PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

Cite this article: Gupta A, Lermusiaux PFJ. 2021 Neural closure models for dynamical systems. *Proc. R. Soc. A* **477**: 20201004. https://doi.org/10.1098/rspa.2020.1004

Received: 7 January 2021
Accepted: 16 July 2021

Subject Areas:
computer modelling and simulation, artificial intelligence, differential equations

Keywords:
delay differential equations, reduced-order-model, turbulence closure, ecosystem modelling, data assimilation, machine learning

Author for correspondence:
Pierre F. J. Lermusiaux
e-mail: pierrel@mit.edu

Electronic supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.5545185.

THE ROYAL SOCIETY PUBLISHING

Neural closure models for dynamical systems

Abhinav Gupta and Pierre F. J. Lermusiaux

Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

AG, 0000-0002-9197-0736; PFJL, 0000-0002-1869-3883

Complex dynamical systems are used for predictions in many domains. Because of computational costs, models are truncated, coarsened or aggregated. As the neglected and unresolved terms become important, the utility of model predictions diminishes. We develop a novel, versatile and rigorous methodology to learn non-Markovian closure parametrizations for known-physics/low-fidelity models using data from high-fidelity simulations. The new *neural closure models* augment low-fidelity models with neural delay differential equations (nDDEs), motivated by the Mori–Zwanzig formulation and the inherent delays in complex dynamical systems. We demonstrate that neural closures efficiently account for truncated modes in reduced-order-models, capture the effects of subgrid-scale processes in coarse models and augment the simplification of complex biological and physical–biogeochemical models. We find that using non-Markovian over Markovian closures improves long-term prediction accuracy and requires smaller networks. We derive adjoint equations and network architectures needed to efficiently implement the new discrete and distributed nDDEs, for any time-integration schemes and allowing non-uniformly spaced temporal training data. The performance of discrete over distributed delays in closure models is explained using information theory, and we find an optimal amount of past information for a specified architecture. Finally, we analyse computational complexity and explain the limited additional cost due to neural closure models.

1. Introduction

Most models only resolve spatio-temporal scales, processes and field variables to a certain level of accuracy because of the high computational costs associated with high-fidelity simulations. Such truncation of scales, processes or variables often limit the reliability and

usefulness of simulations, especially for scientific, engineering and societal applications where longer-term model predictions are needed to guide decisions. There are many ways to truncate high-fidelity models to low-fidelity models. Examples abound and three main classes of truncations are: evolving the original dynamical system in a reduced space, e.g. using reduced-order-models (ROMs) [1,2]; coarsening the model resolution to the scales of interest [3,4]; and reducing the complexity or number of state variables, components and parametrizations [5–7]. In many applications, the neglected and unresolved terms along with their interactions with the resolved ones can become important over time, and a variety of modelling techniques have been developed to represent the missing terms. Techniques that express these missing terms as functions of modelled state variables and parameters are referred to as closure models. A main challenge is that no one closure approach to date is directly applicable to all four main classes of model truncations. Another is that closure models are only well defined for either linear problems or simple cases. Finally, they can easily become ineffective in the face of nonlinearities.

Owing to the explosion of use of a variety of machine learning methods for solving or simulating dynamical systems, a number of data-driven methods have been proposed for the closure problem. Most of them attempt to learn a neural network (NN) as the instantaneous map between the low-fidelity solution and the residual of the high- and low-fidelity solution, or their residual dynamics [8–12]. They often use recurrent networks such as long-short term memory networks (LSTMs), gated recurrent units (GRUs), etc., with justification based on the Mori–Zwanzig (MZ) formulation [13–15] and embedding theorems by Whitney [16] and Takens [17]. These approaches do not however take into account accumulation of numerical time-stepping error in the presence of NNs during training. Moreover, one requires either access to the equations describing the high-fidelity model or frequent enough and uniformly spaced high-fidelity data to be able to compute the time derivative of the state with a high level of accuracy. Such a requirement on the training data can be a luxury in a lot of scenarios. The requirement of very frequent snapshot data of the system is also true for methods which achieve model discovery using sparse-regression and provide interpretable learned models [8,18,19]. All of the above issues are addressed by using neural ordinary differential equations (nODEs; [20]) and some researchers recently used nODEs for closure modelling. Some directly learn the ODE system from high-fidelity simulation data without using the available low-fidelity models [21], which could lead to the requirement of bigger NNs. Others combine nODEs with model discovery using sparse-regression [22] or only learn the values of parameters in existing closure models [23]. Nearly all existing studies primarily only attempt to address the closure for ROMs. Finally, the existing machine learned closure models are not yet used for long-term predictions, i.e. forecasting significantly outside of the time period to which the training data belonged to.

In the present study, we propose a new neural delay differential equations (nDDEs)-based framework to learn closure parametrizations for low-fidelity models using data from high-fidelity simulations and to increase the long-term predictive capabilities of these models. Instead of using ODEs, we learn non-Markovian closure models using delay differential equations (DDEs). We base the theoretical justification for using DDEs on the MZ formulation [13–15] and the presence of inherent delays in many dynamical systems [24], especially biological systems [25,26]. We refer to the new modelling approach as *neural closure models*. We demonstrate that our methodology drastically improves the predictive capability of low-fidelity models for the main classes of model truncations. Specifically, our neural closure models efficiently account for truncated modes in ROMs, capture the effects of subgrid-scale processes in coarse models and augment the simplification of complex mathematical models. We also provide adjoint equation derivations and network architectures needed to efficiently implement nDDEs, for both discrete and distributed delays. In the case of distributed delays, we propose a novel architecture consisting of two coupled NNs, which eliminates the need for using recurrent architectures for incorporating memory. We find that our nDDE closures substantially improve nODE closures and outperform classic dynamic closures such as the Smagorinsky subgrid-scale model. We explain the better performance of nDDE closures based on information theory and the amount of past information

being included. Our first two classes of simulation experiments use the advecting shock problem governed by Burger's partial differential equation (PDE), with low-fidelity models derived either by proper-orthogonal-decomposition Galerkin projection (POD-GP) [27] or by reducing the spatial grid resolution. Our third class of experiments considers marine biological models of varying complexities [28–30] and then their physical–biogeochemical extensions, with low-fidelity models obtained by aggregation of components and other simplifications of processes and parametrizations. Finally, we analyse computational complexity and explain the limited additional computational cost due to the presence of neural closure models.

2. Closure problems

The need for closure modelling in dynamical systems arises for a variety of reasons. They often involve computational costs considerations, but also include the lack of data to resolve complex real processes, the limited understanding of fundamental dynamics, and the inherent nonlinear growth of uncertainties due to model errors and predictability limits (e.g. [31–34]). In this section, we examine three main classes of low-fidelity models that can require closure modelling.

(a) Reduced order modelling

Let us consider a nonlinear dynamical system with state variable $u \in \mathbb{R}^N$ and the full-order-model (FOM) dynamics governed by

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = Lu(t) + h(u(t)), \quad \text{with } u(0) = u_0, \tag{2.1}$$

where $L \in \mathbb{R}^{N\times N}$ is the linear, and $h(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ the nonlinear, part of the system. We are mainly interested in dynamical systems whose solution could be effectively approximated on a manifold of lower dimension, $\mathcal{V} \in \mathbb{R}^m \subset \mathbb{R}^N$ (e.g. [35]). Ideally, the dimension of this manifold is much smaller than that of the system, i.e. $m \ll N$. For the classic Galerkin-based reduced-order modelling, a linear decomposition of the form

$$u(t) \approx \bar{u} + Va, \tag{2.2}$$

is used, where $\bar{u} \in \mathbb{R}^N$ is a reference value, the columns of $V = [v_1, \ldots, v_m] \in \mathbb{R}^{N\times m}$ a basis of the m-dimensional subspace $\mathcal{V}$, and $a \in \mathbb{R}^m$ the vector of coefficients corresponding to the reduced basis. A popular choice for this basis is the POD that defines the subspace such that the manifold $\mathcal{V}$ preserves the variance of the system as much as possible when projected on V for a given m. The reference value ($\bar{u}$) is then commonly chosen as the mean of the system state, in order to prevent the first reduced coefficient from containing the majority of the energy of the system and to help stabilize the reduced system [36].

Now, substituting equation (2.2) into equation (2.1), and projecting the result on the orthonormal modes V, then we obtain the following set of ordinary differential equations for the coefficients a:

$$\frac{\mathrm{d}a}{\mathrm{d}t} = V^T LVa + V^T h(\bar{u} + Va) + V^T L\bar{u}, \quad \text{with } a(0) = V^T(u_0 - \bar{u}). \tag{2.3}$$

This m-dimensional system, with $m \ll N$, is computationally much cheaper than the original FOM equation (2.2). This method of dimensionality reduction is commonly referred to as the POD-GP method. It can suffer from a number of issues. First, the truncated modes can play an important role in the dynamical behaviour of the system, and neglecting them can thus lead to a very different forecast [1]. Second, the error in the reduced state may be simply too large for truncation, i.e. the POD reduction is not efficient. Third, if steady POD are employed, they may quickly become irrelevant for the evolving system state [35,37,38]. To address these issues, several methods try to represent the effect of the truncated modes. The most common approaches introduce a nonlinear parametrization of the coefficients (e.g. [39]) in equation (2.3), however, they are not yet generally applicable to all classes of closures.

A geometric interpretation of the goal of closure modelling for ROMs is provided in the electronic supplementary material (Section SI-1).

(b) Subgrid-scale processes

A key decision while setting up any numerical simulation is the selection of spatio-temporal resolution, which is in general limited by the computing power available. Using a coarse resolution (low-fidelity) model may however lead to a number of undesired artefacts, such as missing critical scales and processes for longer-term predictions or numerical diffusion that causes unintended or unacceptable results [40,41]. These artefacts become especially important in the case of ocean models. For example, present-day global observing systems and global model solutions only resolve open-ocean mesoscale processes ($\mathcal{O}(10-100\,\mathrm{km})$), but the submesocale (subgrid-scale) processes do have global consequences, in relation to the mechanisms of energy dissipation in the general circulation, vertical flux of material concentrations, and intermediate-scale horizontal dispersal of materials [42–44]. The neglected and unresolved scales along with their interactions with the resolved ones are then at the core of closure parametrizations. Most present oceanic models consist of a nonlinear system of PDEs, each of a nonlinear advection type, supplemented by other possible diagnostic nonlinear equations and boundary conditions. There is however no unique way of defining such parametrizations and multiple approaches such as non-dimensional analyses, physical balance hypotheses, statistical correlation constraints and other empirical methods are commonly employed to develop closure models. Similar statements can be made for atmospheric, Earth system and climate models [45]. For all these applications, a general approach for subgrid-scale closures would thus be most useful.

(c) Simplification of complex dynamical systems

Owing to incomplete understanding and limited measurements, it is common when modelling real dynamical systems in nature and engineering that the dynamics cannot be accurately explained just by using conservation laws and fundamental process equations. We refer to such systems as complex dynamical systems. The number of candidate models and equations can then be almost as large as the number of modellers. The resulting models also vary greatly in terms of their complexity. More complex models can capture key processes and feedbacks. Complexity is increased by adding more parameters and parametrizations to the existing components (state variables) of the dynamical model, but at some point, it quickly becomes inevitable to add and model new components to capture the underlying real processes accurately, hence further increasing model complexity [29,46,47]. This is common, for example, in marine ecosystem models, where simpler models only resolve the broad biogeochemical classes, while more complex models capture detailed sub-classes [6,7]. Increasing the number of components however can come at great computational cost, can increase the overall uncertainty and can lead to loss of accuracy or stability due to the nonlinearities. Also, the unknown parameters for models with more components are calibrated from available data and the optimization process and parameter estimation quickly become challenging with the increase in complexity, due to the simultaneous explosion in the number of unknown parameters [48]. Thus, instead of adding more unknown parametrizations or increasing the number of components, one might use adaptive models [49,50] and in general the present neural closure models with time delays to incorporate the effects of missing processes in low-complexity models, enabling them to adapt and emulate the response from high-complexity models.

3. Theory and methodology

In this section, we develop the theory and methodology for learning data-assisted closure models for dynamical systems. We first review the MZ formulation [13–15] which derives the exact functional form of the effects of truncated dynamics for common reduced models. Unfortunately,

apart for very simple linear dynamical systems, the use of this formulation is challenging without making unjustified approximations and simplifications. We then discuss the presence of delays in complex dynamical systems, and their impact on modelling [25]. Motivated by the MZ formulation and the presence of delays, we finally derive the new nDDEs and neural closure models, including adjoint equations and network architectures, for both discrete and distributed delays.

(a) MZ formulation and delays in complex dynamical systems

Without loss of generality, the full nonlinear dynamical system model is written as

$$\frac{\mathrm{d}u_k(t)}{\mathrm{d}t} = R_k(u(t), t), \quad \text{with } u_k(0) = u_{0k}, \quad k \in \mathfrak{F}. \tag{3.1}$$

The full-state vector is $u = (\{u_k\})$, $k \in \mathfrak{F} = \mathfrak{R} \cup \mathfrak{U}$, where $\mathfrak{R}$ is the set corresponding to the resolved variables (e.g. coarse field or reduced variables), and $\mathfrak{U}$ the set corresponding to the unresolved variables (e.g. subgrid field or complement variables), which as a union, $\mathfrak{F}$, form the set for full space of variables. We also denote $u = \{\hat{u}, \tilde{u}\}$, where $\hat{u} = (\{u_k\})$, $k \in \mathfrak{R}$ and $\tilde{u} = (\{u_k\})$, $k \in \mathfrak{U}$. Similarly, $u_0 = \{\hat{u}_0, \tilde{u}_0\}$, with $\hat{u}_0 = (\{u_{0k}\})$, $\in \mathfrak{R}$ and $\tilde{u}_0 = (\{u_{0k}\})$, $k \in \mathfrak{U}$.

The MZ formulation allows rewriting the above nonlinear system of ODEs as

$$\frac{\partial}{\partial t}u_k(u_0, t) = \underbrace{R_k(\hat{u}(u_0, t))}_{\text{Markovian}} + \underbrace{F_k(u_0, t)}_{\text{Noise}} + \underbrace{\int_0^t K_k(\hat{u}(u_0, t-s), s)\,\mathrm{d}s}_{\text{Memory}}, \quad k \in \mathfrak{R}, \tag{3.2}$$

where R_k is as in the full model dynamics (equation (3.1)). Importantly, the above equation is an exact representation of equation (3.1) for the resolved components. A derivation is provided in the electronic supplementary material (Section SI-2). Equation (3.2) provides useful guidance for closure modelling. The first term in equation (3.2) is the Markovian term, dependent only on the values of the variables at the present time, while the closure consists of two terms: the noise term and a memory term that is non-Markovian. We can further simplify equation (3.2) by applying the P projection and using the fact that the noise term lives in the null space of P for all times, which could be easily proved. For ROMs with initial conditions devoid of any unresolved dynamics, i.e. $\tilde{u}_0 = 0$ and thus $u_0 = \hat{u}_0$, we then retain the exact dynamics after the projection step, noticing in this case that $Pu_k(u_0, t) = u_k(\hat{u}_0, t), \forall k \in \mathfrak{R}$,

$$\frac{\partial}{\partial t}u_k(\hat{u}_0, t) = PR_k(\hat{u}(\hat{u}_0, t)) + P\int_0^t K_k(\hat{u}(\hat{u}_0, t-s), s)\,\mathrm{d}s, \quad k \in \mathfrak{R}. \tag{3.3}$$

Hence, for such systems, the closure model would only consider the non-Markovian memory term. The above derivation of the MZ formulation has been adapted from [12,14,15].

The MZ formulation clearly shows that a non-Markovian closure term requiring time-lagged state information is theoretically needed to model the unresolved or missing dynamics. This theoretical basis directly applies to the first two classes of low-fidelity dynamical systems (§2), ROMs and coarse resolution models. For the third category, the simplification of complex dynamical systems, we emphasize biological and chemical systems. Many are modelled using ODEs, with one state variable per biological or chemical component. Such ODEs implicitly assume that information between state variables is exchanged instantaneously. In reality, however, there are often time delays for several reasons. First, changes in populations or reactions have non-negligible time scales (e.g. [24,51,52]). Such time scales are introduced in more complex models by modelling intermediate state variables. Hence, the time response of lower-complexity models can be comparable with that of high-complexity models only by explicitly introducing delays [25,26,52,53]. Second, many reactive systems are modelled assuming smooth concentration fields of state variables governed by PDEs with fluid flow advection and/or mixing, leading to advection–diffusion–reaction PDEs [54,55]. In that case, simplified models still require time delays due to the neglected reactive or biogeochemical dynamics but now also due to

truncated modes and/or subgrid-scale processes of numerical models. For all of these reasons, the need for memory-based closure terms is clearly justified to represent complex dynamical systems.

There are some results for data-assisted/data-driven closure modelling based on the MZ formulation. Some schemes create a coupled system of stochastic differential equations using appropriate hidden variables for approximate Markovization of the non-Markovian term [56,57]. Others use a variational approach to derive nonlinear parametrizations approximating the Markovian term [58]. Schemes using machine learning to learn non-Markovian residual of the high- and low-fidelity dynamics limit themselves to specific functional forms for the residual term, a simple Euler time-stepping scheme, and very frequent and uniformly spaced training data [8,11,12]. They also lack the rigorous use of the theory for time-delay systems [59].

(b) Neural delay differential equations

The non-Markovian closure terms with time-lagged state information lead us to DDEs [60]. DDEs have been widely used in many fields such as biology [61,62], pharmacokinetic–pharmacodynamics [63], chemistry [64], economics [65], transportation [66], control theory [67] and climate dynamics [68,69], etc. Next, we summarize the state of the art for learning and solving differential equations using NNs and develop theory and schemes for neural DDEs including adjoint equations for backpropagation.

The interpretation of residual networks as time integration schemes and flow maps for dynamical systems has led to pioneering development of nODEs [20]. A nODE parametrizes an ODE using an NN and solves the initial value problem given by

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = f_{\mathrm{NN}}(u(t), t; \theta), \quad t \in (0, T], \text{ with } u(0) = u_o, \tag{3.4}$$

where f_{NN} is the prescribed NN and θ are the weights. Starting from the initial conditions, the nODE (equation (3.4)) is integrated forward in time using any time-integration scheme, and then gradients are computed based on a loss function using the adjoint sensitivity method. The gradient computation boils down to solving a second ODE backwards in time. Using standard backpropagation for equation (3.4) has however several issues: it would be very memory expensive as one needs to store the state at every time step; its computational cost would increase when using higher-order time-integration or *implicit* schemes; and it might become infeasible if the forward time-integration code does not support automatic differentiation. The adjoint method, however, provides a backpropagation for nODEs [20] that is memory efficient and flexible as it treats the time-integration scheme as a 'black-box'. In our case, we need to incorporate state-delays. Though extending the nODE framework to incorporate DDEs comes under the ambit of universal differential equations (UDEs) [70], deeper investigations are warranted. First, the UDEs are presently implemented using the Julia library DiffEqFlux.jl [71], which can perform automatic adjoint equation solves, but other popular open-source languages such as Python and R, and ML-Frameworks such as TensorFlow [72], PyTorch [73], etc., would require explicit derivation and coding of the corresponding adjoint equations. Second, we need to study two different types of DDEs, the discrete and distributed delays, which it turns out require different architectures. Next, we thus develop the theory and schemes for efficient implementation of nDDEs in any programming language.

(i) Discrete delays

The most popular form of DDEs is

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = f(u(t), u(t-\tau_1), \ldots, u(t-\tau_K), t), \ t \in (0, T], \quad \text{with } u(t) = h(t), \ t \le 0, \tag{3.5}$$

where $\tau_1, \ldots, \tau_K$ are K number of discrete-delays (discrete DDEs). Instead of a single initial value as in the case of ODEs, DDEs require specification of a history function, $h(t)$. Owing to the presence of a given fixed number of delays, we can parametrize the above system by replacing the time-derivative function with potentially any type of NN. For example, to use fully-connected NNs we would concatenate all the delayed states vertically to form the input vector, or concatenate them horizontally to form an input matrix for a convolutional NN. However, recurrent NN (RNN) architectures, such as simple-RNNs, LSTMs, GRUs, etc., are ideal and most efficient for our need due to the time-series nature of the delayed states. We can assume that the discrete delays are evenly spaced (this is not a hard requirement as we can easily extend schemes to irregularly spaced discrete-delays using ODE-RNNs [74], but for brevity we make this assumption) and use an RNN with weights θ. Hence, our new discrete-DDE system can be written as,

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = f_{\mathrm{RNN}}(u(t), u(t-\tau_1), \ldots, u(t-\tau_K), t; \theta), \quad t \in (0, T], \quad \text{with } u(t) = h(t), \ t \le 0, \tag{3.6}$$

where $f_{\mathrm{RNN}}(\bullet; \theta)$ is the recurrent architecture. We refer to this parametrization of discrete DDEs as *discrete-nDDE*. The graphical representation of equation (3.6) in time-discretized form is depicted in figure 1*a*. Let data be available at M times, $T_1 < \cdots < T_M \le T$. We then optimize the total loss function given by, $\mathcal{L} = \int_0^T \sum_{i=1}^M l(u(t))\delta(t - T_i)\,\mathrm{d}t$, where $l(\bullet)$ are scalar loss functions such as mean-squared-error (MSE), and $\delta(t)$ is the Kronecker delta function. To perform this optimization with any gradient descent algorithm, we need the gradient of the loss function w.r.t. the weights of the RNN, θ. Using the adjoint sensitivity method [75] to compute the required gradients, we start by writing the Lagrangian for the above system

$$L = \mathcal{L}(u(t)) + \int_0^T \lambda^T(t)(d_t u(t) - f_{\mathrm{RNN}}(u(t), u(t-\tau_1), \ldots, u(t-\tau_K), t; \theta))\,\mathrm{d}t + \int_{-\tau_K}^0 \mu^T(t)(u(t) - h(t))\,\mathrm{d}t, \tag{3.7}$$

where $\lambda(t)$ and $\mu(t)$ are the Lagrangian variables. In order to find the gradients of L w.r.t. θ, we first solve the following adjoint equation (for brevity we denote, $\partial/\partial(\bullet) \equiv \partial_{(\bullet)}$ and $\mathrm{d}/\mathrm{d}(\bullet) \equiv d_{(\bullet)}$),

$$\left.\begin{aligned} d_t\lambda^T(t) = &\sum_{i=1}^M \partial_{u(t)} l(u(t))\delta(t - T_i) - \lambda^T(t)\partial_{u(t)} f_{\mathrm{RNN}}(u(t), u(t-\tau_1), \ldots, u(t-\tau_K), t; \theta) \\ &- \sum_{i=1}^K \lambda^T(t+\tau_i)\partial_{u(t)} f_{\mathrm{RNN}}(u(t+\tau_i), u(t-\tau_1+\tau_i), \ldots, u(t-\tau_K+\tau_i), t+\tau_i; \theta), \ t \in [0, T) \\ &\lambda(t) = 0, \quad t \ge T. \end{aligned}\right\} \tag{3.8}$$

Details of the derivation of the above adjoint equation (3.8) are in the accompanying electronic supplementary material. Note that equation (3.8) needs to be solved backward in time, and one would require access to $u(t)$, $0 \le t \le T$. In the original nODE work [20], equation (3.6) is solved backward in time and augmented with the adjoint equation (3.8), so as to shrink the memory footprint by avoiding the need to save u at every time step. Solving equation (3.6) backward can however lead to catastrophic numerical instabilities as is well known in data assimilation [76,77]. Improvements have been proposed, such as the ANODE method [78], but they are not applicable in the case of DDEs. In our present implementation, in order to access $u(t)$, $0 \le t \le T$, while solving the adjoint equation, we create and continuously update an interpolation function using the u obtained at every time step as we solve equation (3.6) forward in time. To be more memory efficient, we can, for example, use the method of *checkpointing* [79], or the interpolated reverse dynamic method [80]. After solving for λ, we can compute the required gradients as

$$d_\theta L = -\int_0^T \lambda^T(t)\partial_\theta f_{\mathrm{RNN}}(u(t), u(t-\tau_1), \ldots, u(t-\tau_K), t; \theta)\,\mathrm{d}t. \tag{3.9}$$

Finally, using any gradient descent algorithm, we can find the optimal values of the weights θ.

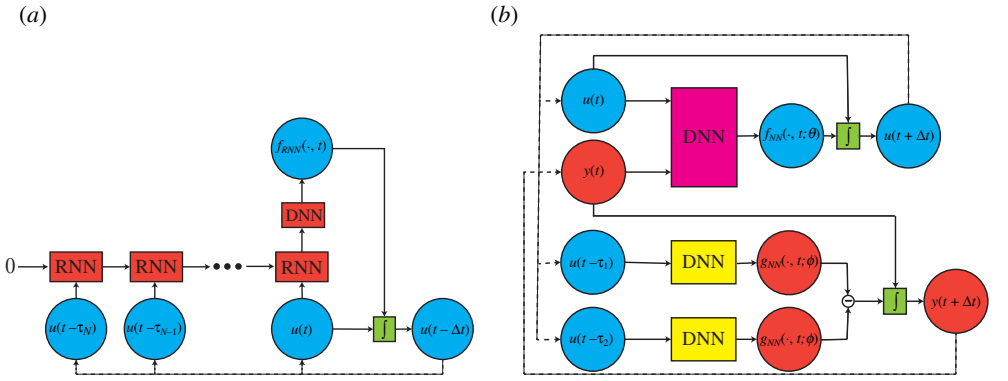

Figure 1. Graphical representation of the time discretized neural delay differential equations (nDDEs). The blocks labelled *RNN* and *DNN* represent any recurrent or deep neural-network architectures, respectively. The block labelled $\int$ symbolizes any time-integration scheme. (*a*) Discrete-nDDE, (*b*) distributed-nDDE. (Online version in colour.)

(ii) Distributed delays

In some applications, the delay is distributed over some past time period [81],

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = f\left(u(t), \int_{t-\tau_2}^{t-\tau_1} g(u(\tau), \tau)\,\mathrm{d}\tau, t\right), \quad t \in (0, T], \text{ with } u(t) = h(t), \quad t \le 0. \tag{3.10}$$

It should be noted that the discrete DDEs can be written as a special case of distributed DDEs using dirac-delta functions. We can approximate the two functions f and g using two different NNs, and rewrite the above equations (3.10) as our new coupled discrete DDEs,

$$\left.\begin{aligned} \frac{\mathrm{d}u(t)}{\mathrm{d}t} &= f_{\mathrm{NN}}(u(t), y(t), t; \theta), \quad t \in (0, T] \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} &= g_{\mathrm{NN}}(u(t-\tau_1), t-\tau_1; \phi) - g_{\mathrm{NN}}(u(t-\tau_2), t-\tau_2; \phi), \ t \in (0, T] \\ \text{with} \qquad u(t) &= h(t), \quad \tau_2 \le t \le 0, \text{ and } y(0) = \int_{-\tau_2}^{-\tau_1} g_{\mathrm{NN}}(h(t), t; \phi)\,\mathrm{d}t, \end{aligned}\right\} \tag{3.11}$$

where $f_{\mathrm{NN}}(\bullet; \theta)$ and $g_{\mathrm{NN}}(\bullet; \phi)$ are the two NNs parametrized by θ and ϕ, respectively. We refer to this parametrization of distributed DDEs as *distributed-nDDE*. The graphical representation of the above system (equations (3.11)) in time-discretized form is depicted in figure 1*b*. Interestingly in the case of distributed-delays, we obtain a novel architecture consisting of two coupled NNs, which enables us to incorporate memory without the use of any recurrent networks such as RNN, LSTMs, GRUs, etc. We can consider f_{NN} as the main network, and g_{NN} as the auxiliary network. Again, we define a scalar loss function given by $\mathcal{L} = \int_0^T \sum_{i=1}^M l(u(t))\delta(t - T_i)\,\mathrm{d}t$ for the available data at M times, $T_1 < \cdots < T_M \le T$. The Lagrangian for the above system is,

$$\begin{aligned} L = \mathcal{L}(u(t)) &+ \int_0^T \lambda^T(t)(d_t u(t) - f_{\mathrm{NN}}(u(t), y(t), t; \theta))\,\mathrm{d}t \\ &+ \int_0^T \mu^T(t)(d_t y(t) - g_{\mathrm{NN}}(u(t-\tau_1), t-\tau_1; \phi) + g_{\mathrm{NN}}(u(t-\tau_2), t-\tau_2; \phi))\,\mathrm{d}t \\ &+ \int_{-\tau_2}^0 \gamma^T(t)(u(t) - h(t))\,\mathrm{d}t + \alpha^T\left(y(0) - \int_{-\tau_2}^{-\tau_1} g_{\mathrm{NN}}(h(t), t; \phi)\,\mathrm{d}t\right), \end{aligned} \tag{3.12}$$

where $\lambda(t), \mu(t), \gamma(t)$ and α are the Lagrangian variables. In order to find the gradients of L w.r.t. the parameters of the two NNs, we first solve the following coupled adjoint equations backward

in time,

$$\left.\begin{aligned} d_t\lambda^T(t) &= \sum_{i=1}^{M} \partial_{u(t)} l(u(t))\delta(t-T_i) - \lambda^T(t)\partial_{u(t)} f_{\mathrm{NN}}(u(t), y(t), t; \theta) \\ &\quad - \mu^T(t+\tau_1)\partial_{u(t)} g_{\mathrm{NN}}(u(t), t; \phi) + \mu^T(t+\tau_2)\partial_{u(t)} g_{\mathrm{NN}}(u(t), t; \phi),\ t \in [0, T) \\ d_t\mu^T(t) &= -\lambda^T(t)\partial_{y(t)} f_{\mathrm{NN}}(u(t), y(t), t; \theta), \quad t \in [0, T) \\ \text{and} \qquad \lambda^T(t) &= 0 \quad \text{and} \quad \mu^T(t) = 0,\ t \geq T. \end{aligned}\right\} \tag{3.13}$$

Details of the derivation of the above adjoint equation (3.13) are in the electronic supplementary material. For accessing u values while solving the adjoint equations, we use the same approach as for our discrete-nDDE (§3(b)((i))). After solving for λ and μ, we can compute the required gradients as

$$\left.\begin{aligned} d_\theta L &= -\int_0^T \lambda^T(t)\partial_\theta f_{\mathrm{NN}}(u(t), y(t), t; \theta)\, \mathrm{d}t \\ \text{and} \qquad d_\phi L &= -\int_0^T \mu^T(t)(\partial_\phi g_{\mathrm{NN}}(u(t-\tau_1), t-\tau_1; \phi) \\ &\quad - \partial_\phi g_{\mathrm{NN}}(u(t-\tau_2), t-\tau_2; \phi))\mathrm{d}t - \mu^T(0)\int_{-\tau_2}^{-\tau_1} \partial_\phi g_{\mathrm{NN}}(h(t), t; \phi)\, \mathrm{d}t. \end{aligned}\right\} \tag{3.14}$$

Finally, using any gradient descent algorithm, we can optimize the NNs f_{NN} and g_{NN}, and find the optimal values of the weights θ and ϕ.

(c) Neural closure models

Now that we have the framework for representing DDEs using NNs, we can replace the non-Markovian memory term in equation (3.3) using nDDEs to obtain a hybrid closure model, which could be trained using data from high-fidelity simulations or real observations. The modified low-fidelity dynamical system with the nDDE closures, which approximates the high-fidelity model, would be given by

$$\frac{\partial \hat{u}(t)}{\partial t} = \underbrace{PR(\hat{u}(t))}_{\text{Low-fidelity}} + \underbrace{f_{\mathrm{RNN}}(\hat{u}(t), \hat{u}(t-\tau_1), \ldots, \hat{u}(t-\tau_K), t; \theta)}_{\text{Neural closure}} \quad \text{with} \quad \hat{u}(0) = \hat{u}_0, \quad t \leq 0 \tag{3.15}$$

using discrete-delays, or by,

$$\frac{\partial \hat{u}(t)}{\partial t} = \underbrace{PR(\hat{u}(t))}_{\text{Low-fidelity}} + \underbrace{f_{\mathrm{NN}}\left(\hat{u}(t), \int_{t-\tau_2}^{t-\tau_1} g_{\mathrm{NN}}(\hat{u}(\tau), \tau; \phi)\, \mathrm{d}\tau, t; \theta\right)}_{\text{Neural closure}} \quad \text{with} \quad \hat{u}(0) = \hat{u}_0, \quad t \leq 0 \tag{3.16}$$

using distributed-delays. The initial conditions at $t = 0$, $\hat{u}_0$, can be used for $t < 0$ as well, as an approximation. Apart from the NN architectures, the amount of delay to be used also becomes a hyperparameter to tune. These novel *neural closure models* provide extreme flexibility in designing the non-Markovian memory term in order to incorporate subject matter expert insights. At the same time, we can also learn the unknown parts of the Markovian low-fidelity model using nODEs if the need arises. Next, we will compare the performance and advantages of using no-delays (nODEs), discrete-delays (discrete-nDDEs) and distributed-delays (distributed-nDDEs) in closure terms for various low-fidelity dynamical systems.

4. Application results and discussion

After presenting the main classes of low-fidelity dynamical models that require closure (§2), we derived a novel, versatile and rigorous methodology for learning and modelling non-Markovian closure terms using nDDEs (§3). The resulting neural closure models have their underpinning in

the MZ formulation and the presence of inherent delays in models of complex dynamical systems such as biogeochemical systems. Now, we evaluate the performance and advantages of these new neural closure models over those of neural ODEs (Markovian).

We run experiments encompassing each of the classes of low-fidelity models (§2). For each experiment, we follow the same training protocol for nODEs (no-delays) and the two nDDEs, discrete-nDDE (discrete-delays) and distributed-nDDE (distributed-delays), closure models. The training data are regularly sampled from high-fidelity simulations in all experiments, but this is not a requirement. We use performance over the validation period (past the period for which high-fidelity data snapshots are used for training) to fine tune various training-related hyperparameters. The final evaluation is based on much longer-term future prediction performance, well past these periods. As the field of scientific machine learning (SciML; [82]) is relatively new, the metrics for performance evaluation vary greatly. On the one hand, many learning studies randomly sample small time sequences from a given period for which high-fidelity data are available, and then split them into training, validation and test sets [21]. As the training, validation and test sets belong to the same time domain, hence, the learned networks are only evaluated for their interpolation performance and predicting the unseen data becomes easy for them. On the other hand, for the few studies where the training and test (prediction) periods do not overlap, the prediction period is often much shorter than the training period [83]. In the present work, we consider a more stringent evaluation. First, our validation period does not overlap the training period. Second, our future prediction period is equal to or much longer than the training and validation periods, and has no overlap with either. Hence, we strictly measure the out-of-sample/generalization performance of the learned network for its extrapolation capabilities into the future. Of course, other evaluation metrics are possible and there is indeed a need for standardization of evaluation procedures in the SciML community. In the rest of the paper, for all the figure, table and section references prefixed with 'SI-', we direct the reader to the electronic supplementary material.

(a) Experiments 1: advecting shock—reduced order model

For the first experiments, neural closure models learn the closure of POD-GP-based reduced order model of the advecting shock problem. The FOM for this problem is given by Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial^2 x}, \tag{4.1}$$

where ν is the non-dimensional diffusion coefficient. The initial and boundary conditions are

$$u(x,0) = \frac{x}{1+\sqrt{1/t_0}\exp(Re(x^2/4))}, \quad u(0,t) = 0 \quad \text{and } u(L,t) = 0, \tag{4.2}$$

where $Re = 1/\nu$ and $t_0 = \exp(Re/8)$. Let the POD of the state variable $u(x,t)$ be given by, $u(x,t) = \bar{u}(x) + \sum_{i=1}^{m} u_i(x)a_i(t)$, then we obtain the reduced-order equations as outlined in §2a,

$$\begin{aligned}\frac{\mathrm{d}a_k}{\mathrm{d}t} = &-\left\langle \bar{u}\frac{\partial \bar{u}}{\partial x}, u_k\right\rangle - a_i\left\langle u_i\frac{\partial \bar{u}}{\partial x}, u_k\right\rangle - a_j\left\langle \bar{u}\frac{\partial u_j}{\partial x}, u_k\right\rangle - a_i a_j\left\langle u_i\frac{\partial u_j}{\partial x}, u_k\right\rangle \\ &+\left\langle \nu\frac{\partial^2 \bar{u}}{\partial x^2}, u_k\right\rangle + a_i\left\langle \nu\frac{\partial^2 u_i}{\partial x^2}, u_k\right\rangle, \quad \text{with } a_k(0) = \langle (u(x,0) - \bar{u}(x)), u_k(x)\rangle. \end{aligned} \tag{4.3}$$

We solve the FOM (equations (4.1) and (4.2)) for $Re = 1000$, $L = 1$ and maximum time $T = 4.0$. The singular value decomposition of this solution form the POD modes for the ROM. We only keep the first three modes, which capture 60.8% of energy, and evolve the corresponding coefficients using equation (4.3), thus requiring a closure. The high-fidelity or true coefficients are obtained solving the FOM (equation (4.1)) with initial conditions without the contribution from the unresolved modes, i.e. $u(x,0) = \bar{u}(x) + \sum_{i=1}^{3} u_i(x)a_i(0)$, and projecting the obtained solution onto the first three modes. For comparison, we also present the true coefficients in figure 2, which is what the ROMs with neural closure are trying to match. For this true data generation, we solve the FOM using an

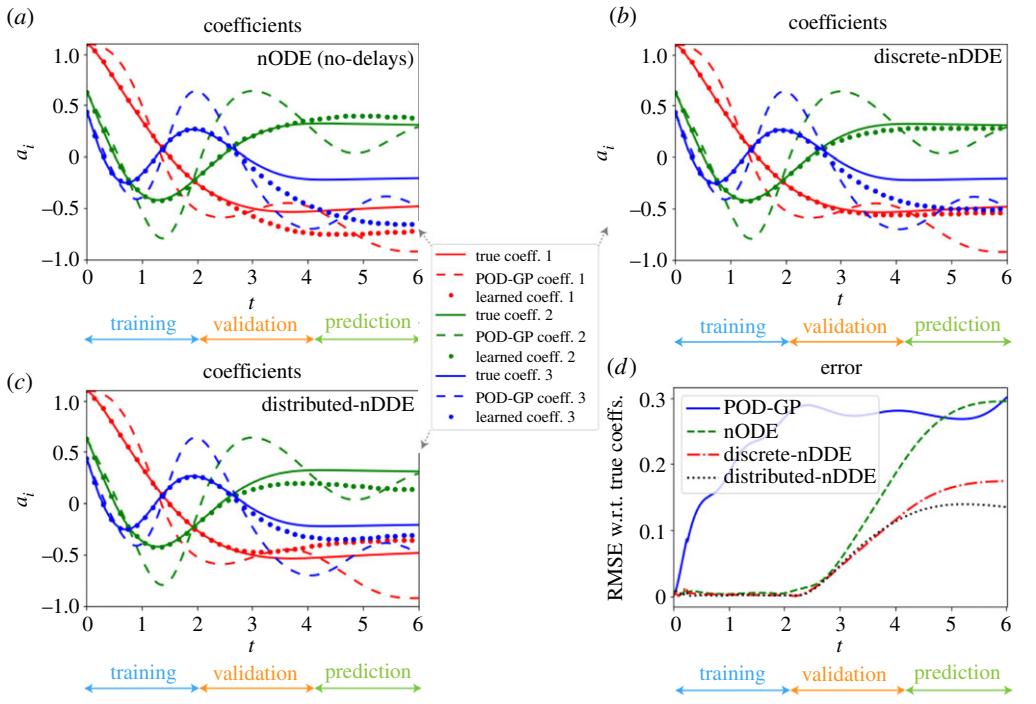

Figure 2. Comparison of the true coefficients (solid) with the coefficients from the POD-GP ROM (dashed-dot) and from the POD-GP ROMs augmented with the three different learned neural closure models at the end of training (dashed). For each neural closure, the training period is from $t = 0$ to 2.0, the validation period from $t = 2.0$ to 4.0 and the future prediction period from $t = 4.0$ to 6.0. (*a*) Neural ODEs with no-delays (nODE); (*b*) neural DDEs with discrete-delays (discrete-nDDE); and (*c*) neural DDEs with distributed-delays (distributed-nDDE). (*d*) Evolution of root-mean-squared-error $\left(\text{RMSE}(t) = \sqrt{(1/3)\sum_{k=1}^{3} |a_k^{\text{pred}}(t) - a_k^{\text{true}}(t)|^2}\right)$ of coefficients from the four different ROMs. These results correspond to the architectures detailed in the electronic supplementary material, table SI-1. (Online version in colour.)

explicit Runge–Kutta (RK) time-integration of order (4)5 (*dopri5*; [84]) with adaptive time-stepping (storing data at time steps of $\Delta t = 0.01$) and grid spacing of $\Delta x = 0.01$, using finite-difference schemes (upwind for advection and central difference for diffusion).

Our three test periods for the advecting shock ROM (equation (4.3)) with three modes are as follows. For training our neural closure models, we only use the true coefficient values up to time $t = 2.0$. For validation (used only to tune hyperparameters), we use true coefficient values from $t = 2.0$ to $t = 4.0$. Finally for testing, we make a future prediction from $t = 4.0$ to final time $T = 6.0$. We compare the three different closures: nODE (no-delays), discrete-nDDE and distributed-nDDE with architecture details presented in the electronic supplementary material, table SI-1. The architectures are not exactly the same for the three cases, but they are set up to be of comparable expressive power. Mostly, we employ a bigger architecture for the no-delays case in order to help it compensate the lack of past information. We also ensure that the networks are neither under-parametrized nor over-parametrized. Along with the classical hyperparameters such as batch size, number of iterations per epoch, number of epochs, learning rate schedule, etc., we also have the delay values ($\tau_1, \ldots, \tau_K$ for discrete-nDDE; and τ_1, τ_2 for distributed-nDDE) as additional hyperparameters to tune. We chose to use six discrete delays for the discrete-nDDE in the present experiments. The values of other hyperparameters are given in the electronic supplementary material, Section SI-4.2. For evaluation, at each epoch, we evolve the coefficient of the learned system ($\{a^{\text{pred}}(T_i) = \{a_k^{\text{pred}}(T_i)\}_{k=1}^{3}\}_{i=1}^{M}$) using the RK time-integration scheme mentioned earlier, and compare them with the true coefficients ($\{a^{\text{true}}(T_i)\}_{i=1}^{M}$) using the time-averaged L_2 error,

$\mathcal{L} = (1/M)\sum_{i=1}^{M}\left(\sqrt{\sum_{k=1}^{3}|a_k^{\text{pred}}(T_i) - a_k^{\text{true}}(T_i)|^2}\right)$, which is also our loss function for training. The error for the time period $t = 0$–2.0 forms the training loss, the error for $t = 2.0$–4.0 the validation loss and the error for $t = 4.0$–6.0 the prediction loss.

The performance of the three neural closure models after 200 epochs (the stochastic gradient descent nearly converges, see the electronic supplementary material, figure SI-2a) is evaluated by comparison with the true coefficients and with the POD-GP coefficients spanning training, validation and future prediction periods. Results are shown in figure 2. The details of the architectures employed are in the electronic supplementary material, table SI-1. We find that using no-delays (nODE), discrete-delays (discrete-nDDE) and distributed-delays (distributed-nDDE) perform equally well for the training period, exactly matching the true coefficients. As soon as one enters the validation period, all the neural closure models start to slightly diverge, with the nODE diverging the most by the end of the prediction period. Importantly, both nDDE closures maintain a great improvement over just using the POD-GP model, and showcase a better performance than the nODE closure, even though the latter had a deeper architecture with significantly more trainable parameters. We also find that the performance of the distributed-nDDE closure is a little better than that of the discrete-nDDE closure for the prediction period. In a similar set of experiments, Maulik *et al.* [21] (section 3.1, 'Advecting Shock') used nODE and LSTM to learn the time evolution of the first three high-fidelity (true) coefficients without using the known physics/low-fidelity model. As a result, they required bigger architectures and more time samples for training data than we do. This confirms our benefits of learning only the unknown closure model. Due to the highly nonlinear nature of NNs, analytical stability analyses are not direct. Nonetheless, we provide empirical stability results by reporting the evolution of the root-mean-square-error (RMSE) (figure 2). We find that both discrete-nDDE and distributed-nDDE closures, due to the existence of delays, may have a stronger dissipative character and thus show better stability at later times than the POD-GP and the nODE closure.

One might expect the distributed-delay (distributed-nDDE) to always perform better than the discrete-nDDE closure because of the presence of the integral of the state variable over a delay period instead of the state variable at specific points in the past. The former thus seemingly contains more information, but there is in fact no guarantee for this being true in all cases. We can derive an intuition for this from information theory. According to the data processing inequality [85], let X and Y be two random variables, then,

$$I(g(X); Y) \leq I(X; Y), \tag{4.4}$$

where I is the mutual information and g is any function which post-processes X. Now, if X is composed of K random variables, $X = \{X_1, \ldots, X_K\}$, and $g(X) = X_1 + \cdots + X_K$, then,

$$I(X_1 + \cdots + X_K; Y) \leq I(\{X_1, \ldots, X_K\}; Y). \tag{4.5}$$

If we consider the effect of the integral of the state variable over the delay period in the case of distributed-nDDE as a data processing step, this might actually be decreasing the information content compared with the discrete-nDDE closure. We use 'might', even though equation (4.5) is a strong bound, because in the present experiments we only use six delay values for the discrete-nDDE, while the integral in the distributed-nDDE is computed using many past state values, and also the architectures are different. Hence, a direct comparison using the data processing inequality (equation (4.5)) is not possible, but it provides us with a plausible explanation.

In addition to the results just illustrated, we completed many other Experiments-1 to assess the sensitivity of our framework to various hyperparameters. In all cases, the time period corresponding to the training data should be at least equal to one characteristic time scale of the dynamics, otherwise the prediction performance deteriorated, as shown in the electronic supplementary material, figure SI-3a and discussed in the electronic supplementary material, Section SI-4.3. Adding the neural closure to the low-fidelity model improved its matching with the high-fidelity data in nearly all cases. Its performance deteriorated with increasing the length of the time sequences used to form the batches, and also with increasing the batch size (the

number of iterations per epoch is a dependent hyperparameter as mentioned earlier). This also led to an increase in training time. Depth of the networks affected the performance significantly, with shallower networks performing poorer than deeper networks as expected, however, the incremental gain in performance starts to taper off after certain depths (see the electronic supplementary material, figure SI-3b and Section SI-4.3). Using an exponentially decaying learning schedule over a constant learning rate tremendously improved learning performance and reduced the number of epochs needed. Further, training times slightly increased when using more delay times in the case of discrete-nDDE. In general, we found that the training time for discrete-nDDEs was similar to that for distributed-nDDEs. Such behaviours by machine learning methods are difficult to anticipate in advance but they should be mentioned.

Overall, in the Experiments-1, we find that using memory-based neural closure models as we derived from the MZ formulation is advantageous over just a Markovian closure. Using the new nDDEs as closure models helps maintain generalizability of the learned models for longer time periods, and significantly reduces the longer-term prediction error of the ROM.

(b) Experiments 2: Advecting shock—subgrid-scale processes

In the second experiments, we again use the advecting shock problem governed by Burger's equation (equation (4.1)), but we now reduce the computational cost of the FOM by coarsening the spatial resolution, again leading to the need of a closure model (§2b). For the high-fidelity/high-resolution solution, we employ a fine grid with the number of grid points in the x-direction $N_x = 100$, while for the low-fidelity/low-resolution solution, we employ a four times coarser grid with $N_x = 25$. A comparison of high- and low-resolution solutions solved using exactly the same numerical schemes and data stored at every time step of $\Delta t = 0.01$ is provided in figure 3. We observe that by decreasing the resolution, we introduce numerical diffusion and error in the location of the shock peak at later times. The goal of the neural closure models in these experiments is thus to augment the low-resolution model such that it matches the sub-sampled/interpolated high-resolution solution at the coarse (low-resolution) grid points.

For training our neural closure models for the low-resolution discretization with $N_x = 25$, we use the same training regiment as in Experiments-1 (§4a), with architecture details presented in the electronic supplementary material, table SI-1. In order to exploit the fact that each grid point only affects its immediate neighbours over a single time step, we use one-dimensional convolutional layers for these experiments. For the nODE, we again employ a deeper architecture with more trainable parameters, and for the discrete-nDDE, six discrete delay values are again used. The values of the other hyperparameters are in the electronic supplementary material, Section SI-4.2. The validation period is from $t = 1.25$ to 2.5, and the future prediction period from $t = 2.5$ to 5.0. We have chosen a prediction period of the combined length of training and validation periods. For time-integration, we use the *Vode* scheme [86] with adaptive time-stepping. The true data are generated by interpolating the high-resolution solution onto the low-resolution grid ($\{\{u^{\text{true}}(x_k, T_i)\}_{k=1}^{N_x=25}\}_{i=1}^{M}$), as shown in figure 3*c*, and we use the time-averaged L_2 error, $\mathcal{L} = (1/M)\sum_{i=1}^{M}\left(\sqrt{\sum_{k=1}^{N_x=25}|u^{\text{pred}}(x_k, T_i) - u^{\text{true}}(x_k, T_i)|^2}\right)$, as the loss function.

The performance of the three neural closure models after 250 epochs (the stochastic gradient descent nearly converges, see the electronic supplementary material, figure SI-2b) is evaluated by taking the absolute difference with the high-resolution solution interpolated onto the low-resolution grid (figure 3*c*) spanning training, validation and prediction periods. We further benchmark our performance against the popular Smagorinsky model [87] used for subgrid-scale turbulence closure in large eddy simulation (LES). For Burger's equation (4.1), it introduces a dynamic turbulent eddy viscosity (ν_e) leading to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} + \frac{\partial}{\partial x}\left(\nu_e\frac{\partial u}{\partial x}\right), \tag{4.6}$$

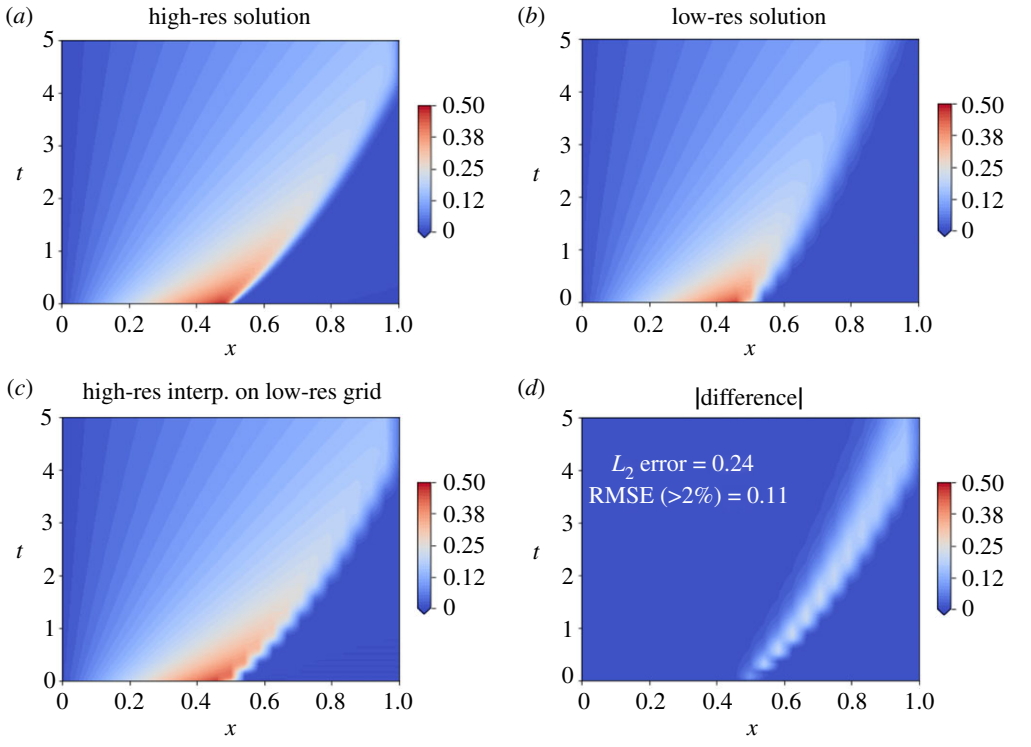

Figure 3. Comparison of solutions of Burger's equation (equation (4.1)) for different grid resolutions. (*a*) solution for a high-resolution grid with number of grid points, $N_x = 100$; (*b*) solution for a low-resolution grid with $N_x = 25$; (*c*) high-resolution solution interpolated onto the low-resolution grid. (*d*) Absolute difference between fields in panels (*b,c*). We also provide a pair of time-averaged errors, specifically: L_2 error; and RMSE considering only the grid points where the error is at least 2% of the maximum velocity value, denoted by RMSE(>2%). (Online version in colour.)

where $\nu_e = (C_s\Delta x)^2|\partial u/\partial x|$ and C_s is the Smagorinsky constant. Results are shown in figure 4. The details of the architectures employed are in the electronic supplementary material, table SI-1. As shown by the error fields of the baseline (figure 3*d*) and closure models (figure 4), and by the corresponding pairs of averaged error numbers (see figures), all closures improve the baseline. However, the nODE and Smagorinsky closures only lead to a 55–60% decrease in error, while the nDDE closures achieve a 80–90% decrease. Despite the deeper architecture for the nODE, both the discrete-nDDE and distributed-nDDE (with smaller architectures) again achieve smaller errors, for the whole period of $t = 0$–5.0. This means that they have lower numerical diffusion, thus capturing the targeted subgrid-scale process. As opposed to the findings of Experiments-1 (figure 2), in the present Experiments-2, the discrete-nDDE performs slightly better than the distributed-nDDE in the prediction period.

We now study the effect of changing the amount of past information incorporated in the closure model on the time-averaged L_2 error. For this, we fix $\tau_1 = 0$ for the distributed-nDDE closure, and vary the values of only τ_2, keeping the architecture the same (electronic supplementary material, table SI-1). First, for the time-averaged training loss (not shown), we found no discernible trend and differences were mostly due the stochastic gradient descent. Second, for the validation period, figure 5*a* shows the statistical summary of the validation losses (time-averaged L_2 error) between the last epochs 200–250, for different delay-period lengths. In order to ensure statistical soundness of the results, 10–15 repeats of the training were done, and the results aggregated for each delay-period length. Results indicate that the validation loss first decreases and then increases as we incorporate more-and-more past information, starting from a very small delay period. For a specified architecture, neither too little nor too much past

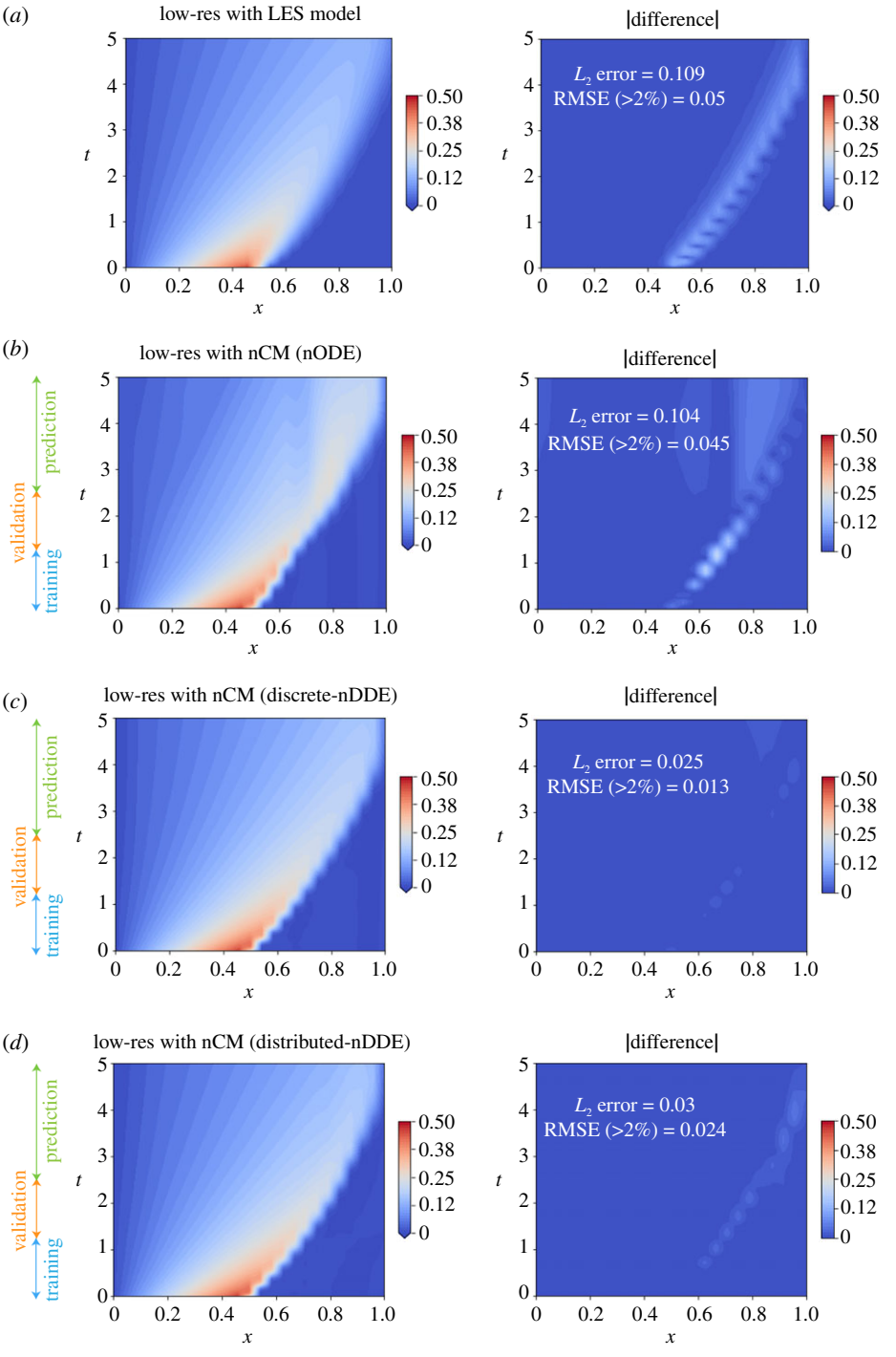

Figure 4. Solutions of Burger's PDE on the low-resolution grid with different closure models (*left column*), and their absolute differences (*right column*) with the high-resolution solution interpolated onto the low-resolution grid (figure 3*c*). For the trained neural closure models, the training period is from $t = 0$ to 1.25, the validation period from $t = 1.25$ to 2.5, and the prediction period from $t = 2.5$ to 5.0. For each closure, we also provide the pair of time-averaged errors (see figure 3 for description). (*a*) Smagorinsky LES model with $C_s = 1.0$; (*b*) neural closure model with no-delays (nODE); (*c*) neural closure model with discrete-delays (discrete-nDDE); (*d*) neural closure model with distributed-delays (distributed-nDDE). These results correspond to the architectures detailed in electronic supplementary material, table SI-1. (Online version in colour.)

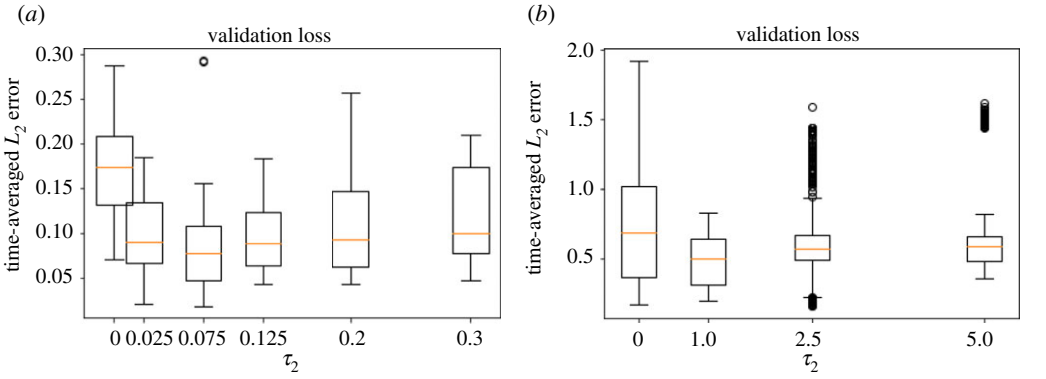

Figure 5. Variation of distributed-nDDE closure validation loss (time-averaged L_2 error) averaged over the last 50 training epoch for Experiments-2 and -3a. All the experiments have $\tau_1 = 0$ and different τ_2 (horizontal axis). Note that $\tau_2 = 0$ corresponds to the nODE closure. We use boxplots to provide statistical summaries for multiple training repeats done for each experiment. The box and its whiskers provide a five-number summary: minimum, first quartile (Q1), median (orange solid line), third quartile (Q3) and maximum, along with outliers (black circles) if any. (*a*) Experiments-2 and (*b*) Experiments-3a. (Online version in colour.)

information is helpful: there is likely an optimal amount of information to incorporate. The initial improvement in the performance of the closure models as the delay period is increased is due to the increase in information content about the recent past. However, a particular network architecture of finite size will have a limit on capturing the increasing information content effectively due to its limited expressive power, thus leading to a decrease in performance when too much information is provided. An estimate of the range of delay period lengths to consider can be obtained from properties of the given dynamical system such as the main time scales, e.g. advection and diffusion times scales in the present system, and main decorrelation times of state variables. Overall, from figure 5*a*, we can notice the optimal delay period length to be around 0.075. Similar trends between performance of neural closure models and delay period lengths were also found in Experiments-1 (not shown). Some published studies attempt to derive analytical expressions for the optimal memory length, making many approximations in the process [8,88]. A final option is to learn the delay amount as a part of the training process itself, however, this requires modified adjoint equations.

We conducted again a series of Experiments-2 to assess the sensitivity to the various other hyperparameters, and found similar trends (not shown here) as in the Experiments-1. Finally, we noticed that using the *dopri5* [84] time-integration scheme severely impaired the learning ability in the Experiments-2.

Overall, these results demonstrate the superiority of using our new memory-based closure models in capturing subgrid-scale processes.

(c) Experiments 3a: 0-D marine biological models

For our third experiments, we use neural closure models to incorporate the effects of missing processes and state variables in lower-complexity biological models, thus targeting the third class of closure modelling (§2c). Marine biological models are based on ODEs that describe the food-web interactions in the ecosystem. They can vary greatly in terms of complexity [29]. The marine biological models used in our experiments are adapted from Newberger *et al.* [30]. They were used to simulate the ecosystem in the Oregon coastal upwelling zone. They provide hierarchical embedded models compatible with each other. We employ the three-component NPZ model (nutrients (N), phytoplankton (P) and zooplankton (Z)) and the five-component NNPZD model (ammonia (NH_4), nitrate (NO_3), P, Z and detritus (D)) in a zero-dimensional setting (0-D; only

temporal variation). The low complexity NPZ model is given by

$$\left.\begin{aligned}
\frac{\mathrm{d}N}{\mathrm{d}t} &= -G\frac{PN}{N+K_u} + \Xi P + \Gamma Z + R_m\gamma Z(1-\exp^{-\Lambda P}),\\
\frac{\mathrm{d}P}{\mathrm{d}t} &= G\frac{PN}{N+K_u} - \Xi P - R_m Z(1-\exp^{-\Lambda P}),\\
\frac{\mathrm{d}Z}{\mathrm{d}t} &= R_m(1-\gamma)Z(1-\exp^{-\Lambda P}) - \Gamma Z \quad \text{with} \quad N(0)=T_{\mathrm{bio}}, \quad P(0)=0 \quad \text{and} \quad Z(0)=0,
\end{aligned}\right\} \tag{4.7}$$

with G representing the optical model,

$$G = V_m\frac{\alpha I}{(V_m^2+\alpha^2 I^2)^{1/2}} \quad \text{and} \quad I(z) = I_0\exp(k_w z), \tag{4.8}$$

where z is depth and $I(z)$ models the availability of sunlight for photo-chemical reactions. The parameters in equations (4.7) and (4.8) are: k_w, light attenuation by sea water; α, initial slope of the $P-I$ curve; I_0, surface photosynthetically available radiation; V_m, phytoplankton maximum uptake rate; K_u, half-saturation for phytoplankton uptake of nutrients; Ξ, phytoplankton-specific mortality rate; R_m, zooplankton maximum grazing rate; Λ, Ivlev constant; γ, fraction of zooplankton grazing egested; Γ, zooplankton-specific excretion/mortality rate; and T_{bio}, total biomass concentration. In the NPZ model (equation (4.7)), the nutrient uptake by phytoplankton is governed by a Michaelis–Menten formulation, which amounts to a linear uptake relationship at low nutrient concentrations that saturates to a constant at high concentrations. The grazing of phytoplankton by zooplankton follows a similar behaviour: their growth rate becomes independent of P in the case of abundance, but proportional to available P when resources are scarce, hence zooplankton grazing is modelled by an Ivlev function. The death rates of both P and Z are linear, and a portion of zooplankton grazing in the form of excretion goes directly to nutrients.

In the higher complexity NNPZD model, the nutrients are divided into ammonia and nitrates, which are the two most important forms of nitrogen in the ocean. With the intermediate of decomposed organic matter, detritus, the NNPZD model captures new processes such as: phytoplankton cells preferentially taking up ammonia over nitrate because the presence of ammonia inhibits the activity of the enzyme nitrate reductase essential for the uptake kinetics; the pool of ammonium coming from remineralization of detritus; and part of this ammonium pool getting oxidized to become a source of nitrate called the process of nitrification, etc. Overall, the NNPZD model is given by,

$$\left.\begin{aligned}
\frac{\mathrm{dNO_3}}{\mathrm{d}t} &= \Omega\mathrm{NH_4} - G\left[\frac{\mathrm{NO_3}}{\mathrm{NO_3}+K_u}\exp^{-\Psi\mathrm{NH_4}}\right]P\\
\frac{\mathrm{dNH_4}}{\mathrm{d}t} &= -\Omega\mathrm{NH_4} + \Phi D + \Gamma Z - G\left[\frac{\mathrm{NH_4}}{\mathrm{NH_4}+K_u}\right]P\\
\frac{\mathrm{d}P}{\mathrm{d}t} &= G\left[\frac{\mathrm{NO_3}}{\mathrm{NO_3}+K_u}\exp^{-\Psi\mathrm{NH_4}} + \frac{\mathrm{NH_4}}{\mathrm{NH_4}+K_u}\right]P - \Xi P - R_m Z(1-\exp^{-\Lambda P})\\
\frac{\mathrm{d}Z}{\mathrm{d}t} &= R_m(1-\gamma)Z(1-\exp^{-\Lambda P}) - \Gamma Z\\
\frac{\mathrm{d}D}{\mathrm{d}t} &= R_m\gamma Z(1-\exp^{-\Lambda P}) + \Xi P - \Phi D\\
\text{with} \quad & \mathrm{NO_3}(0)=T_{\mathrm{bio}}/2, \quad \mathrm{NH_4}(0)=T_{\mathrm{bio}}/2, \quad P(0)=0,\ Z(0)=0, \text{ and } D(0)=0,
\end{aligned}\right\} \tag{4.9}$$

where the new parameters are: Ψ, $\mathrm{NH_4}$ inhibition parameter; Φ, detritus decomposition rate and Ω, $\mathrm{NH_4}$ oxidation rate.

Solutions of the above two models are presented in figure 6. Different values of the parameters and initial conditions set these models in different dynamical regimes. From the responses in time, the present solutions in Experiments-3a are in a stable nonlinear limit-cycle regime. The N class in the NPZ model is a broader class encompassing $\mathrm{NO_3}$, $\mathrm{NH_4}$ and D from the NNPZD

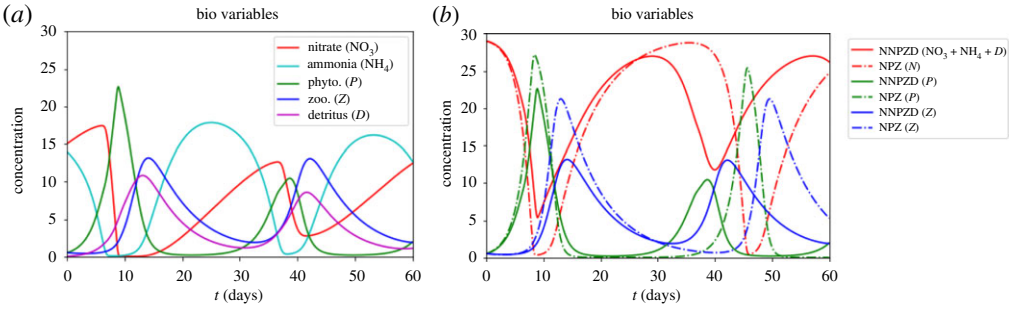

Figure 6. Solutions of the marine biological models used in Experiments-3a (concentrations versus time in *days*). Parameter values used are (adapted from [30]): $k_w = 0.067\ \mathrm{m}^{-1}$, $\alpha = 0.025\ (\mathrm{W\,m^{-2}\,d})^{-1}$, $V_m = 1.5\ \mathrm{d}^{-1}$, $I_0 = 158.075\ \mathrm{W\,m^{-2}}$, $K_u = 1\ \mathrm{mmol\,N\,m^{-3}}$, $\Psi = 1.46\ (\mathrm{mmol\,N\,m^{-3}})^{-1}$, $\Xi = 0.1\ \mathrm{d}^{-1}$, $R_m = 1.52\ \mathrm{d}^{-1}$, $\Lambda = 0.06\ (\mathrm{mmol\,N\,m^{-3}})^{-1}$, $\gamma = 0.3$, $\Gamma = 0.145\ \mathrm{d}^{-1}$, $\Phi = 0.175\ \mathrm{d}^{-1}$, $\Omega = 0.041\ \mathrm{d}^{-1}$, $z = -25$ m, and $T_{\mathrm{bio}} = 30\ \mathrm{mmol\,N\,m^{-3}}$. (*a*): Nitrate-ammonia-phytoplankton-zooplankton-detritus (NNPZD) model (equation (4.9)); (*b*): comparison between $NO_3 + NH_4 + D$, P and Z from the NNPZD model (*solid*) with N, P and Z from the nutrient-phytoplankton-zooplankton (NPZ) model (*dashed-dot*; Eq. (4.9)). (Online version in colour.)

model. Since the NNPZD model resolves many more processes, the concentrations of $NO_3 + NH_4 + D$, P and Z differ significantly from the N, P and Z of the NPZ model. The goal of the neural closure models in these experiments is thus to augment the low-complexity NPZ model such that it matches the aggregated states of the high-complexity NNPZD model.

For training our neural closure models for the NPZ model, we use the same training regiment as in Experiments-1 and -2 (§4a,b), with architecture details presented in the electronic supplementary material, table SI-2. For the nODE, we again employ a bigger architecture, and for the discrete-nDDE, six discrete delay values are again used. The values of other hyperparameters are given in the electronic supplementary material, Section SI-4.2. The training period ranges from $t = 0$ to 30 days, validation period from $t = 30$ to 60 days and the prediction period from $t = 60$ to 330 days. We have chosen a prediction period nine times longer than the training period. For biological ODE models, there exists invariant knowledge about the system, such as biological state variables cannot be negative, and the sum of all the states remains constant with time (this can be verified by summing the ODEs of NPZ or NNPZD models). We enforce the constraints as follows. The positivity is enforced as a penalization term in the loss function. The constant total biomass constraint is embedded in the architectures of neural closures by introducing a new custom layer named *BioConstrainLayer*. This layer is applied at the end, and expects an input of size 1. The output of this layer is formed by splitting the input into three with the proportions, β, -1 and $1 - \beta$, where β is a trainable parameter. This ensures that summing the right-hand side of the augmented NPZ model does not leave any new residual due to the neural closure terms. The stiffness of such biological ODE models also poses a challenge in maintaining these desired properties [89]. The flexibility of our framework however allows the use of appropriate time-stepping schemes, such as A-stable implicit schemes, etc., to overcome stiffness. The true data are generated by aggregating the variables of the NNPZD model ($N \equiv NO_3 + NH_4 + D$, P and Z; $\{\{B^{\mathrm{true}}(T_i)\}_{B\in\{N,P,Z\}}\}_{i=1}^{M}$). Finally, we use the *dopri5* [84] scheme with adaptive time-stepping and simulation data were stored at every $\Delta t = 0.05$ days for all our time-integration requirements, along with a L_2 error loss function, $\mathcal{L} = (1/M)\sum_{i=1}^{M}\left(\sqrt{\sum_{B\in\{N,P,Z\}}|B^{\mathrm{pred}}(T_i) - B^{\mathrm{true}}(T_i)|^2}\right)$.

The performance of the three neural closure models augmenting the NPZ is evaluated after 350 epochs of training (the stochastic gradient descent nearly converges, as evident from the electronic supplementary material, figure SI-2c) by comparison with the aggregated biology variables from the high-complexity NNPZD model (equation (4.9)) spanning training, validation and prediction periods. Results are presented in figure 7. The details of the architectures employed

are in the electronic supplementary material, table SI-2. When compared with the aggregated NNPZD variables (true variables), we find again that despite the bigger architecture of the nODE, it starts to develop significant errors around $t = 180$ days and quickly gets out-of-phase thereafter (figure 7*a*). The discrete-nDDE and distributed-nDDE, both with smaller architectures, are however able to match the true variables for nearly the whole period of $t = 0$–330 days (figure 7*b*), with only distributed-nDDE starting to get out-of-phase after $t = 270$ days (figure 7*c*) at the end of the long prediction period. These results are corroborated by the time evolution of the RMSE and average cross-correlation for the three variables over the prediction period (figure 7*d*). From the progression of the time-averaged L_2 loss (here, the error between the variables from the closure-model-augmented NPZ system, and the true variables), the nODE performs either equally well or better than both discrete-nDDE and distributed-nDDE during training and validation periods (electronic supplementary material, figure SI-2c), however, it is not able to maintain long-term accuracy. We also notice very large spikes in the first half of the training regime, which are due to weights of the NNs taking values that lead to negative biology variables. As training progresses, we however do not observe this behaviour anymore because the trainable weight start to converge towards biologically feasible regimes. In conclusion, using a memory-based closure for a low-complexity model can efficiently help emulate the high-complexity model.

As for Experiments-2, we conducted a series of Experiments-3a to study the effect of changing the amount of past information incorporated in the neural closure models. In figure 5*b*, we show the variation of the average validation loss (time-averaged L_2 error) between the last epochs 300 to 350, for different delay-period lengths ($\tau_1 = 0$, and τ_2 varying in case of distributed-nDDE). In order to ensure statistical soundness of the results, 10–12 repeats of the training were done, and the results were aggregated for each delay-period length (excluding the runs which diverged). We again find an optimal memory length for a specified architecture, however, with more and more runs failing to converge for longer delay period lengths. For the present system, estimates of delay period lengths to consider can be obtained from the time scales of biological behaviours and adjustments, and from the decorrelation times of the biological state variables. Taking into account the limited effectiveness of a network architecture of finite size for capturing increasing information content, from figure 5*b*, we find an optimal delay period length to be around 1 *day*. We also conducted a series of Experiments-3a to study the sensitivity to the various hyperparameters, and found similar trends (not shown here) as in Experiments-1 and -2. For good performance, we further found that using a small enough time step was critical as well as limiting the number of internal steps in the *dopri5* [84] time-integration scheme, while penalizing negative values in the loss function did not make much of a difference. Whenever multiple terms are present in the loss function enforcing different inherent properties of the system, they should be normalized (e.g. using non-dimensional variables) and given appropriate relative weights.

In general, the ecosystem ODEs are coupled with regional or global ocean modelling systems, leading to advection–diffusion–reaction PDEs [90]. If highly complex ecosystem models are employed, a very large number of PDE state variables need to be solved for, rendering the computations very expensive. A large number of unknown parameter values as well as uncertain initial conditions then also need to be estimated, requiring specific methods (e.g. [91]). The available biogeochemical observations are not always sufficient for calibrating these many unknown parameters and for estimating the initial conditions of high-complexity models. If the corresponding errors are large, this can lead to integrating models in the wrong dynamical regimes (e.g. [92]). Finally, in some applications, one is only interested in the aggregated state variables, but cannot use low-complexity models because their dynamics are too inaccurate for the goals of the applications. Using neural closure models as shown here, one can increase the accuracy of the low-complexity models to match the response of high-complexity models (possibly up to models such as ERSEM [93]) without adding the computational burden of modelling all the intermediate biological states and processes, while reducing the effects of other uncertainties listed above. Results of our neural closures in one-dimensional PDEs are showcased next.

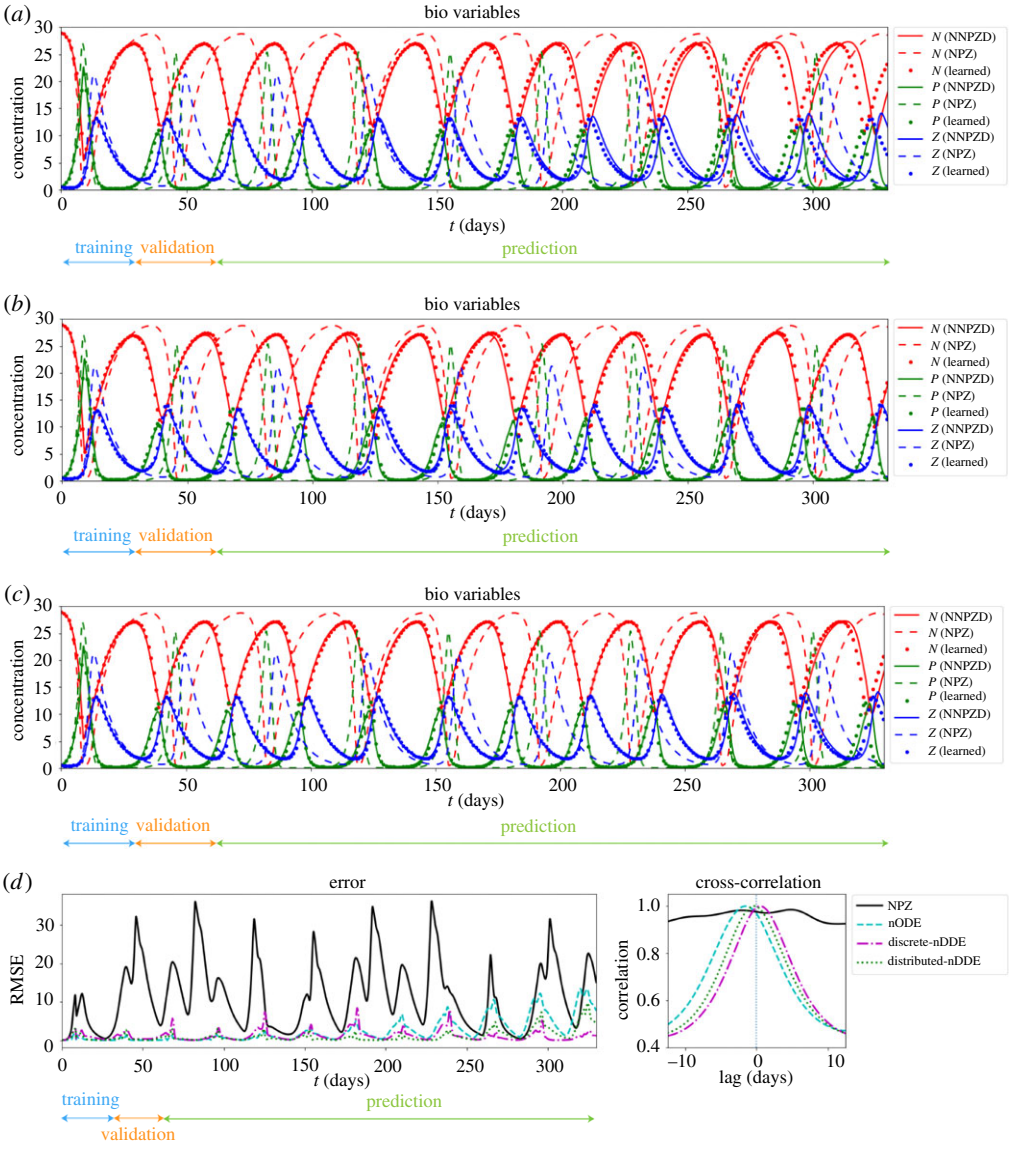

Figure 7. Comparison of the biological variables from the learned NPZ model augmented with the three neural closure models (*dashed*), aggregated variables from the NNPZD model (ground truth; *solid*) and variables from the NPZ model (*dashed-dot*) at the end of training. For each neural closure, the training period is from $t = 0$ to 30 days, the validation period is from $t = 30$ to 60 days, while the prediction period is from $t = 60$ to 330 days. (*a*): Neural closure model with no-delays (nODE); (*b*): neural closure model with discrete-delays (discrete-nDDE); (*c*): neural closure model with distributed-delays (distributed-nDDE); and (*d*): performance comparison of different neural closure models. The *left* plot shows the evolution of root-mean-squared-error (RMSE), and the *right* plot shows the average cross-correlation (only for the prediction period) w.r.t. the ground truth. These results correspond to the architectures detailed in electronic supplementary material, table SI-2. (Online version in colour.)

(d) Experiments 3b: one-dimensional marine biogeochemical models

For our final set of experiments, we extend the ODE models used in Experiments-3a (§4c) to contain a vertical dimension (thus, one-dimensional) and vertical eddy mixing parametrized by the operator, $\partial/\partial z(K_z(z, M)\partial/\partial z(\bullet))$, where K_z is a dynamic eddy diffusion coefficient. A mixed layer of varying depth ($M = M(t)$) is used as a physical input to the ecosystem models. Thus, each

biological state variable $B(z,t)$ is governed by the following non-autonomous PDE:

$$\frac{\partial B}{\partial t} = S^B + \frac{\partial}{\partial z}\left(K_z(z, M(t))\frac{\partial B}{\partial z}\right), \tag{4.10}$$

where S^B are the corresponding biology source terms, which also makes it stiff. The dynamic depth-dependent diffusion parameter K_z is given by

$$K_z(z, M(t)) = K_{z_b} + \frac{(K_{z_0} - K_{z_b})(\arctan(-\gamma(M(t) - z)) - \arctan(-\gamma(M(t) - D)))}{\arctan(-\gamma M(t)) - \arctan(-\gamma(M(t) - D))}, \tag{4.11}$$

where K_{z_b} and K_{z_0} are the diffusion at the bottom and surface, respectively, γ is the thermocline sharpness and D is the total depth. The one-dimensional model and parametrizations are adapted from Eknes & Evensen [94] and Newberger *et al*. [30]. They simulate the seasonal variability in upwelling, sunlight and biomass vertical profiles. The dynamic mixed layer depth, surface photosynthetically available radiation $I_0(t)$ and biomass fields $B(z,t)$ are shown in figure 8*a*. The radiation $I_0(t)$ and total biomass concentration, $T_{\text{bio}}(z,t)$, affect S^B and the initial conditions.

For these Experiments-3b, we consider 20 grid points in the vertical and use the *dopri5* [84] scheme with adaptive time-stepping. Data are stored at every time step of $\Delta t = 0.1$ days for all our time-integration requirements. Solutions of aggregated states of the high-complexity one-dimensional NNPZD model (true data) and their absolute difference with the corresponding low-complexity one-dimensional NPZ model states are provided in figure 8*a*,*b*, respectively. For training our neural closure models for the one-dimensional NPZ model, we use the same training regiment as in Experiments-1, -2 and -3a (§4a–c). We note that in the one-dimensional NPZ model, the local mixing across depths occurs only due to the eddy diffusion term, and not to the biology source terms. Thus, we employ one-dimensional convolutional layers with receptive fields of size 1. We again use the custom layer, *BioConstrainLayer* (§4c), to ensure that the sum of the biology source terms of the augmented one-dimensional NPZ model does not leave any new residual due to the neural closure terms. Along with this, we define a new custom layer, called *AddExtraChannels*, to add additional channels to the input of this layer. We add one for the depths at different grid points, and the other for the corresponding values of available sunlight for photo-chemical reactions ($I(z,t)$). The architecture details for the three closure models used are presented in the electronic supplementary material, table SI-2. For the nODE, we again employ a bigger architecture, and for the discrete-nDDE, only four discrete delay values are used. The values of other hyperparameters are given in the electronic supplementary material, Section SI-4.2. The training period ranges from $t = 0$ to 30 days and validation period from $t = 30$ to 60 days, both within the first season. The prediction period, however, ranges from $t = 60$ to 364 days: it is more than 10 times longer than the training period and involves the four seasons. Together, the three periods span a full year. For loss function, we combine the L_2 errors, considering all the biological states computed for individual depths, and then averaged over all the depths and times, $\mathcal{L} = (1/M)\sum_{i=1}^{M}\left((1/N_z)\sum_{k=1}^{N_z=20}\left(\sqrt{\sum_{B\in\{N,P,Z\}}|B^{\text{pred}}(z_k,T_i) - B^{\text{true}}(z_k,T_i)|^2}\right)\right)$.

The performance of the three neural closure models augmenting the one-dimensional NPZ model is evaluated after 200 epochs of training (the stochastic gradient descent nearly converges, see the electronic supplementary material, figure SI-2d). The truth fields are the aggregated biology variables from the high-complexity one-dimensional NNPZD model (equations (4.9) and (4.10)) spanning training, validation and prediction periods. Results are presented in figure 8. We find again that despite the bigger architecture for the nODE case, it develops spurious oscillations around $t = 250$ days. The discrete-nDDE and distributed-nDDE, both with smaller architectures, however match well with the true variables for nearly the full year of simulation, about 10 months of which is future prediction. The distributed-nDDE performs slightly better than its counterpart. In figure 8, we also provide averaged error numbers for the baseline (figure 8*b*) and the different closure models, all of which improve the baseline. As in Experiments-3a, we again notice large spikes in the starting of the training regime, for the same reason as given earlier and similar trends for hyperparameter sensitivity. We also found that the Experiments-3b were affected by the choice of loss function. For example, using L_2 error computed for each biological state vector

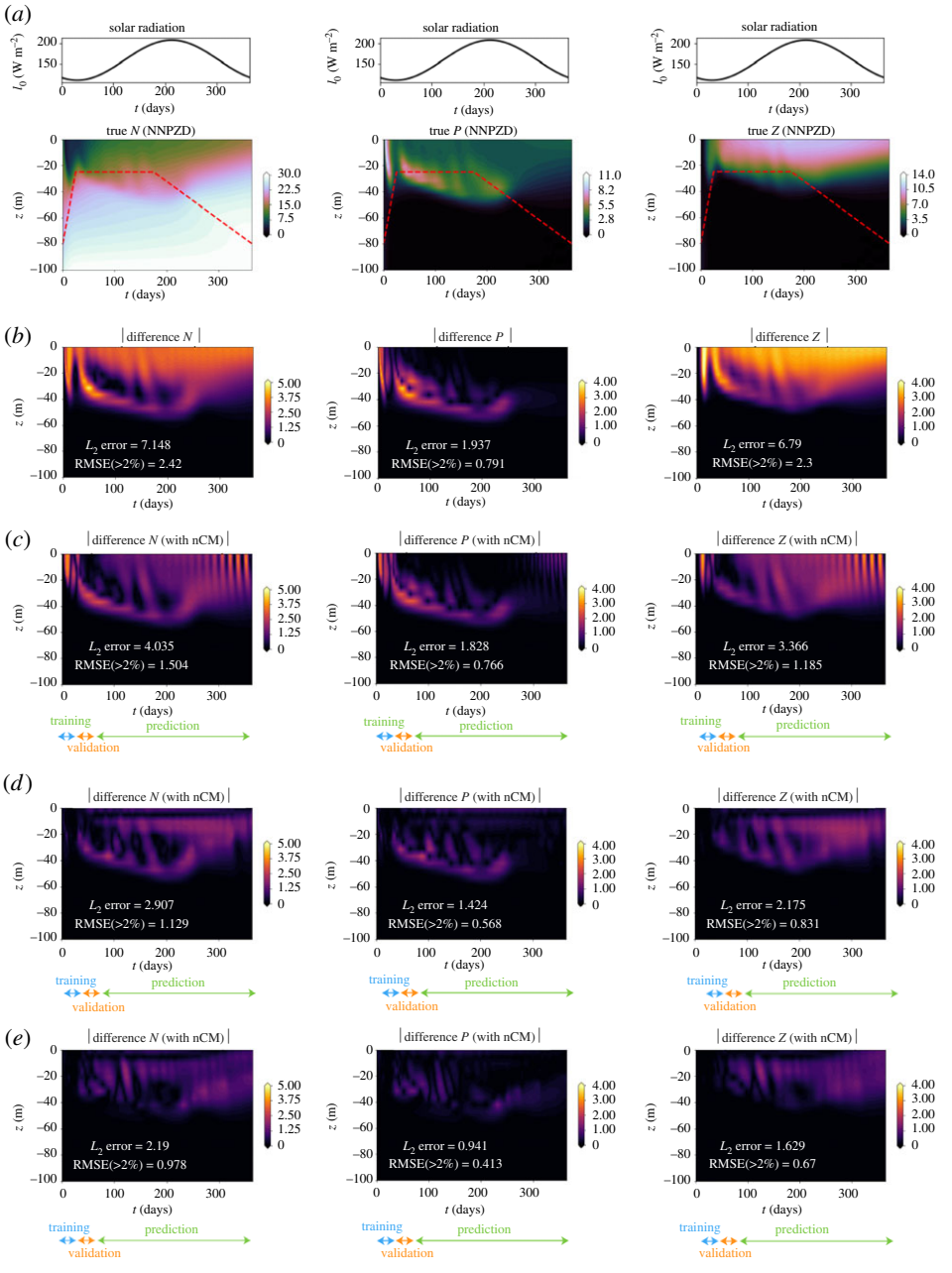

Figure 8. Comparison of the one-dimensional physical-biogeochemical PDE models used in Experiments-3b with and without closure models. Along with the parameter values mentioned in figure 6, we consider: a sinusoidal variation in $I_0(t)$; linear vertical variation in total biomass $T_{\text{bio}}(z)$ from 10 mmol N m^{-3} at the surface to 30 mmol N m^{-3} at $z = 100$ m; $K_{z_b} = 0.0864$ (m^2 d^{-1}); $K_{z_0} = 8.64$ (m^2 d^{-1}); $\gamma = 0.1$ m^{-1}; and $D = -100$ m, all adapted from [30,94]. For the neural closure models, the training period is from $t = 0$ to 30 days, the validation period from $t = 30$ to 60 days, and the long future prediction period from $t =$ 60 to 364 days. (*a*) *Top* plots show the yearly variation of solar radiation and the *bottom* plots the aggregated states from the NNPZD model (*ground truth*) overlaid with the dynamic mixed layer depth in *dashed red* lines. In the subsequent plots we show the absolute difference of the different neural closure cases with the ground truth. (*b*): NPZ model (without neural closure model); (*c*): NPZ model augmented with no-delay neural closure (nODE); (*d*): NPZ model augmented with discrete-delay neural closure (discrete-nDDE); and (*e*): NPZ model augmented with distributed-delay neural closure (distributed-nDDE). For each case, we also provide the pair of time-averaged errors (see figure 3 for description). These results correspond to the architectures given in electronic supplementary material, table SI-2. (Online version in colour.)

(containing values for all the depths) and then averaging over the number of biological states and times deteriorated the quality of learning.

Despite the presence of complex physical processes and relatively large dimensions compared with the previous experiments, the nDDEs closures were found to effectively match the high-complexity model and maintain long-term accuracy.

(e) Computational complexity

It is crucial to analyse complexity and in particular the cost of adding a neural closure model to a low-fidelity model. In this section, we analyse the computational complexity in terms of *flop* (floating point operations) count for evaluating the right-hand side of the low-fidelity models, and for the forward-pass of the neural closure models [95]. We will also comment on the training costs. The Burger's PDE considered for Experiments-1 and -2 (§4a,b) has a nonlinear advection term. Hence, for the POD-GP ROM and the FOM, the upper *flops* is of the order of the square of number of resolved modes and of the spatial grid resolution, respectively. In general, for reaction terms and biogeochemical systems, the number of nonlinear parametrizations present are of the order of the number of components in the model. Hence, even for Experiments-3a (§4c), the upper *flops* is of the order of the square of the number of biological components. For Experiments-3b (§4d), the upper *flops* is also affected by the diffusion terms. Let the number of state variables in the low-fidelity models be $N \in \mathbb{N}$, thus the leading-order computational complexity would be $\mathcal{O}(cN^2)$, where $c \in \mathbb{R}^+$ is some constant dependent on the numerical schemes used for spatial discretization, the exact functional form of the right-hand side, etc.

Now, when neural closure models are added to the low-fidelity models, the time integration requires a forward pass through the NN. This cost varies with the neural architecture and model type, here either a fully connected or convolutional, and discrete-nDDE or distributed-nDDE, respectively. As observed in our experiments, using delays in the closure model enables us to use shallower networks, with a depth independent of the number of state variables (N). We also found that the width of the networks was similar to, or smaller than, N. In the case of distributed-nDDEs, we observed that the width of the auxiliary network (g_{NN}) could be on an average nearly half the size of the main network (f_{NN}). Let the size of the hidden state for the RNN in discrete-nDDEs be $N_h \in \mathbb{N}$, and the number of neurons in the hidden layers of the main and auxiliary networks in the case of distributed-nDDEs be N_h and $N_h/2$, respectively, with $N_h \lesssim N$. It could be easily shown that the leading order complexity for a single iteration of RNN would be $\mathcal{O}(N_h^2 + N_h N)$, which is due to the hidden and input state vectors being multiplied by the weight matrices, while the application of activation function would be $\mathcal{O}(N_h)$ only. As the number of discrete-delays in discrete-nDDEs are independent of N and $\mathcal{O}(1)$, it does not affect the complexity of the RNN. The complexity of the first hidden layer and/or the output layer of the deep neural-networks used in discrete-nDDEs and distributed-nDDEs (main network, f_{NN}) will be $\mathcal{O}(N_h N)$ and $\mathcal{O}(N_h(N + N_h/2))$, respectively, while for the remaining hidden layers, it will be $\mathcal{O}(N_h^2)$. Focusing on the integral of the auxiliary network (g_{NN}) over the delay period in distributed-nDDEs, if implemented efficiently, at every time step, we only need to compute the integral twice over periods of size Δt, each adjacent to the ends of the present delay period. We can add and subtract these integrals over Δt periods to compute the overall integral in a rolling window sense. Hence, the contribution to the computational cost by the auxiliary network would be $\mathcal{O}(N_h N/2)$ (for first hidden layer) and $\mathcal{O}(N_h^2/4)$ (subsequent hidden layers). Considering $D \in \mathbb{N}$ as the depth for all of the networks considered, the complexity for the forward pass through the discrete-nDDE closure is $\mathcal{O}(DN_h^2 + DN_h N)$, while for distributed-nDDE it is $\mathcal{O}((3/2)N_h N + (7/4)DN_h^2)$. These costs were computed considering fully connected layers, however, will only be cheaper in the case of convolutional layers. Thus, the additional computational cost due to the presence of neural closure models is of similar or lower complexity than the existing low-fidelity model.

Estimating the computational cost/complexity of training in *flops* is not common because apart from time-integrating the forward model and adjoint equations, there are many other operations such as here: automatic differentiation through the NNs; creation and use of interpolation functions; the integral to compute the final derivatives; and the gradient descent step, etc.

The overall cost also depends on the number of epochs needed for convergence. The present training cost is of course non-negligible, as with any supervised learning algorithm. However, in applications where one needs to repeatedly solve a low-fidelity model, investing in a one-time cost of training a neural closure model can later lead to accuracy close to that of the high-fidelity model with only a small increase in the computational cost of the low-fidelity model.

5. Conclusion

We developed a novel, versatile, rigorous and unified methodology to learn closure parametrizations for low-fidelity models using data from high-fidelity simulations. The MZ formulation [13–15] and the presence of inherent delays in complex dynamical systems [96], especially biological systems [25,26], justify the need for non-Markovian closure parametrizations. To learn such non-Markovian closures, our new *neural closure models* extend nODEs [20] to nDDEs. Our nDDEs do not require access to the high-fidelity model or frequent enough and uniformly spaced high-fidelity data to compute the time derivative of the state with high accuracy. Further, it enables the accounting of errors in the time evolution of the states in the presence of NNs during training. We derive the adjoint equations and network architectures needed to efficiently implement the nDDEs, for both discrete and distributed delays, agnostic to the specifics of the time-integration scheme, and capable of handling stiff systems. For distributed-delays, we propose a novel architecture consisting of two coupled deep NNs, which enables us to incorporate memory without the use of any recurrent architectures.

Through simulation experiments, we showed that our methodology drastically improves the long-term predictive capability of low-fidelity models for the main classes of model truncations. Specifically, our neural closure models efficiently account for truncated modes in ROMs, capture the effects of subgrid-scale processes in coarse models and augment the simplification of complex biological and non-autonomous physical–biogeochemical models. Our first two classes of simulation experiments use the advecting shock problem governed by Burger's PDE, with its low-fidelity models derived either by POD-GP or by reducing the spatial grid resolution. Our third class of experiments considers marine biological ODEs of varying complexities and their physical–biogeochemical PDE extensions with non-autonomous dynamic parametrizations. The low-fidelity models are obtained by aggregation of components and other simplifications of processes and parametrizations. In each of these classes, results consistently show that using non-Markovian over Markovian closures improves the accuracy of the learned system while also requiring smaller network architectures. Our use of the known physics/low-fidelity model also helps to reduce the required size of the network architecture and the number of time samples for the training data. We also outperform classic dynamic closures such as the Smagorinsky subgrid-scale model. These results are obtained using stringent evaluations: we compare the performance of the learned system for the training period (during which high-fidelity data snapshots are used for training) and validation period (during which hyperparameter tuning occurs) as often done, but we also compare it for much longer-term future prediction periods with no overlap with the preceding two. We even consider a prediction period reaching 10 times the length of the training/validation period, thus successfully demonstrating the extrapolation capabilities of nDDE closures.

In our experiments, we find that just using a few numbers of discrete delays might perform equally well or better than using a distributed delay, which involves an integral of the state variable over a delay period. We provide a plausible explanation of this counterintuitive observation using the data processing inequality from information theory. We also show that there exists an optimal amount of past information to incorporate for a specified architecture and the relevant time scales present in the dynamical system, thus indicating that neither too little nor too much past information is helpful. Finally, a computational complexity analysis using *flop* (floating point operation) count proves that the additional computational cost due to the presence of our neural closure models is of similar or lower complexity than the existing low-fidelity model.

The present work provides a unified framework to learn non-Markovian closure parametrization using DDEs and NNs. It enables the use of the often elusive MZ formulation [13–15] in its full glory without unjustified approximations and simplifications. Our nDDE closures are not just limited to the shown experiments, but could be widely extended to other fields such as control theory, robotics, pharmacokinetic–pharmacodynamics, chemistry, economics and biological regulatory systems, etc.

Data accessibility. The codes and data used in this work are available in the GitHub repository: https://github.com/mit-mseas/neuralClosureModels.git.

Authors' contributions. A.G. conceived the idea of using nDDEs for closure parametrizations; derived the adjoint equations; implemented the NN architectures and the simulation experiments; interpreted the computational results; and wrote a first draft of the manuscript. P.F.J.L. supervised the work; conceived the ideas to extend the methodology to capture the effects of subgrid-scale processes in coarse models, and to augment the simplification of complex mathematical models; interpreted the computational results; and edited and wrote significant parts of the manuscript.

Competing interests. We declare we have no competing interests.

Funding. We are grateful to the Office of Naval Research for partial support under grant no. N00014-20-1-2023 (MURI ML-SCOPE) to the Massachusetts Institute of Technology. We also thank MathWorks and the Mechanical Engineering Department at MIT for awarding a competitive 2020–2021 MathWorks Mechanical Engineering Fellowship for A.G. Some of the computations were made possible due to the Google Cloud Platform research credits, which we gratefully acknowledge.

Acknowledgements. We thank the members of our MSEAS group for their collaboration and insights, especially Mr Aaron Charous. We thank our ML-SCOPE team for many useful discussions, especially Dr Mickaël Chekroun for his comments on the final draft of the manuscript. We also thank the anonymous reviewers for their constructive feedback which helped improve the manuscript.

References

1. Kutz JN, Brunton SL, Brunton BW, Proctor JL. 2016 *Dynamic mode decomposition: data-driven modeling of complex systems*. Philadelphia, PA: SIAM.
2. Wang Z, Akhtar I, Borggaard J, Iliescu T. 2012 Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison. *CMAME* **237**, 10–26. (doi:10.1016/j.cma.2012.04.015)
3. Alfonsi G. 2009 Reynolds-averaged navier–stokes equations for turbulence modeling. *Appl. Mech. Rev.* **62**, 040802. (doi:10.1115/1.3124648)
4. Lesieur M, Métais O, Comte P. 2005 *Large-eddy simulations of turbulence*. Cambridge, UK: Cambridge University Press.
5. Chassignet EP, Verron J. 2012 *Ocean modeling and parameterization*, vol. 516. Berlin, Germany: Springer Science & Business Media.
6. Los F, Blaas M. 2010 Complexity, accuracy and practical applicability of different biogeochemical model versions. *J. Mar. Sys.* **81**, 44–74. (doi:10.1016/j.jmarsys.2009.12.011)
7. Ward BA, Schartau M, Oschlies A, Martin AP, Follows MJ, Anderson TR. 2013 When is a biogeochemical model too complex? objective model reduction and selection for North Atlantic time-series sites. *Prog. Oceanogr.* **116**, 49–65. (doi:10.1016/j.pocean.2013.06.002)
8. Pan S, Duraisamy K. 2018 Data-driven discovery of closure models. *SIAM J. Appl. Dyn. Syst.* **17**, 2381–2413. (doi:10.1137/18M1177263)
9. Pawar S, Ahmed SE, San O, Rasheed A. 2020 Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Phys. Fluids* **32**, 036602. (doi:10.1063/5.0002051)
10. San O, Maulik R. 2018 Neural network closures for nonlinear model order reduction. *Adv. Comput. Math.* **44**, 1717–1750. (doi:10.1007/s10444-018-9590-z)
11. Wan ZY, Vlachas P, Koumoutsakos P, Sapsis T. 2018 Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLoS ONE* **13**, e0197704. (doi:10.1371/journal.pone.0197704)
12. Wang Q, Ripamonti N, Hesthaven JS. 2020 Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism. *J. Comput. Phys.* **410**, 109402. (doi:10.1016/j.jcp.2020.109402)

13. Chorin AJ, Hald OH, Kupferman R. 2000 Optimal prediction and the Mori–Zwanzig representation of irreversible processes. *Proc. Natl Acad. Sci. USA* **97**, 2968–2973. (doi:10.1073/pnas.97.7.2968)
14. Gouasmi A, Parish EJ, Duraisamy K. 2017 A priori estimation of memory effects in reduced-order models of nonlinear systems using the Mori–Zwanzig formalism. *Proc. R. Soc. A* **473**, 20170385. (doi:10.1098/rspa.2017.0385)
15. Stinis P. 2015 Renormalized Mori–Zwanzig-reduced models for systems without scale separation. *Proc. R. Soc. A* **471**, 20140446. (doi:10.1098/rspa.2014.0446)
16. Whitney H. 1936 Differentiable manifolds. *Ann. Math.* **37**, 645–680. (doi:10.2307/1968482)
17. Takens F. 1981 Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980* (eds D Rand, LS Young), pp. 366–381. Berlin, Germany: Springer.
18. Brunton SL, Proctor JL, Kutz JN. 2016 Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937. (doi:10.1073/pnas.1517384113)
19. Kulkarni CS, Gupta A, Lermusiaux PF. 2020 Sparse regression and adaptive feature generation for the discovery of dynamical systems. In *Int. Conf. on Dynamic Data Driven Application Systems* (eds F Darema, E Blasch, S Ravela, A Aved), pp. 208–216. Berlin, Germany: Springer.
20. Chen TQ, Rubanova Y, Bettencourt J, Duvenaud DK. 2018 Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583.
21. Maulik R, Mohan A, Lusch B, Madireddy S, Balaprakash P, Livescu D. 2020 Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D* **405**, 132368. (doi:10.1016/j.physd.2020.132368)
22. Yang Y, Aziz Bhouri M, Perdikaris P. 2020 Bayesian differential programming for robust systems identification under uncertainty. *Proc. R. Soc. A* **476**, 20200290. (doi:10.1098/rspa.2020.0290)
23. Portwood GD *et al.* 2019 Turbulence forecasting via neural ODE. (http://arxiv.org/abs/1911.05180)
24. Otto A, Just W, Radons G. 2019 Nonlinear dynamics of delay systems: an overview. *Phil. Trans. R. Soc. A* **377**, 20180389. (doi:10.1098/rsta.2018.0389)
25. Glass DS, Jin X, Riedel-Kruse IH. 2021 Nonlinear delay differential equations and their application to modeling biological network motifs. *Nat. Commun.* **12**, 1–19. (doi:10.1038/s41467-021-21700-8)
26. Tokuda IT, Akman OE, Locke JC. 2019 Reducing the complexity of mathematical models for the plant circadian clock by distributed delays. *J. Theor. Biol.* **463**, 155–166. (doi:10.1016/j.jtbi.2018.12.014)
27. Behzad F, Helenbrook BT, Ahmadi G. 2015 On the sensitivity and accuracy of proper-orthogonal-decomposition-based reduced order models for Burgers equation. *Comput. Fluids* **106**, 19–32. (doi:10.1016/j.compfluid.2014.09.041)
28. Borja A *et al.* 2014 Tales from a thousand and one ways to integrate marine ecosystem components when assessing the environmental status. *Front. Mar. Sci.* **1**, 72. (doi:10.3389/fmars.2014.00072)
29. Fennel W, Neumann T. 2014 *Introduction to the modelling of marine ecosystems*. Amsterdam, The Netherlands: Elsevier.
30. Newberger PA, Allen JS, Spitz YH. 2003 Analysis and comparison of three ecosystem models. *J. Geophys. Res.: Oceans (1978–2012)* **108**, 3061. (doi:10.1029/2001JC001182)
31. Lermusiaux PFJ, Malanotte-Rizzoli P, Stammer D, Carton J, Cummings J, Moore AM. 2006 Progress and prospects of U.S. data assimilation in ocean research. *Oceanography* **19**, 172–183. (doi:10.5670/oceanog.2006.102)
32. Lermusiaux PFJ *et al.* 2006 Quantifying uncertainties in ocean predictions. *Oceanography* **19**, 92–105. (doi:10.5670/oceanog.2006.93)
33. Robinson AR, Haley PJ, Lermusiaux PFJ, Leslie WG. 2002 Predictive skill, predictive capability and predictability in ocean forecasting. In *Proc. of 'The OCEANS 2002 MTS/IEEE' Conference, Biloxi, MI, 29–31 October*, pp. 787–794. IEEE.
34. Robinson NM, Nelson WA, Costello MJ, Sutherland JE, Lundquist CJ. 2017 A systematic review of marine-based species distribution models (SDMs) with recommendations for best practice. *Front. Mar. Sci.* **4**, 421. (doi:10.3389/fmars.2017.00421)
35. Feppon F, Lermusiaux PFJ. 2018 A geometric approach to dynamical model-order reduction. *SIAM J. Matrix Anal. Appl.* **39**, 510–538. (doi:10.1137/16M1095202)

36. Holmes P, Lumley JL, Berkooz G, Rowley CW. 2012 *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge, UK: Cambridge University Press.
37. Feppon F, Lermusiaux PFJ. 2018 Dynamically orthogonal numerical schemes for efficient stochastic advection and Lagrangian transport. *SIAM Rev.* **60**, 595–625. (doi:10.1137/16M1109394)
38. Sapsis TP, Lermusiaux PFJ. 2012 Dynamical criteria for the evolution of the stochastic dimensionality in flows with uncertainty. *Physica D* **241**, 60–76. (doi:10.1016/j.physd.2011.10.001)
39. Matthies HG, Meyer M. 2003 Nonlinear Galerkin methods for the model reduction of nonlinear dynamical systems. *Comput. Struct.* **81**, 1277–1286. (doi:10.1016/S0045-7949(03)00042-7)
40. Laizet S, Nedić J, Vassilicos C. 2015 Influence of the spatial resolution on fine-scale features in DNS of turbulence generated by a single square grid. *Int. J. Comput. Fluid Dyn.* **29**, 286–302. (doi:10.1080/10618562.2015.1058371)
41. Yeung P, Sreenivasan KR, Pope SB. 2018 Effects of finite spatial and temporal resolution in direct numerical simulations of incompressible isotropic turbulence. *Phys. Rev. Fluids* **3**, 064603. (doi:10.1103/PhysRevFluids.3.064603)
42. Dauhajre DP, McWilliams JC, Renault L. 2019 Nearshore lagrangian connectivity: submesoscale influence and resolution sensitivity. *JGR: Oceans* **124**, 5180–5204. (doi:10.1029/2019JC014943)
43. McWilliams JC. 2016 Submesoscale currents in the ocean. *Proc. R. Soc. A* **472**, 20160117. (doi:10.1098/rspa.2016.0117)
44. McWilliams JC. 2017 Submesoscale surface fronts and filaments: secondary circulation, buoyancy flux, and frontogenesis. *J. Fluid Mech.* **823**, 391. (doi:10.1017/jfm.2017.294)
45. Schneider T, Lan S, Stuart A, Teixeira J. 2017 Earth system modeling 2.0: a blueprint for models that learn from observations and targeted high-resolution simulations. *Geophy. Res. L.* **44**, 12–396. (doi:10.1002/2016GL071741)
46. May RM. 2019 *Stability and complexity in model ecosystems*, vol. 1. Princeton, NJ: Princeton University Press.
47. Nowak MA. 2006 *Evolutionary dynamics: exploring the equations of life*. Cambridge, UK: Harvard University Press.
48. Robinson AR, Lermusiaux PFJ. 2002 Data assimilation for modeling and predicting coupled physical–biological interactions in the sea. In *Biological-Physical Interactions in the Sea* (eds J McCarthy, BJ Rothschild), vol. 12 of *The Sea*, ch. 12, pp. 475–536. New York, NY: John Wiley and Sons.
49. Lermusiaux PFJ. 2007 Adaptive modeling, adaptive data assimilation and adaptive sampling. *Physica D* **230**, 172–196. (doi:10.1016/j.physd.2007.02.014)
50. Lermusiaux PFJ, Evangelinos C, Tian R, Haley PJ, McCarthy JJ, Patrikalakis NM, Robinson AR, Schmidt H. 2004 Adaptive coupled physical and biogeochemical ocean predictions: a conceptual basis. In *Computational Science - ICCS 2004*, vol. 3038 of *Lecture Notes in Computer Science* (eds M Bubak, GD van Albada, PMA Sloot, J Dongarra), pp. 685–692. Berlin, Germany: Springer.
51. Dell'Anna L. 2020 Solvable delay model for epidemic spreading: the case of Covid-19 in Italy. *Sci. Rep.* **10**, 1–10. (doi:10.1038/s41598-019-56847-4)
52. Kuang Y. 1993 *Delay differential equations: with applications in population dynamics*. Academic Press.
53. Bocharov GA, Rihan FA. 2000 Numerical modelling in biosciences using delay differential equations. *J. Comput. Appl. Math.* **125**, 183–199. (doi:10.1016/S0377-0427(00)00468-4)
54. Faugeras B, Maury O. 2007 Modeling fish population movements: from an individual-based representation to an advection–diffusion equation. *J. Theor. Bio.* **247**, 837–848. (doi:10.1016/j.jtbi.2007.04.012)
55. Hundsdorfer W, Verwer JG. 2013 *Numerical solution of time-dependent advection-diffusion-reaction equations*, vol. 33. Berlin, Germany: Springer Science & Business Media.
56. Boers N, Chekroun MD, Liu H, Kondrashov D, Rousseau DD, Svensson A, Bigler M, Ghil M. 2017 Inverse stochastic–dynamic models for high-resolution greenland ice core records. *Earth Syst. Dyn.* **8**, 1171–1190. (doi:10.5194/esd-8-1171-2017)
57. Kondrashov D, Chekroun MD, Ghil M. 2015 Data-driven non-Markovian closure models. *Physica D* **297**, 33–55. (doi:10.1016/j.physd.2014.12.005)

58. Chekroun MD, Liu H, McWilliams JC. 2020 Variational approach to closure of nonlinear dynamical systems: autonomous case. *J. Stat. Phys.* **179**, 1073–1160. (doi:10.1007/s10955-019-02458-2)
59. Richard J-P. 2003 Time-delay systems: an overview of some recent advances and open problems. *Automatica* **39**, 1667–1694. (doi:10.1016/S0005-1098(03)00167-5)
60. Diekmann O, Van Gils SA, Lunel SM, Walther H-O. 2012 *Delay equations: functional-, complex-, and nonlinear analysis*, vol. 110. Berlin, Germany: Springer Science & Business Media.
61. MacDonald N. 2008 *Biological delay systems: linear stability theory*. Cambridge, UK: Cambridge University Press.
62. Smith HL. 2011 *An introduction to delay differential equations with applications to the life sciences*, vol. 57. New York, NY: Springer.
63. Koch G, Krzyzanski W, Pérez-Ruixo JJ, Schropp J. 2014 Modeling of delays in PKPD: classical approaches and a tutorial for delay differential equations. *J. Pharmacokinet. Pharmacodyn.* **41**, 291–318. (doi:10.1007/s10928-014-9368-y)
64. Roussel MR. 1996 The use of delay differential equations in chemical kinetics. *J. Phys. Chem.* **100**, 8323–8330. (doi:10.1021/jp9600672)
65. Keller AA. 2010 Generalized delay differential equations to economic dynamics and control. *American-Math* **10**, 278–286.
66. Matsuya K, Kanai M. 2015 Exact solution of a delay difference equation modeling traffic flow and their ultra-discrete limit. (http://arxiv.org/abs/1509.07861).
67. Kunisch K. 1982 Approximation schemes for the linear-quadratic optimal control problem associated with delay equations. *SIAM J. Control Optim.* **20**, 506–540. (doi:10.1137/0320038)
68. Bhattacharya K, Ghil M, Vulis I. 1982 Internal variability of an energy-balance model with delayed albedo effects. *J. Atmos. Sci.* **39**, 1747–1773. (doi:10.1175/1520-0469(1982)039<1747:IVOAEB>2.0.CO;2)
69. Ghil M, Chekroun MD, Stepan G. 2015 A collection on 'climate dynamics: multiple scales and memory effects. *Proc. R. Soc. A* **471**, 20150097. (doi:10.1098/rspa.2015.0097)
70. Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, Skinner D, Ramadhan A, Edelman A. 2020 Universal differential equations for scientific machine learning. (http://arxiv.org/abs/2001.04385).
71. Rackauckas C, Ma Y, Martensen J, Warner C, Zubov K, Supekar R, Skinner D, Ramadhan A, Edelman A. 2019 Diffeqflux.jl - A julia library for neural differential equations. (http://arxiv.org/abs/1902.02376).
72. Abadi M *et al.* 2015 TensorFlow: Large-scale machine learning on heterogeneous systems. See www.tensorflow.org/.
73. Paszke A *et al.* 2019 Pytorch: an imperative style, high-performance deep learning library. In *Advances in neural information processing systems, 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, pp. 8026–8037.
74. Rubanova Y, Chen RT, Duvenaud D. 2019 Latent odes for irregularly-sampled time series. (http://arxiv.org/abs/1907.03907).
75. Calver J, Enright W. 2017 Numerical methods for computing sensitivities for ODEs and DDEs. *Numer. Algorithms* **74**, 1101–1117. (doi:10.1007/s11075-016-0188-6)
76. Robinson AR, Lermusiaux PFJ, Sloan III NQ. 1998 Data assimilation. In *The Global Coastal Ocean-Processes and Methods*, vol. 10 of *The Sea*, ch. 20, pp. 541–594. New York, NY: John Wiley and Sons.
77. Wunsch C. 1996 *The ocean circulation inverse problem*. Cambridge, UK: Cambridge University Press.
78. Gholami A, Keutzer K, Biros G. 2019 Anode: Unconditionally accurate memory-efficient gradients for neural odes. (http://arxiv.org/abs/1902.10298).
79. Griewank A. 1992 Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optim. Methods Softw.* **1**, 35–54. (doi:10.1080/10556789208805505)
80. Daulbaev T, Katrutsa A, Markeeva L, Gusak J, Cichocki A, Oseledets I. 2020 Interpolation technique to speed up gradients propagation in neural ODEs. *Adv. Neural Inform. Process. Systems* **33**.
81. Rasmussen H, Wake G, Donaldson J. 2003 Analysis of a class of distributed delay logistic differential equations. *Math. Comput. Modell.* **38**, 123–132. (doi:10.1016/S0895-7177(03)90010-0)
82. Rackauckas C *et al.* SciML Scientific Machine Learning Software. See https://sciml.ai/.

83. Yuval J, Hill CN, O'Gorman PA. 2020 Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. (http://arxiv.org/abs/2010.09947).
84. Hairer E, Nørsett SP, Wanner G. 1993 *Solving ordinary differential equations I*. Berlin, Germany: Springer.
85. Cover TM. 1999 *Elements of information theory*. New York, NY: John Wiley & Sons.
86. Brown PN, Byrne GD, Hindmarsh AC. 1989 VODE: A variable-coefficient ODE solver. *SIAM J. Sci. Stat. Comput.* **10**, 1038–1051. (doi:10.1137/0910062)
87. Maulik R, San O. 2018 Explicit and implicit les closures for burgers turbulence. *J. Comput. Appl. Math.* **327**, 12–40. (doi:10.1016/j.cam.2017.06.003)
88. Li J, Stinis P. 2019 Mori-Zwanzig reduced models for uncertainty quantification. *J. Comput. Dyn.* **6**, 39–68.
89. Burchard H, Deleersnijder E, Meister A. 2005 Application of modified patankar schemes to stiff biogeochemical models for the water column. *Ocean Dyn.* **55**, 326–337. (doi:10.1007/s10236-005-0001-x)
90. Beşiktepe ŞT, Lermusiaux PFJ, Robinson AR. 2003 Coupled physical and biogeochemical data-driven simulations of Massachusetts Bay in late summer: real-time and post-cruise data assimilation. *J. Mar. Syst.* **40–41**, 171–212. (doi:10.1016/S0924-7963(03)00018-6)
91. Lermusiaux PFJ *et al.* 2011 Multiscale physical and biological dynamics in the Philippine Archipelago: predictions and processes. *Oceanography* **24**, 70–89. (doi:10.5670/oceanog.2011.05)
92. Ueckermann MP, Lermusiaux PFJ. 2010 High order schemes for 2D unsteady biogeochemical ocean models. *Ocean Dyn.* **60**, 1415–1445. (doi:10.1007/s10236-010-0351-x)
93. Baretta J, Ebenhöh W, Ruardij P. 1995 The European regional seas ecosystem model, a complex marine ecosystem model. *Neth. J. Sea Res.* **33**, 233–246. (doi:10.1016/0077-7579(95)90047-0)
94. Eknes M, Evensen G. 2002 An ensemble Kalman filter with a 1-D marine ecosystem model. *J. Mar. Sys.* **36**, 75–100. (doi:10.1016/S0924-7963(02)00134-3)
95. Mizutani E, Dreyfus SE. 2001 On complexity analysis of supervised MLP-learning for algorithmic comparisons. In *IJCNN'01. Int. Joint Conf. on Neural Networks. Proc. (Cat. No. 01CH37222), Washington, DC, 15–19 July*, vol. 1, pp. 347–352. IEEE.
96. Erneux T. 2009 *Applied delay differential equations*, vol. 3. Berlin, Germany: Springer.