(t)), y(\mathbf{u}(t-\tau)), \ldots, y(\mathbf{u}(t-n\tau)))$, $n \in \mathbb{N}$, defines an embedding, a smooth diffeomorphism onto its image. It guarantees a topological equivalence between the original dynamical system and the one constructed from the memory of the observable. The dynamics of the system can then be reformulated on the set $(y(\mathbf{u}(t)), y(\mathbf{u}(t-\tau)), \ldots, y(\mathbf{u}(t-n\tau)))$. However, Takens' theorem, later extended by Sauer et al. (1991), provides a sufficient condition for reconstruction, but does not specify how to choose the time delay τ. From a mathematical viewpoint, the delay could be arbitrary, besides some specific values excluded by the theorem. In practice, its value can strongly condition a successful embedding. If too small, entries of the delay vector data are too similar; if too large, the entries tend to be completely uncorrelated and cannot be numerically linked to a consistent dynamical system (Kantz and Schreiber, 2003).

We here illustrate the impact of suitable delays on the relevance of the information available to inform the future evolution of the observable. The relevance of the delays $\{\tau_1, \tau_2\}$ for informing $y(t+\Delta t)$ is assessed in terms of the mutual information between $(y(t-\tau_1), y(t-\tau_2))$ and $y(t)$, as suggested in Frase and Swinney (1986). The mutual information $I(X, Y)$ between two random variables X and Y quantifies the amount of information observing a variable brings about the other one. By considering the problem in Equation (3.5), we estimate the information shared between the current observations $y(t)$ and their past

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

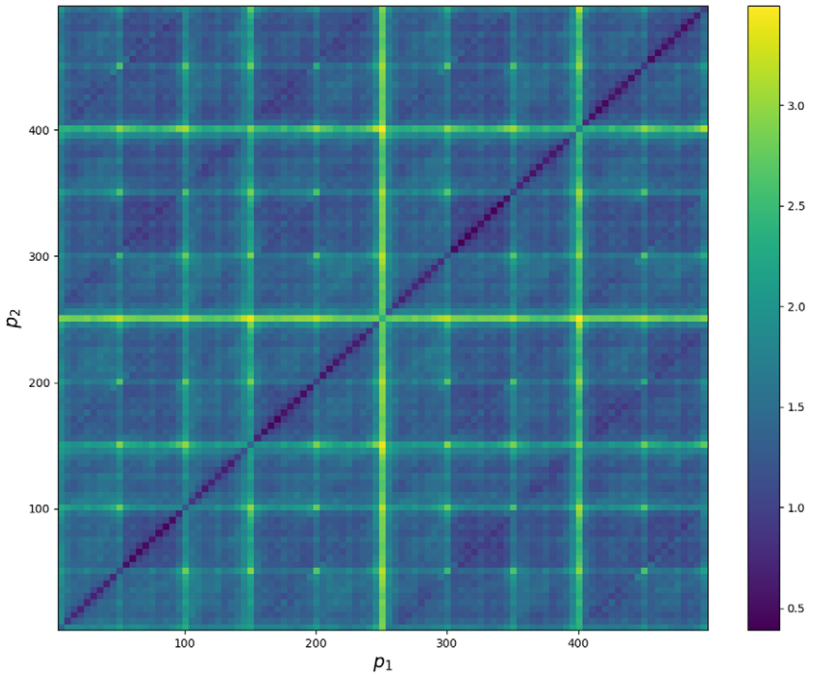


Figure 2. *$\{\tau_1 = p_1 \Delta t, \tau_2 = p_2 \Delta t\}$-map of delayed mutual information, $I((y(t-\tau_1), y(t-\tau_2)), y(t))$. The maximum is exhibited at $(p_1, p_2) = (250,\ 400)$ and $(400,\ 250)$, in agreement with $p_1^\star = 250$, $p_2^\star = 400$.*

values $y(t-\tau_1)$, $y(t-\tau_2)$ and is shown in Figure 2 as a 2-D map in terms of p_1 and p_2. The map is symmetric, as expected since $I((y(t-\tau_1), y(t-\tau_2)), y(t)) = I((y(t-\tau_2), y(t-\tau_1)), y(t))$. It can be seen that the amount of information shared between the current observation and a delay vector of the observable significantly varies with the delays and indeed reaches a maximum for $\tau_1 = p_1^\star \Delta t$, $\tau_2 = p_2^\star \Delta t$. The ability of the present Neural DDE method to learn the delays, in addition to the model $\mathbf{h}_\theta$, is thus key to its performance and wide applicability.

4. Experiments

We now demonstrate the present approach and its capability to jointly learn the delays in addition to a model of the dynamics. Several experiments are considered, ranging from synthetic low-dimensional dynamical systems to time-series predictions in an experimental configuration. In addition to accurately learning the dynamics, the relevance of being able to learn suitable delays is also demonstrated.

4.1. Dynamical systems

In the following, the dynamical systems considered in this article are presented. All datasets are divided into training, validation, and test sets with proportions of 70%, 10%, and 20%, respectively, and the Dormand–Prince (Dopri5) solver was used for data generation and training (Dormand and Prince, 1980). Table 1 outlines the number of delays employed in NDDE for each experiment.

4.1.1. Population dynamics model

As a first example, we consider the dynamics of a scalar-valued system used to model population dynamics in biology (Arino et al., 2009; Banks et al., 2017). Such a system is formulated through the following DDE:

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = u(t)(1 - u(t-\tau)), \qquad u(t \leq 0) = \psi(t), \tag{4.1}$$

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

Table 1. *Number of delays used in NDDE for each experiment*

	KS	Cavity	Brusselator
NDDE	5	1	2

where we integrate from $t \in [0,10]$, $\tau = 1$, $\psi(t) = u_0$, u_0 is sampled from the uniform distribution $\mathcal{U}(2.0,3.0)$ and 256 trajectories were generated.

4.1.2. Brusselator

A second experiment showcases how NDDEs can effectively model partially observed systems with past state values. We consider the 2-species Belousov–Zhabotinsky kinetic equation (Belousov, 1959; Zhabotinskii, 1964) that can be modeled by the so-called Brusselator system:

$$\begin{cases} \dfrac{\mathrm{d}u_1(t)}{\mathrm{d}t} = A - Bu_1 - u_1 + u_1^2 u_2, \\ \dfrac{\mathrm{d}u_2(t)}{\mathrm{d}t} = Bu_1 - u_1^2 u_2, \end{cases} \tag{4.2}$$

with $u_1(t)$ and $u_2(t)$ the two species concentrations at a given time. We integrate in the time domain $t \in [0,25]$. The initial condition u_1 is sampled from the uniform distribution $\mathcal{U}(0,2.0)$ while $u_2 = 0.0$ and 1024 trajectories were generated. We set ourselves in the partially observable case, where we only have access to the dynamics of u_1, that is, $y(t) \equiv u_1(t)$, and wish to reconstruct the whole dynamics.

4.1.3. Kuramoto–Sivashinsky (KS) system

This model was originally derived to describe the complex dynamics of flames in a combustion. The Kuramoto–Sivashinsky system is described as:

$$\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^4 u}{\partial x^4} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0. \tag{4.3}$$

The system is integrated over the time domain $t \in [0,30]$ and its spatial domain $D_x = [0,22]$ is discretized into 128 points. A total of 2048 trajectories were generated from the initial condition samples. To put ourselves in the partially observed setting, we choose to observe the solution in time at p locations uniformly spread across the spatial domain (we retain $p = 5$).

4.1.4. Incompressible open cavity flow

As an experimental demonstrator, we consider the modeling based on time-series derived from wind tunnel experiments of an open cavity flow sketched in Figure 3. The facility and the details of the experimental setup are described in Tuerke et al. (2020). Open cavity flows have attracted numerous research efforts in the last few decades due to the interesting dynamics at work. The flow is characterized by an impinging shear layer activating a centrifugal instability in a cavity. This interplay, reminiscent of the feedback acoustic mechanisms described in Rossiter (1964)), leads to a self-sustained oscillation. A broad range of dynamics is observed, ranging from limit cycles to toroidal and chaotic dynamics. The system is observed through a sensor in P measuring the local pressure fluctuations, hence $y(t) \equiv P(t; \mathbf{x}_P)$. The data are obtained for a Reynolds number $Re = 9190$ based on the length L of the cavity.

4.2. Results

We now assess the performance of the models with their ability to predict future measurements of a partially observed system. In this study, LSTM, NODE, ANODE, Latent ODE, and NDDE were selected

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

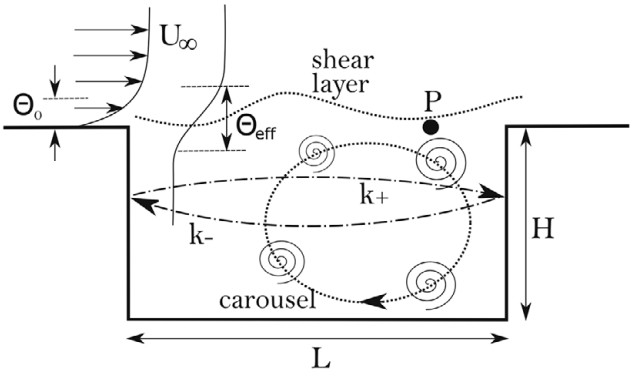


Figure 3. *Sketch of open cavity flow taken from (Tuerke et al., 2020). A data-acquiring sensor is placed in P. The cavity has a length L and depth H. The incoming laminar boundary layer flow is characterized by the freestream velocity U_∞ and the momentum thickness Θ_0.*

Table 2. *Model performance (MSE) over the test set in each experiment averaged over five runs*

	LSTM	NODE	ANODE	Latent ODE	NDDE
Brusselator	$\mathbf{0.0051 \pm 0.0031}$	0.77 ± 0.00080	$\mathbf{0.0050 \pm 0.0050}$	0.014 ± 0.0076	0.016 ± 0.0076
KS	0.77 ± 0.041	0.71 ± 0.10	0.55 ± 0.027	0.43 ± 0.07	$\mathbf{0.28 \pm 0.024}$
Cavity	0.75 ± 0.46	0.96 ± 0.0011	0.65 ± 0.0090	0.25 ± 0.14	$\mathbf{0.13 \pm 0.012}$

for comparison. Table 2 displays the test MSE loss over each experiment. Appendix D goes into more detail about each model's architecture and the training and testing procedure. Every model incorporates a form of "memory" into its architecture, with the exception of NODE. While LSTM and Latent ODE utilize hidden units and ANODE employs its augmented state $\mathbf{a}(t)$, NDDE leverages past observations such as $\mathbf{y}(t-\tau)$. In all subsequent figures, the y-axis $\mathbf{y}(t)$ represents our observables (introduced for each system in Section 4.1), defined as $\mathbf{y}(t) = y(\mathbf{u}(t))$.

4.2.1. Population dynamics model

Figures 4 and 5, respectively, depict the model's robust convergence to accurate dynamics and the evolution of the delay τ during training over many seeds, showcasing a consequence of Takens' theorem (Takens, 1981). Using a delay-coordinate map, one can indeed construct a diffeomorphic shadow manifold M' from univariate observations of the original system in the generic sense. We here observe the whole state, so that $y(t) \equiv u(t)$, and the delay coordinate map is then in terms of $(y(t), y(t-\tau))$, with $\tau > 0$. The result of Figure 5 shows that the learned delay can lie between about 0.8 and 1.4. In classical approaches (see Tan et al. (2023), the selected delay with Takens' theorem for State Space Reconstruction is typically chosen close to the minimum delayed mutual information of the time series.

4.2.2. Brusselator

In the case of this stiff and periodic dynamics, all models demonstrate satisfactory performance except NODE, which essentially predicts a mean trajectory thus highlighting the importance of incorporating memory terms for an accurate prediction. Remarkably, both LSTM and ANODE perform equally well, with NDDE and Latent ODE slightly trailing by a narrow margin as shown in Figure 6. In addition to evaluating the MSE loss performance, Figure 7 demonstrates the stability of each trained model on the Brusselator system over an extended period. After training within a specific time interval, we lengthened

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

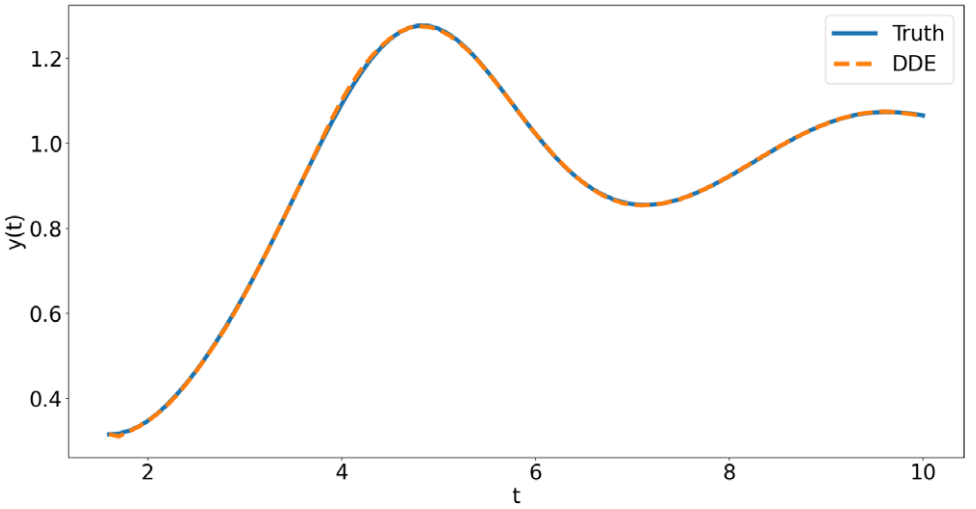


Figure 4. *The prediction of the DDE model is seen to accurately match the true evolution.*

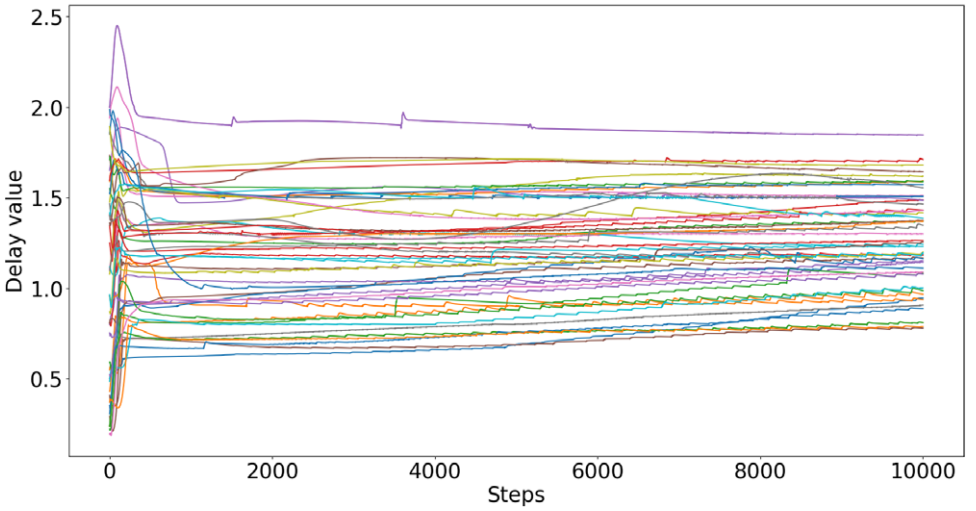


Figure 5. *Toy dataset delay evolution during training for several initial conditions.*

the integration period to five times the original training duration to assess the models' performance. It is observed that NDDE, along with LSTM, NODE, and Latent ODE, are the only models that remain stable throughout this duration, with NDDE exhibiting the best performance over the extended horizon.

4.2.3. KS system

This numerical experiment deals with a partially observed system in a chaotic regime, observed through the solution at p locations uniformly spread across the spatial domain. Figure 8 showcases random test samples from two different initial conditions, highlighting how NDDEs outperform other models struggling with the dynamics of the selected features. In a chaotic setting, the statistics of the dynamics are often more informative than the trajectory itself. We hence focus on the probability distribution of the prediction across a large time horizon for the different p components of the observables $\{y_i\}_{i=1}^{p}$, *cf.*

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

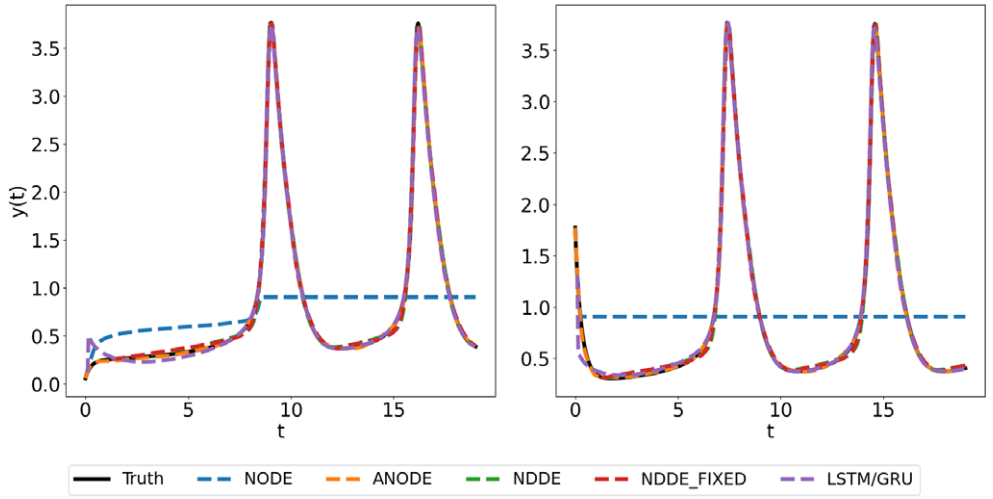


Figure 6. *Examples of prediction of the Brusselator dynamics in terms of $y(t) \equiv u_1(t)$, for initial conditions sampled at random.*

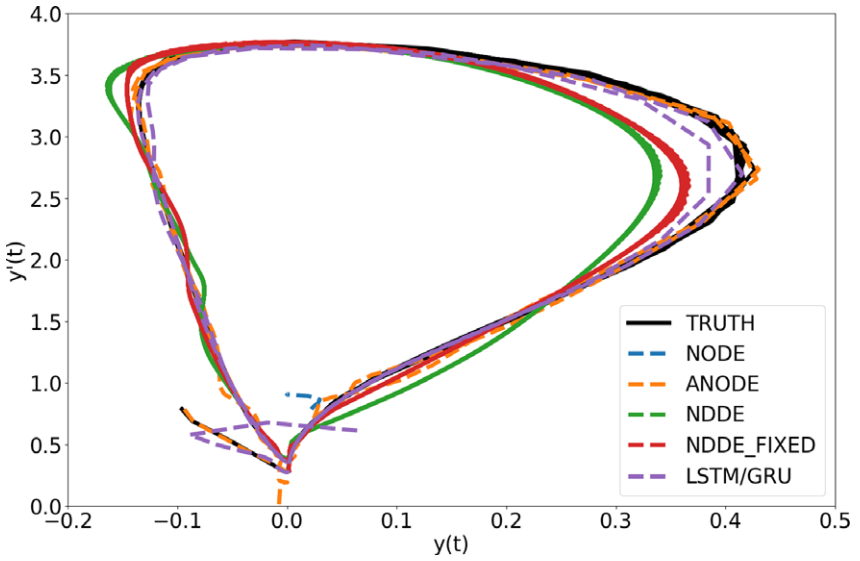


Figure 7. *Phase-portrait of $(y(t), \dot{y}(t))$. Long-term behavior of each trained model for the Brusselator system.*

Figure 9. It is seen that NDDE again outperforms the other approaches, with predictions that are statistically closer to the ground truth. This is further supported by the evolution of the maximum Lyapunov exponent (MLE) of the resulting models. Table 3 displays the MLE estimates for each model, showing that the Neural DDE with learnable delays closely aligns with the ground truth compared to other models.

4.2.4. Cavity

In this experimental fluid flow configuration, the NDDE formulation again outperforms other models, as illustrated in Figure 10 where the predictions are plotted for two test samples. The benefit of a time-delay

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

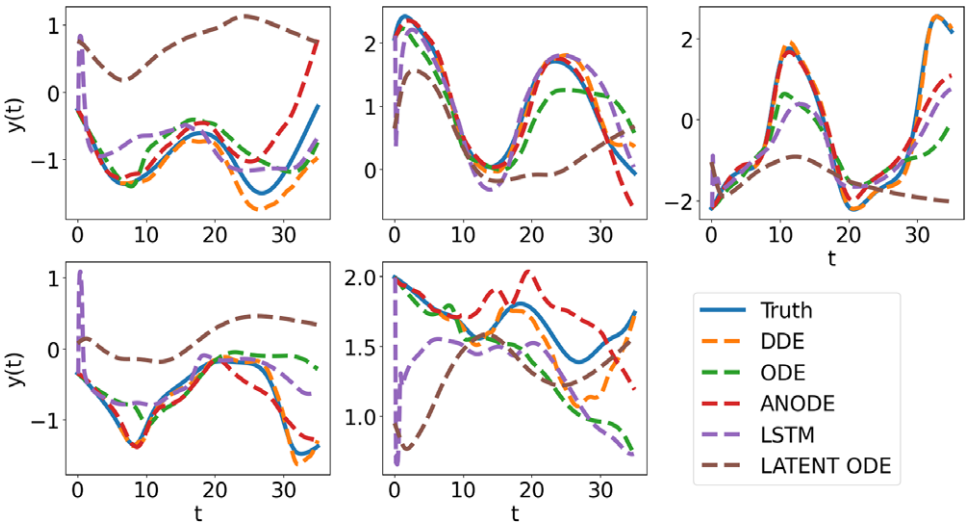


Figure 8. *Prediction of the KS system from a test sample for different models.*

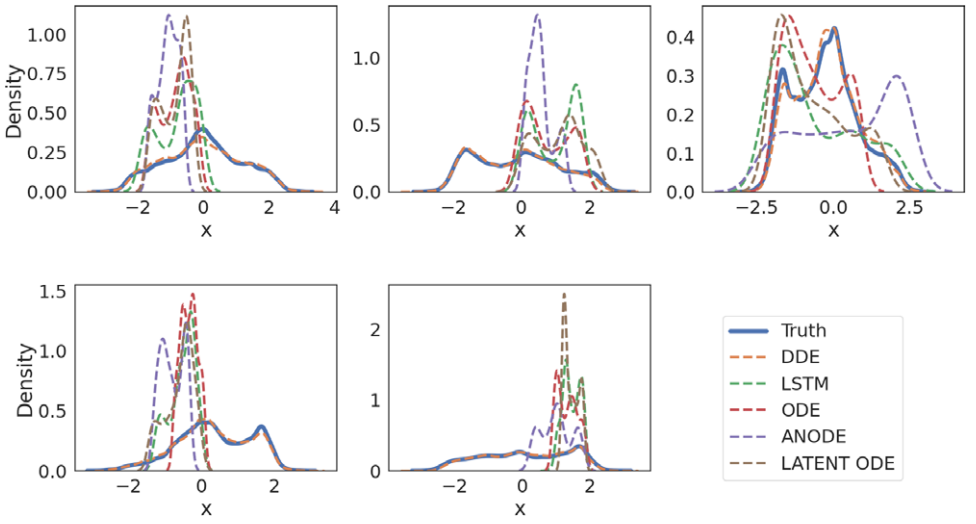


Figure 9. *Probability density functions of the predictions $y_i(t)$ over the time horizon and across several initial conditions for different models. KS system.*

Table 3. *Estimation of the maximum Lyapunov exponent $\lambda_{\max}$ for the KS system based on the generated trajectories from the test set for each model*

	Ground truth	NDDE	NODE	ANODE	Latent ODE
$\lambda_{\max}$	0.129	**0.128**	0.097	0.120	0.035

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

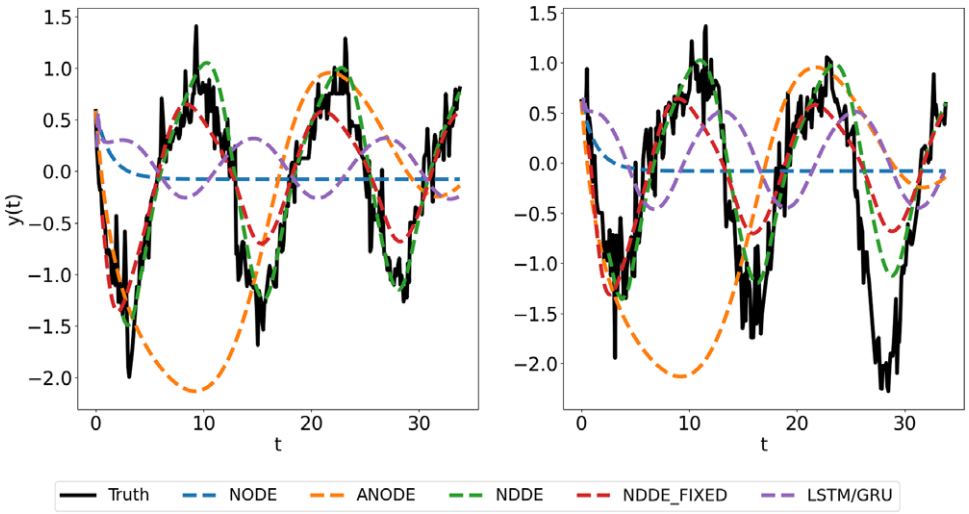


Figure 10. *Prediction of the cavity observables from different models for two test samples.*

model here comes from the fact that this experimental setup involves a large vortex within the cavity, effectively acting like a feedback loop and hence suitably described by governing equations involving delayed contributions. Latent ODE yields acceptable results compared to NODE, which only generates the system's average trajectory, while LSTM and ANODE capture oscillations, albeit occasionally in conflicting phases. These experiments demonstrate that NDDE can effectively model trajectories even in the presence of noise in the data, such as in this experimental dataset.

Since the noise in the measurements results from phenomena not related to the physical system at hand, such as, for instance, the electronic noise of the sensors, it is statistically independent from the true observables' time history. Learning to relate these past data with current measurements then quickly averages out these spurious additional dimensions introduced by the noise as the amount of observations increases. More formally, the noise $\boldsymbol{\xi}$ being assumed additive, low amplitude, and zero-mean, the expectation of the map $\mathbf{h}_{\boldsymbol{\theta}}$ writes

$$\begin{aligned}
&\mathbb{E}[\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t)+\boldsymbol{\xi}(t),\mathbf{y}(t-\tau_1)+\boldsymbol{\xi}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n)+\boldsymbol{\xi}(t-\tau_n))]\\
&\quad\approx \mathbb{E}[\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n))]+\mathbb{E}[\nabla\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n))\cdot\boldsymbol{\xi}],\\
&\quad= \mathbb{E}[\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n))]+\mathbb{E}[\nabla\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n))]\cdot\mathbb{E}[\boldsymbol{\xi}],\\
&\quad= \mathbb{E}[\mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n))],
\end{aligned}$$

so that the noise is effectively averaged out, provided the empirical mean approximates the expectation well enough.

To illustrate the benefit of learning the delays, instead of a priori setting them, the performance of NDDEs is here illustrated in both situations. A set of delays is considered, ranging from 0.2 to 1, and the mean squared error (MSE) associated with each resulting model is monitored, with and without learning the delays (see Figure 11). The results show that learned delays consistently outperform fixed delays in leading to models associated with a significantly lower MSE.

4.3. Closure modeling with the ROMs

We now revisit the Kuramoto–Sivashinsky configuration from a closure modeling viewpoint. Conceptually, one assumes an accurate model of the system to be known, but the approximate time-dependent solution to be evaluated with a reduced-order model (ROM) Galerkin approach. Specifically, the solution

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

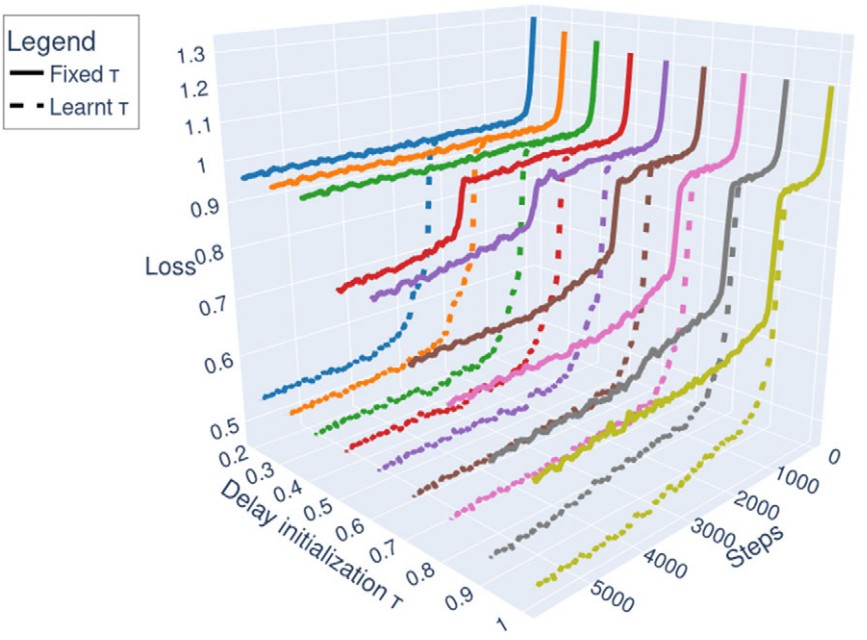


Figure 11. *Evolution of the MSE train loss of the NDDE model with constant (solid lines) and learnable delays (dashed lines) for different delay initialization values ranging from* 0.2 *to* 1.0*.*

$\mathbf{u}(t)$ is approximated as $\mathbf{u}(t) \approx \sum_{i=1}^{N_{\text{terms}}} \varphi_i(\mathbf{x}) a_i(t)$, with $\{\varphi_i(\mathbf{x}) \equiv \boldsymbol{\varphi}_i\}_{i=1}^{N_{\text{terms}}} =: \Phi$ spanning a spatial basis, so that describing the solution boils down to deriving an evolution equation for the coefficients $\mathbf{a}(t) = (a_1(t), \ldots, a_{N_{\text{terms}}}(t))$. Because the approximation basis Φ does not span the whole state space whenever $N_{\text{terms}} < \text{card}(\mathbf{u})$, the Φ-projected model describing the dynamics of the coefficients $\{a_i(t)\}_i$, resulting from a Galerkin formulation, is not Φ-invariant. To compensate for the loss of information leaking from the subspace spanned by Φ, one can complement the reduced-order model with an additional, data-based, *closure* term in the governing equations. Our experiment employs a Galerkin-projected ROM approach, with modes $\{\boldsymbol{\varphi}_i\}_i$ derived from a proper orthogonal decomposition (POD), with $N_{\text{terms}} = 4$, 8, and 10 POD modes (low, medium, and high data regimes), respectively accounting for 58%, 87%, and 94% of the system's energy (i.e., variance). In practice, the more modes selected in our POD Galerkin ROM, the smaller the correction required by the closure term needs to be.

We now consider different models for representing the closure term and compare their performance in terms of the test MSE loss in Table 4. It is seen that the DDE closure term with learnable delays consistently outperforms the ODE closure term across all data regimes (low, medium, and high). This

Table 4. *Summary of model performance metrics, that is, the test MSE loss. CD-ROM, ODE, and DDE closure models are compared across the 4, 8, and 10 POD mode settings*

Model	4 modes	8 modes	10 modes
CD-ROM	7.94 ± 0.53	0.43 ± 0.24	0.099 ± 0.019
ODE-ROM	13.61 ± 0.34	0.44 ± 0.044	0.084 ± 0.0028
DDE-ROM	$\mathbf{3.39 \pm 0.034}$	$\mathbf{0.18 \pm 0.07}$	$\mathbf{0.067 \pm 0.0078}$

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

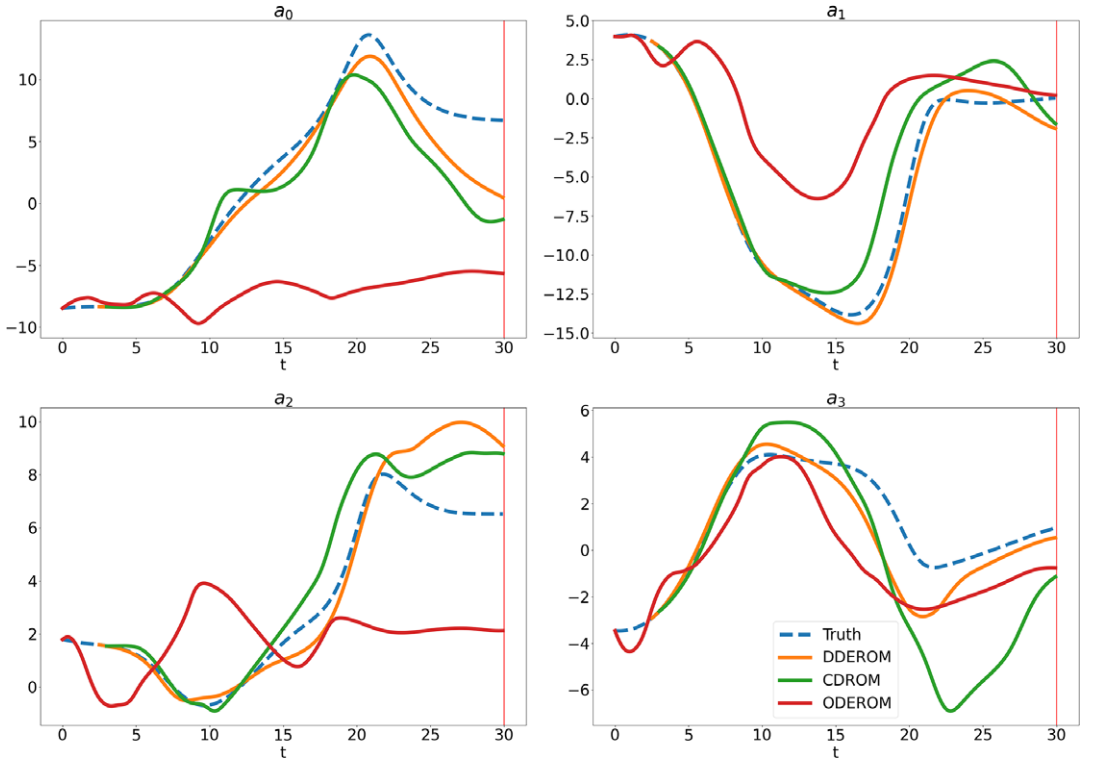


Figure 12. *Examples of predictions of POD Galerkin ROM (4 modes).*

performance difference becomes especially pronounced in the low data regime of the four-mode POD Galerkin ROM, indicating that the Neural ODE closures are not well-suited for such scenarios. Figure 12, along with their respective representations in the original state space in Figure 13, showcase the prediction from two samples of initial conditions from the test dataset for the four-mode POD Galerkin ROM. As the number of modes increases, the discrepancy between ODE and DDE closures decreases quantitatively, as seen in Table 4. This trend is expected since the ROM then captures most of the system's information. By studying these specific ROMs, we can identify where ODE closure terms fall short and how incorporating past states with the DDE closure term can address ODE's deficiencies in low data scenarios. Our approach is compared with the closure model CD-ROM (Menier et al., 2023), which considers a continuous embedding of past information in the form of an exponentially decaying dynamics auxiliary term $\mathbf{z}(t)$ mimicking a memory:

$$\mathbf{z}(t) = \int_{t_0}^{t} e^{(t-\tau)\Lambda} \mathbf{y}(\tau)\, \mathrm{d}\tau \tag{4.4}$$

with Λ a diagonal matrix whose elements are associated with the relevant time scales of the system under consideration; see (Menier et al., 2023) for details. Interestingly, compared to a memoryless Markovian model such as NODE, it only improves the quality of the prediction in the low-data regime, and considering a higher-dimensional memory variable $\mathbf{z}(t)$ does not improve the learning process in high-data regimes.

Several numbers of learnable delays were considered for the NDDE approach, ranging from 1 to 3. For this system, the results were essentially unaffected by the number of delays, showing that a single delay is enough to recover most of the necessary information for a good prediction.

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

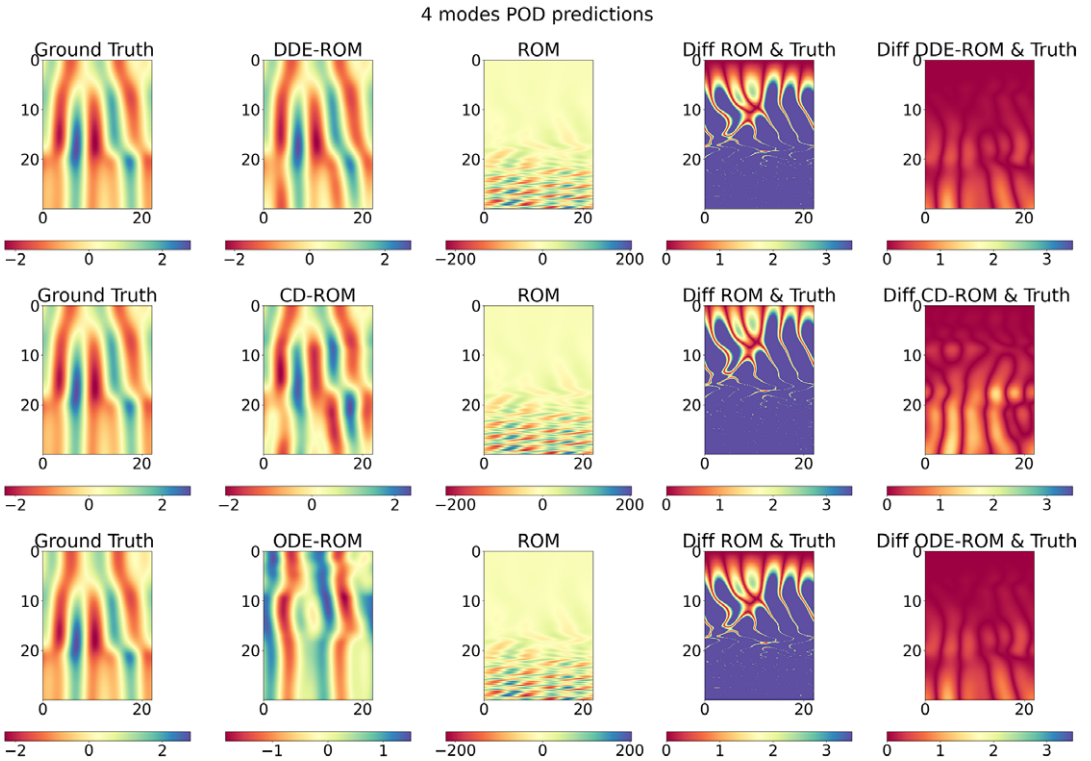


Figure 13. *Examples of predictions of POD Galerkin ROM (4 modes) with different models from KS testset reconstructed in terms of the full solution field $u(\mathbf{x},t)$ (first three columns). The two rightmost columns show the absolute value of the reconstruction error associated with each model.*

5. Conclusion

We introduced neural delay differential equations (NDDEs) as a principled and data-efficient framework for modeling partially observed dynamical systems. This approach is motivated on one hand by the Mori–Zwanzig framework first developed in the statistical physics community for accounting for the effect of unobservable quantities on observed ones; and on the other hand by the Takens' theorem. Relying on these frameworks, we leverage the link between unresolved dynamics and explicit time-delay representations via NDDEs, which allow for a continuous-time, physically interpretable representation of non-Markovian dynamics. In particular, we show that the memory term can be approximated with a finite number of learnable delays without loss of generality under smooth dynamics, providing a compact yet expressive formulation for modeling systems with memory.

Methodologically, we proposed an adjoint-based training procedure for NDDEs with learnable delays, enabling efficient end-to-end optimization over both model parameters and delay variables. The accompanying open-source implementation, `torchdde`, provides an accessible and reproducible platform for future research and applications.

Numerical and experimental validations are discussed to assess the efficacy of NDDEs across different settings, including synthetic, chaotic, and real-world noisy data, such as the Kuramoto–Sivashinsky equation, and experimental data from cavity-flow configurations. The performed experiments revealed two key insights: first, the pivotal role of a form of memory in accurately capturing dynamics; second, it was demonstrated that LSTMs' and Latent ODEs' hidden latent states or ANODEs' latent variables sometimes fall short in achieving optimal performance, emphasizing the efficacy of delayed terms as an efficient dynamics memory mechanism. In fact, across all scenarios, NDDEs consistently outperformed

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

or matched state-of-the-art continuous-depth and recurrent models while maintaining a smaller parameter footprint and improved interpretability. Finally, we revisited the Kuramoto–Sivashinsky configuration using NDDEs as a closure model of a reduced-order model based on proper orthogonal decomposition modes. With respect to current state-of-the-art closure models, the present approach again exhibits superior performance.

In conclusion, neural delay differential equations provide a theoretically grounded and computationally efficient alternative to traditional recurrent and latent-state neural models for complex non-Markovian dynamical systems. Future research will aim at further generalizing the framework; while overestimating the number of delays does not affect the final performance, rigorously determining the optimal number of delays to consider in NDDE is a challenge. A potentially useful step toward a more principled and meaningful estimation might be achieved by promoting the delayed mutual information between past data and the prediction of future observations (Fraser and Swinney, 1986).

Data availability statement. The codes associated with the examples discussed in the article can be found at the following Zenodo-GitHub repository: https://doi.org/10.5281/zenodo.20113438 (Monsel et al., 2026). The `torchdde` library for learning the delays jointly and the DDE's dynamics in these models is made open-source at https://doi.org/10.5281/zenodo.20113429 (Monsel et al., 2026).

Acknowledgments. The authors would like to thank Dr. Emmanuel Menier for helpful discussions.

Author contribution. Conceptualization: T.M., O.S., G.C., and L.M. Methodology: T.M., O.S., G.C., and L.M. Data curation: T.M. Data visualization: T.M. Writing original draft: T.M., O.S., and L.M. All authors approved the final submitted draft.

Funding statement. This work has been funded by the French National Agency for Research under project # ANR-20-CE23-0025-01.

Competing interests. The authors declare none.

Ethical standard. The research meets all ethical guidelines, including adherence to the legal requirements of the study country.

References

Achdou Y, **Bokanowski O and Lelièvre T** (2012) *Partial differential equations in finance*. In FabozziFJ (ed), *Encyclopedia of Financial Models*. Hoboken,New Jersey: John Wiley & Sons.

Arino O, **Hbid ML and Dads EA** (2009) *Delay differential equations and applications: Proceedings of the NATO advanced study institute held in Marrakech, Morocco, 9-21 September 2002.* Nato Science Series II. Netherlands: Springer .

Banks HT, **Banks JE**, **Bommarco R**, **Laubmeier AN**, **Myers NJ**, **Rundlöf M and Tillman K** (2017) Modeling bumble bee population dynamics with delay differential equations. *Ecological Modelling 351*, 14–23.

Belousov BP (1959) *A Periodic Reaction and Its Mechanism. Collection of Short Papers on Radiation Medicine for1958*. Moscow, Russia: Meditsina Publishers.

Chen TQ, **Rubanova Y**, **Bettencourt J and Duvenaud D** (2018) Neural ordinary differential equations. In Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N and Garnett R (eds), *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*, *31*, pp. 6572–6583.

Chorin A and Stinis P (2007) *Problem reduction, renormalization, and memory*, Commun. *Applied Mathematics and Computer Science 1*(1), 1–27.

Chorin AJ, **Hald OH and Kupferman R** (2002) Optimal prediction with memory. *Physica D 166*(3-4), 239–257.

Dormand JR and Prince PJ (1980) A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics 6*(1), 19–26.

Dupont E, **Doucet A and Teh YW** (2019) *Augmented neural ODEs*, Adv. *Neural Inf. Process. Syst. 32*, 3140–3150.

Epstein IR (1990) Differential delay equations in chemical kinetics: Some simple linear model systems. *The Journal of Chemical Physics 92*(3), 1702–1712.

Fraser AM and Swinney HL (1986) Independent coordinates for strange attractors from mutual information. *Physical Review A 33*, 1134–1140.

Gallage RS (2017) *Approximation of continuously distributed delay differential equations, Master's thesis*. Carbondale, IL: Southern Illinois University Carbondale.

Ghil M, **Zaliapin I and Thompson S** (2008) *A delay differential model of ENSO variability: parametric instability and the distribution of extremes*, Nonlinear Process. *Geophys. 15*(3), 417–433.

Guidorzi R (2003) *Multivariable system identification: from observations to models*. Bologna, Italy: Bononia University Press.

Gupta A and Lermusiaux PFJ (2021) Neural closure models for dynamical systems. *Proceedings of the Royal Society A 477*(2253), 20210036.

Gupta P, **Schmid P**, **Sipp D**, **Sayadi T and Rigas G** (2025) *Mori–Zwanzig latent space Koopman closure for nonlinear autoencoder*, Proc. *The Royal Society A 481*(2313), 20240259.

He KG, **Zhang X**, **Ren S and Sun J** (2016) Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural Computation 9*(8), 1735–1780.

Jaeger H and Haas H (2004) Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science 304*(5667), 78–80.

Jordan MI (1986) *Serial order: a parallel distributed processing approach. technical report*. University of California, San Diego: La Jolla (USA). Inst. for Cognitive Science.

Kantz H and Schreiber T (2003) *Nonlinear time series analysis*. Cambridge University Press.

Keane A, **Krauskopf B, and Dijkstra HA** (2019) The effect of state dependence in a delay differential equation model for the El Niño southern oscillation, El Niño southern oscillation. *Proceedings of the Royal Society A 377*(2153), 20180121.

Kidger P (2021) *On neural differential equations*. UK: Mathematical Institute, University of Oxford.

Lin YT, **Tian Y**, **Anghel M and Livescu D** (2021) Data-driven learning for the Mori–Zwanzig formalism. *SIAM Journal on Applied Dynamical Systems 20*(4), 2227–2257.

Maass W, **Natschläger T and Markram H** (2002) Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation 14*(11), 2531–2560.

Menier E, **Bucci MA**, **Yagoubi M**, **Mathelin L and Schoenauer M** (2023) *CD-ROM: Complemented deep - reduced order model*, Comput. *Methods Appl. Mech. Eng. 410*, 115985.

Monsel T, **Semeraro O**, **Mathelin L and Charpiat G** (2024) Time and state dependent neural delay differential equations. In Coelho C, Zimmering B, Costa MFP, Ferrás LL, and Niggemann O (eds.), *Proceedings of the 1st ECAI Workshop on Machine Learning Meets Differential Equations: From Theory to Applications, Proceedings of Machine Learning Research*, Vol. 255, pp. 1–20.

Monsel T, **Semeraro G**, **Charpiat O and Mathelin L** (2026) Replication data for: Neural delay differential equations: learning non-Markovian closures for partially known dynamical systems.

Monsel T, **Menier E**, **Semeraro G**, **Charpiat O and Mathelin L** (2026) TorchDDE release for: Neural delay differential equations: learning non-Markovian closures for partially known dynamical systems.

Mori H (1965) *A Continued-Fraction Representation of the Time-Correlation Functions*, Prog. *Theor. Phys. 34*(3), 399–416.

Noakes L (1991) The Takens embedding theorem. *International Journal of Bifurcation and Chaos 1*(04), 867–872.

Oprea M, **Walth M**, **Stephany R**, **Nothaft GT**, **Rodriguez-Gonzalez A and Clark W** (2023) Learning the delay using neural delay differential equations. In *AI4DifferentialEquations in Science 2023 Workshop, Proceedings of Machine Learning Research (PMLR)*.

Parish EJ and Duraisamy K (2017a) A dynamic subgrid scale model for large eddy simulations based on the Mori–Zwanzig formalism. *Physical Review Fluids 2*(1), 014604.

Parish EJ and Duraisamy K (2017b) A unified framework for multiscale modeling using the Mori–Zwanzig formalism and the variational multiscale method. arXiv:1712.09669.

Parish EJ and Duraisamy K (2017c) Non-markovian closure models for large eddy simulations using the Mori–Zwanzig formalism. *Physical Review Fluids 2*(1), 014604.

Robert CP and Casella G (1999) *Monte Carlo integration*. Springer, pp. 71–138.

J. E. Rossiter, *Wind-tunnel experiments on the flow over rectangular cavities at subsonic and transonic speeds*, Technical report, Aeronautical Research Council (Reports & Memoranda), 1964.

Roussel MR (1996) The use of delay differential equations in chemical kinetics. *The Journal of Physical Chemistry 100*(20), 8323–8330.

Rubanova Y, **Chen RTQ and Duvenaud DK** (2019) Latent ordinary differential equations for irregularly-sampled time series. In Wallach HM, Larochelle H, Beygelzimer A, Buc F, Fox EB and Garnett R (eds.), *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Red Hook, NY: Curran Associates, Inc., pp. 5320–5330.

Rumelhart DE, **Hinton GE and Williams RJ** (1986) *Learning internal representations by error propagation*. Cambridge, MA: MIT Press, pp. 318–362.

Sanderse B, **Stinis P**, **Maulik R and Ahmed SE** (2025) *Scientific machine learning for closure models in multiscale problems: a review*, Found. *Data Science 7*(1), 298–337.

Sauer T, **Yorke JA and Casdagli M** (1991) Embedology. *Journal of Statistical Physics 65*, 579–616.

Schlaginhaufen A, **Wenk P**, **Krause A, and Dörfler F** (2021) Learning stable deep dynamics models for partially observed or delayed dynamical systems, In Ranzato M, Beygelzimer A, Dauphin Y, Liang P and Vaughan JW (eds.), *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. Red Hook, NY: Curran Associates, Inc.

Soviany P, **Ionescu RT**, **Rota P and Sebe N** (2022) Curriculum learning: A survey. *International Journal of Computer Vision 130*, 1526–1565.

Stinis P (2004) *Stochastic optimal prediction for the Kuramoto–Sivashinsky equation*, Multiscale Model. *Simul. 2*(4), 580–612.

Stinis P (2007) *Higher order Mori–Zwanzig models for the Euler equations*, Multiscale Model. *Simul. 6*(3), 741–760.

Takens F (1981) Detecting strange attractors in turbulence. In RandD and YoungL-S (ed.), *Dynamical Systems and Turbulence, Warwick 1980*. Berlin Heidelberg: Springer, pp. 366–381.

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

Tan E, **Algar S**, **Corrêa D**, **Small M**, **Stemler T and Walker D** (2023) Selecting embedding delays: An overview of embedding techniques and a new method using persistent homology. *Chaos 33*(3), 032101.

Tuerke F, **Lusseyran F**, **Sciamarella D**, **Pastur L and Artana G** (2020) Nonlinear delayed feedback model for incompressible open cavity flow. *Physical Review Fluids 5*(2), 024401.

Vaswani A, **Shazeer N**, **Parmar N**, **Uszkoreit J**, **Jones L**, **Gomez A**, **Kaiser L and Polosukhin I** (2017) Attention is all you need. In Guyon I, von Luxburg U, Bengio S, Wallach H, Fergus R, Vishwanathan SVN and Garnett R (eds.), *Advances in Neural Information Processing Systems (NeurIPS 2017), Vol. 30*. Red Hook, NY: Curran Associates, Inc., pp. 5998–6008.

Vlachas P-R, **Pathak J**, **Hunt BR**, **Sapsis TP**, **Girvan M**, **Ott E and Koumoutsakos P** (2020) Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks 126*, 191–217.

Zappala E, **de O. Fonseca AH**, **Moberly AH**, **Higley MJ**, **Abdallah C**, **Cardin JA and van Dijk D** (2023) *Neural integro-differential equations*, Proc. *AAAI Conf. Artif. Intell. 37*(9), 11104–11112.

A. M. Zhabotinskii, Periodic course of the oxidation of malonic acid in a solution (studies on the kinetics of Belousov's reaction). *Biofizika. 9* (1964), 306–311.

Zhu Q, **Guo Y and Lin W** (2021) Neural Delay Differential Equations, The Ninth International Conference on Learning Representations.

Zhu Q, **Guo Y and Lin W** (2023) Neural delay differential equations: System reconstruction and image classification, arXiv 2304.05310.

Zhu Y (2019) *Mori–zwanzig equation: Theory and applications, Ph.D. thesis*. Santa Cruz, Santa Cruz, CA: University of California.

Zwanzig R (1966) Approximate eigenfunctions of the Liouville operator in classical many-body systems. *Physics Review 144*, 170–177.

Zwanzig R, **Nordholm KSJ and Mitchell WC** (1972) Memory effects in irreversible thermodynamics: Corrected derivation of transport equations. *Physical Review A 5*, 2680–2682.

A. Neural IDE and Neural DDE Benchmark

First, let us compare both Neural IDE and Neural DDE analytically, where any function $\mathbf{f}_{\theta}$ here denotes a parameterized network:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = M_{\theta}(\mathbf{y}(t)) + \int_0^t K_{\theta}(\mathbf{y}(t-s),\ s)\mathrm{d}s, \tag{A.1}$$

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{h}_{\theta_1}(\mathbf{y}(t)) + \mathbf{h}_{\theta_2}(t, \mathbf{y}(t),\ \mathbf{y}(t-\tau_1), \ldots, \mathbf{y}(t-\tau_n)). \tag{A.2}$$

The second term on the right-hand side of Equation (A.1) is much more computationally involved than the second term on the right-hand side (RHS) of Equation (A.2). Indeed, Equation (A.2) only needs two function evaluations to evaluate its right-hand side, while, on the other hand, the number of function evaluations required to integrate Equation (A.1) increases as t grows to get a correct estimate of the integral term. For a more theoretical examination of computational complexity, one can refer to Appendix A.6 of Monsel et al. (2024).

A.1. Computation time

Figure A1 compares the computation time of a forward pass between a Neural IDE and Neural DDE, as considered above, of similar size (roughly 500 parameters), with the following setup:

- Runge–Kutta 4 (RK4) solver with a timestep of $dt = 0.1$,
- Batch size of 128,
- Neural DDE uses 5 delays,
- Different numbers of observations are considered: {5,10,50,100},
- The upper integration bound is varied from $t = 0.2$ to $t = 2.0$.

This benchmark clearly shows how expensive Neural IDE is compared to NDDE. NDDEs integration is at least an order of magnitude faster. Note that in Figure A1, the notation "#i" refers to the number of observables $y_i(\mathbf{u}, t)$.

A.2. `torchdde`'s memory and time benchmark

In this subsection, we provide time and memory benchmarks on some of `torchdde`'s solvers. In order to compare both training methods of optimize-then-discretize (i.e., the adjoint method) and discretize-then-optimize (i.e., regular backpropagation), we present the Brusselator's experiment time duration and memory usage for various solvers during training (with a batch size of 1024) in Tables A1 and A2, respectively. The adjoint method is slower (by a small factor) and requires less memory than the regular backpropagation. These results are consistent with NODE's examination of the adjoint method and conventional backpropagation tradeoffs (Chen et al., 2018).

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

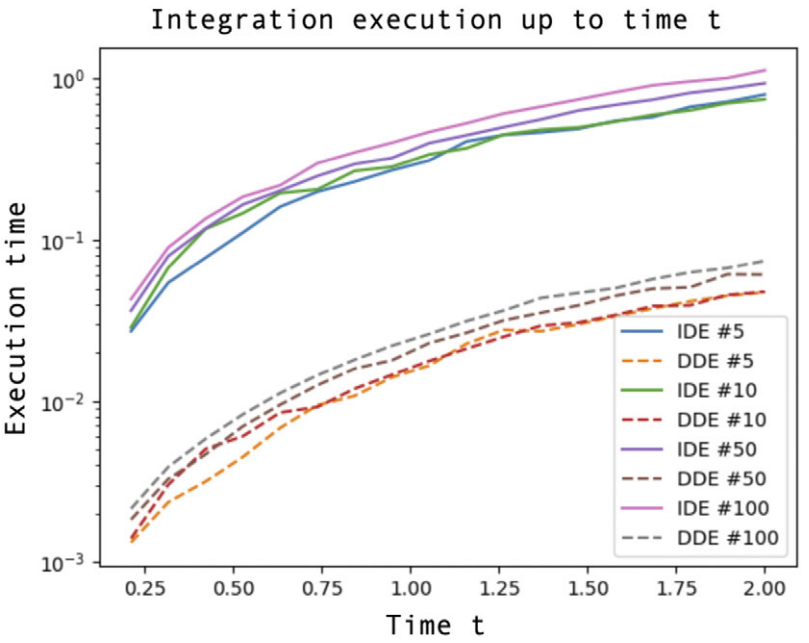


Figure A1. *Computation time of the forward pass (averaged over 5 runs) as a function of the size of the horizon for different numbers of observations (5, 10, 50, 100).*

Table A1. *Clock time (s) per batch*

	Adjoint	Backpropagation
RK4	4.8 ± 0.23	1.89 ± 0.09
RK2	2.4 ± 0.005	0.90 ± 0.005
Euler	1.5 ± 0.01	0.47 ± 0.003

Table A2. *GPU consumption (Gb ± Mb) per batch*

	Adjoint	Backpropagation
RK4	2.2 ± 18	2.87 ± 4
RK2	2.15 ± 20	2.48 ± 3
Euler	2.09 ± 15	2.264 ± 9

Figures A2 and A3 compares time and memory consumption for a forward of a Neural DDE with a varying number of delays, respectively, each having ~28,000 parameters. We use the following setup:

- Use an RK4 solver with a timestep of $dt = 0.1$,
- Use a batch size of 128,
- Neural DDE uses {1,3,5,10,20} delays,
- The number of observations is $p = 100$,
- The upper integration bound is varied from $t = 0.2$ to $t = 5.0$,
- The notation "#i" in the figure refers to the number of delays used in the Neural DDE.

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

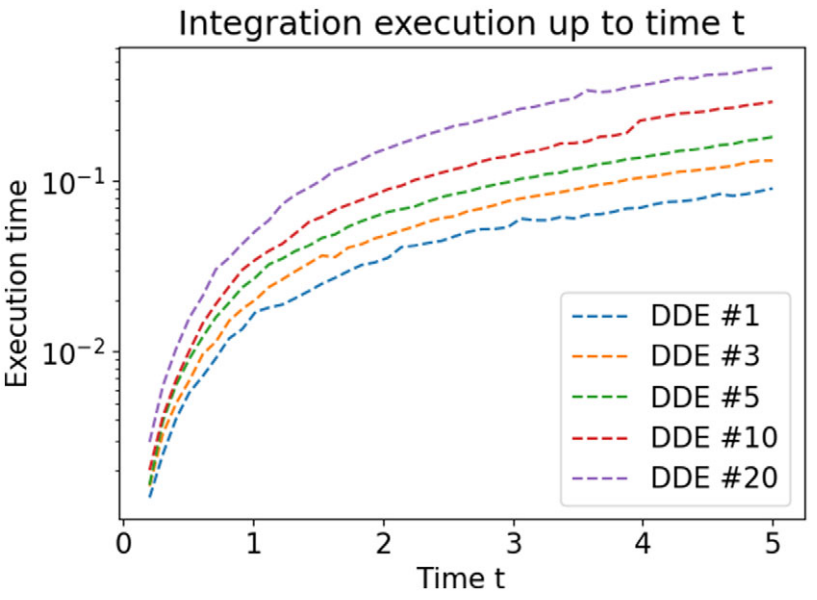


Figure A2. *Time duration of forward pass averaged over 5 runs.*

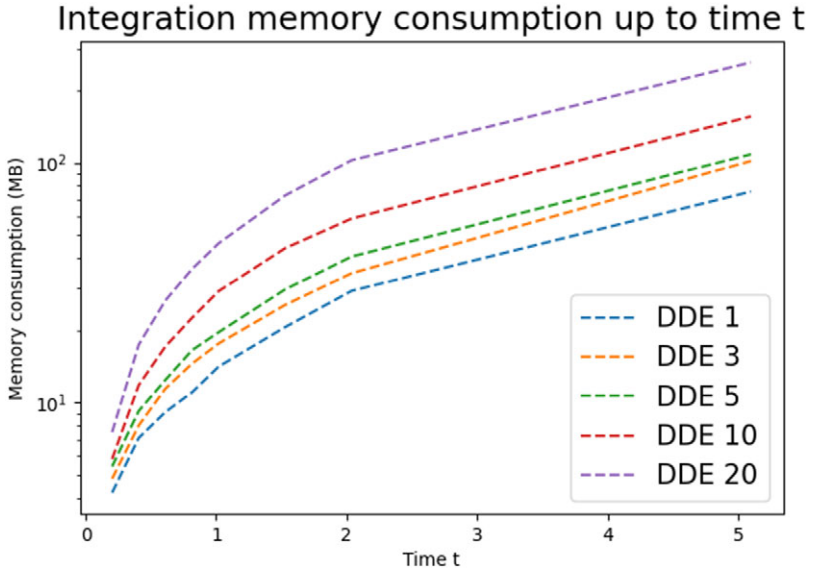


Figure A3. *Memory consumption of the forward pass averaged over 5 runs.*

B. Derivation of Proposition 3.1

We want to solve the constrained optimization problem introduced in Section 3.1. The delays are part of the learning and are hence a subset of the parameter vector $\boldsymbol{\theta}$ to be optimized. The derivation is first presented for a single delay before being extended to the multiple delay situation below. For convenience, we denote $\mathbf{y}_{(\tau)}(t)$ the third set of variables that $\mathbf{h}_{\boldsymbol{\theta}}(t, \mathbf{y}(t), \mathbf{y}(t-\tau))$ depend on, and with a dot, the time derivative $\dot{\mathbf{z}}(t)$ of any quantity $\mathbf{z}(t)$. To recall, we consider the following minimization problem:

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

$$\begin{aligned} &\min_{\boldsymbol{\theta}} L(\mathbf{y}),\\ \text{s.t.}\quad &L(\mathbf{y}) = \int_{t_0}^{T} l(\mathbf{y}(t))\mathrm{d}t,\\ &\frac{\mathrm{d}\mathbf{y}(t)}{\mathrm{d}t} - \mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}(t-\tau)) = \mathbf{0},\\ &\mathbf{y}(t \le t_0) = \boldsymbol{\psi}(t). \end{aligned} \tag{B.1}$$

We consider the following Lagrangian $\mathscr{L}$:

$$\mathscr{L} := L(\mathbf{y}) + \int_{t_0}^{T} \boldsymbol{\lambda}(t)\left(\frac{\mathrm{d}\mathbf{y}(t)}{\mathrm{d}t} - \mathbf{h}_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t))\right)\mathrm{d}t. \tag{B.2}$$

We use $\mathbf{y}(t-\tau) = \mathbf{y}_{(\tau)}(t)$ wherever convenient. At optimality, the state equation is satisfied ($\mathbf{y}$ is a solution to the associated DDE), implying that $\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{\theta}}$.

Integration by parts of the expression of the Lagrangian (Eq. (B.2)) yields:

$$\mathscr{L} = \int_{t_0}^{T} l(\mathbf{y}(t))\mathrm{d}t + [\boldsymbol{\lambda}(t)\mathbf{y}(t)]_{t_0}^{T} - \int_{t_0}^{T}\left(\dot{\boldsymbol{\lambda}}(t)\mathbf{y}(t) + \boldsymbol{\lambda}(t) f_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t))\right)\mathrm{d}t. \tag{B.3}$$

Stationarity of the Lagrangian at optimality yields $\frac{\partial \mathscr{L}}{\partial \mathbf{y}} = \mathbf{0}$ and expressed as:

$$\mathbf{0} = \int_{t_0}^{T}\left(\frac{\partial l(\mathbf{y}(t))}{\partial \mathbf{y}} - \frac{\partial \boldsymbol{\lambda}(t)}{\partial t} - \boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t))}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t))}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t-\tau)}{\partial \mathbf{y}}\right)\mathrm{d}t + \frac{\partial[\boldsymbol{\lambda}(t)\mathbf{y}(t)]_{t_0}^{T}}{\partial \mathbf{y}} \tag{B.4}$$

With the change of variable $t \to t-\tau$, the fourth term of the integrand rewrites as:

$$\begin{aligned} -\int_{t_0}^{T} \boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t-\tau)}{\partial \mathbf{y}}\mathrm{d}t &= -\int_{t_0-\tau}^{T-\tau} \boldsymbol{\lambda}(t+\tau)\frac{\partial f_{\boldsymbol{\theta}}\left(t+\tau,\mathbf{y}(t+\tau),\mathbf{y}_{(\tau)}(t+\tau)\right)}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t)}{\partial \mathbf{y}}\mathrm{d}t,\\ &= -\int_{t_0}^{T} \boldsymbol{\lambda}(t+\tau)\frac{\partial f_{\boldsymbol{\theta}}\left(t+\tau,\mathbf{y}(t+\tau),\mathbf{y}_{(\tau)}(t+\tau)\right)}{\partial \mathbf{y}_{(\tau)}}\mathrm{d}t \end{aligned} \tag{B.5}$$

where we have set $\boldsymbol{\lambda}(t) = \mathbf{0}$ whenever $t \ge T$ and recognized that $\mathbf{y}(t \le t_0) \equiv \boldsymbol{\psi}(t)$ with $\frac{\partial \boldsymbol{\psi}(t)}{\partial \mathbf{y}} = \mathbf{0}$.

The boundary terms $\frac{\partial[\boldsymbol{\lambda}(t)\mathbf{y}(t)]_{t_0}^{T}}{\partial \mathbf{y}}$ in Eq. (B.4) then vanish, and the equation is satisfied whenever the integrand is identically zero. Hence, the following adjoint equation, to be integrated backwards in time:

$$\begin{aligned} &\frac{\mathrm{d}\boldsymbol{\lambda}(t)}{\mathrm{d}t} = \frac{\partial l(\mathbf{y}(t))}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}} - \boldsymbol{\lambda}(t+\tau)\frac{\partial f_{\boldsymbol{\theta}}\left(t+\tau,\mathbf{y}(t+\tau),\mathbf{y}_{(\tau)}(t+\tau)\right)}{\partial \mathbf{y}_{(\tau)}},\\ &\boldsymbol{\lambda}(t \ge T) = \mathbf{0}. \end{aligned} \tag{B.6}$$

Now, taking the derivative of the Lagrangian with respect to $\boldsymbol{\theta}$ expresses:

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = &\int_{t_0}^{T} \frac{\partial l(\mathbf{y}(t))}{\partial \mathbf{y}}\frac{\partial \mathbf{y}(t)}{\partial \boldsymbol{\theta}}\mathrm{d}t + \boldsymbol{\lambda}(T)\frac{\partial \mathbf{y}(T)}{\partial \boldsymbol{\theta}} - \boldsymbol{\lambda}(t_0)\frac{\partial \mathbf{y}(t_0)}{\partial \boldsymbol{\theta}}\\ &+ \int_{t_0}^{T} -\dot{\boldsymbol{\lambda}}(t)\frac{\partial \mathbf{y}(t)}{\partial \boldsymbol{\theta}}\mathrm{d}t + \int_{t_0}^{T} -\boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \boldsymbol{\theta}}\mathrm{d}t\\ &+ \int_{t_0}^{T} -\boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}}\frac{\partial \mathbf{y}(t)}{\partial \boldsymbol{\theta}}\mathrm{d}t\\ &+ \int_{t_0}^{T} -\boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t-\tau)}{\partial \boldsymbol{\theta}}\mathrm{d}t. \end{aligned} \tag{B.7}$$

Since $\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} \equiv \mathbf{0}$ and owing to the adjoint dynamics in Equation (B.6), it finally comes

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\int_{t_0}^{T} \boldsymbol{\lambda}(t)\left(\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \boldsymbol{\theta}} + \frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t-\tau)}{\partial \boldsymbol{\theta}}\right)\mathrm{d}t. \tag{B.8}$$

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

B.1. Case for multiple constant delays

In the case of multiple delays τ_i, $i \in 1,\ldots,n$, the equations in Theorem 3.1 are extended as follows. For each i, we define $\mathbf{y}_{(\tau_i)}(t) \equiv \mathbf{y}(t-\tau_i)$.

The second term $\boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}}$ in the adjoint dynamics (Eq. (B.6)) is replaced by:

$$\boldsymbol{\lambda}(t)\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau_1)}(t),\ldots,\mathbf{y}_{(\tau_n)}(t)\right)}{\partial \mathbf{y}} \tag{B.9}$$

and the last term $\boldsymbol{\lambda}(t+\tau)\frac{\partial f_{\boldsymbol{\theta}}\left(t+\tau,\mathbf{y}(t+\tau),\mathbf{y}_{(\tau)}(t+\tau)\right)}{\partial \mathbf{y}_{(\tau)}}$ by the following:

$$\sum_{i=1}^{n}\boldsymbol{\lambda}(t+\tau_i)\frac{\partial f_{\boldsymbol{\theta}}\left(t+\tau_i,\mathbf{y}(t+\tau_i),\mathbf{y}_{(\tau_1)}(t+\tau_i),\ldots,\mathbf{y}_{(\tau_n)}(t+\tau_i)\right)}{\partial \mathbf{y}_{(\tau_i)}}. \tag{B.10}$$

The first term $\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \boldsymbol{\theta}}$ in the gradient's Equation (B.8) becomes

$$\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau_1)}(t),\ldots,\mathbf{y}_{(\tau_n)}(t)\right)}{\partial \boldsymbol{\theta}} \tag{B.11}$$

and $\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau)}(t)\right)}{\partial \mathbf{y}_{(\tau)}}\frac{\partial \mathbf{y}(t-\tau)}{\partial \boldsymbol{\theta}}$ is now:

$$\sum_{i=1}^{n}\frac{\partial f_{\boldsymbol{\theta}}\left(t,\mathbf{y}(t),\mathbf{y}_{(\tau_1)}(t),\ldots,\mathbf{y}_{(\tau_n)}(t)\right)}{\partial \mathbf{y}_{(\tau_i)}}\frac{\partial \mathbf{y}(t-\tau_i)}{\partial \boldsymbol{\theta}}. \tag{B.12}$$

C. Derivation of Proposition 2.1

Let us start by stating Takens' theorem as expressed by Takens (1981)), Noakes (1991)):

Theorem C.1 (Takens' embedding theorem). Let $\mathcal{M}$ be a compact space. There is an open dense subset $\mathcal{D}$ of $\mathrm{Diff}(\mathcal{M}) \times C^2(\mathcal{M},\mathbb{R})$, with $\mathrm{Diff}(\mathcal{M})$ the diffeomorphism group of $\mathcal{M}$, with the property that the Takens map

$$\mathfrak{y} : \mathcal{M} \to \mathbb{R}^{2m+1}$$

given by $\mathfrak{y}(\mathbf{u}) = \left(\mathbf{y}(\mathbf{u}),\mathbf{y}(\phi(\mathbf{u})),\mathbf{y}(\phi\circ\phi(\mathbf{u})),\ldots,\mathbf{y}\left(\phi^{2m}(\mathbf{u})\right)\right)$ is an embedding of C^2 manifolds, where $(\phi,\mathbf{y}) \in \mathcal{D}$.

Here, ϕ stands for the operator that advances the dynamical system by a time step τ, *i.e.*, that sends $\mathbf{u}(t)$ to $\mathbf{u}(t+\tau)$, and $\mathbf{y}$ is the observable operator that sends a full state $\mathbf{u}(t)$ to actual observables $\mathbf{y}(\mathbf{u}(t))=:\mathbf{y}(t)$. Variants of this Theorem, for example, Sauer et al. (Sauer et al., 1991), include the consideration of any set of different delays τ_i instead of uniformly spaced ones. The representation $\mathfrak{y}(\mathbf{u}(t)) = (\mathbf{y}(\mathbf{u}(t)),\mathbf{y}(\mathbf{u}(t-\tau)),\mathbf{y}(\mathbf{u}(t-2\tau)),\ldots\mathbf{y}(\mathbf{u}(t-2m\tau))$ then becomes $\mathfrak{y}(\mathbf{u}(t)) = (\mathbf{y}(\mathbf{u}(t)),\mathbf{y}(\mathbf{u}(t-\tau_1)),$ $\mathbf{y}(\mathbf{u}(t-\tau_2)),\ldots\mathbf{y}(\mathbf{u}(t-\tau_{2m}))$. In the proof of Takens' theorem, m is the intrinsic dimension of the dynamical system, that is, the dimension of the manifold $\mathcal{M}$.

Now, given the full state $\mathbf{u}$ that follows the dynamics:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}), \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{C.1}$$

we use the chain rule on the observable $\mathbf{y}$:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{y}'(\mathbf{u}(t))\mathbf{f}(\mathbf{u}). \tag{C.2}$$

By applying the inverse of the delay coordinate map $\mathfrak{y}^{-1}$ from Theorem C.1, which is invertible from its image as it is an embedding, we show that the dynamics of $\mathbf{y}$ possess a DDE structure:

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = (\mathbf{y}'\times\mathbf{f})\circ\mathfrak{y}^{-1}(\mathbf{y}(t),\mathbf{y}(t-\tau_1),\ldots,\mathbf{y}(t-\tau_n)). \tag{C.3}$$

The proof is completed by choosing $\mathbf{h} = (\mathbf{y}'\times\mathbf{f})\circ\mathfrak{y}^{-1} - M$ where M is obtained by the Mori–Zwanzig formalism (Eq. (2.6)).

Note that to be able to apply Takens' theorem, we needed the step-forward operator ϕ to be a diffeomorphism, that is, the flow of the dynamical system to be smooth and smoothly invertible. Also, we used the differentiability of the observables $\mathbf{y}$ to express $\mathbf{y}'$.

D. Training hyperparameters

Our training approach incorporates a progressive strategy considered to be a curriculum learning strategy (Soviany et al., 2022). We begin by feeding the models shorter trajectory segments and gradually increase their length when the so-called *patience*

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

Table A3. *How long are the trajectory chunks given at first, and the patience used for each experiment*

	KS	Cavity	Brusselator
Length start	15%	50%	25%
Patience	40	50	20

Table A4. *Number of parameters for each experiment*

	LSTM	NODE	ANODE	Latent ODE	NDDE
Brusselator	1764	3265	3395	3666	3331
KS	18130	9029	11609	8118	19343
Cavity	2234	2209	2274	3642	2242

hyperparameter is exceeded. This process continues until we reach the desired trajectory length. Each time the trajectory length is increased, we reset the patience hyperparameter to 0. This *patience* is then incremented by 1 if the validation loss fails to decrease, and reset to 0 if the validation loss improves. This method aligns with the principles of curriculum learning, a technique that involves training machine learning models in a structured order, typically progressing from simpler to more complex examples. In our case, this translates into moving from shorter to longer trajectories. This approach aims to improve the learning process and the resulting model performance. Table A3 displays the patience parameter and the length of the trajectory considered initially. Table A4 refers to the number of training parameters of each model. The loss function used across all experiments is the MSE loss, and we employ the Adam optimizer with a weight decay of 10^{-7}. Table A5 provides the initial and final learning rates (η_i, η_f) for each experiment, which are associated with the scheduler. The scheduler is a StepLR scheduler with a gamma factor ($\gamma = \exp\left(\frac{\log(\eta_f/\eta_i)}{N}\right)$, with N the trajectory's length). The scheduler adjusts the learning rate as the trajectory length increases, allowing training to start with the initial learning rate η_i and gradually decrease to the final learning rate η_f. All continuous-time models (NODE, ANODE, Latent ODE, and NDDE) used Runge–Kutta for numerical integration. Table A6 shows the width and depth of the multi-layer perceptrons (MLPs) for

Table A5. *Initial and final learning rates for each experiment*

Experiment	η_i	η_f
Brusselator	0.001	0.0001
Cavity	0.005	0.00005
KS	0.01	0.0001
Shallow water	0.001	0.00001

Table A6. *MLP width and depth for each experiment*

	NODE/ANODE/NDDE	
	Width	Depth
Brusselator	32	4
KS	64	3
Cavity	32	3
Shallow Water	32	3

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press

Table A7. *Hidden size and number of layers for each experiment for the LSTM model*

Experiment	Hidden size	Number of layers
Brusselator	5	10
KS	25	5
Shallow water	6	10
Cavity	7	7

Table A8. *Configuration parameters for each experiment for Latent ODE*

Experiment	Width size	Depth	Hidden size	Latent size
Brusselator	16	3	16	16
KS	32	3	16	16
Cavity	16	3	8	8
Shallow water	32	3	8	8

NODE, ANODE, and NDDE across all experiments. Additionally, we provide the hidden size and number of layers for the LSTM model in Table A7. Finally, Table A8 summarizes the Latent ODE hyperparameters, where the vector field f_θ (defined in the Introduction) is an MLP with the width and depth specified in the second and third columns, respectively, the size of the latent variable z_0 in the last column, and the RNN's hidden size in the fourth column. If some models have fewer parameters compared to others, it is because we found that they provided better results with fewer. ANODE's augmented state dimension matches that of the number of delays used by NDDE displayed in Table 1.

Cite this article: Monsel T, Semeraro O, Charpiat G and Mathelin L (2026). Neural delay differential equations: learning non-Markovian closures for partially known dynamical systems. *Data-Centric Engineering*, 7, e22. doi:10.1017/dce.2026.10053

https://doi.org/10.1017/dce.2026.10053 Published online by Cambridge University Press