Computers and Chemical Engineering 178 (2023) 108343

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Some of the variables, some of the parameters, some of the times, with some physics known: Identification with partial information

Saurabh Malani [a], Tom S. Bertalan [b], Tianqi Cui [b], José L. Avalos [a,d,e,f,g], Michael Betenbaugh [b], Ioannis G. Kevrekidis [b,c,*]

[a] Department of Chemical and Biological Engineering, Princeton University, United States of America
[b] Department of Chemical and Biomolecular Engineering, Whiting School of Engineering, Johns Hopkins University, United States of America
[c] Department of Applied Mathematics, Whiting School of Engineering, Johns Hopkins University, United States of America
[d] The Omenn-Darling Bioengineering Institute, Princeton University, United States of America
[e] Andlinger Center for Energy and the Environment, Princeton University, United States of America
[f] Department of Molecular Biology, Princeton University, United States of America
[g] High Meadows Environmental Institute, Princeton University, United States of America

ARTICLE INFO

Keywords:
System identification
Dynamical systems
Partial information
Recurrent neural networks
Gray boxes

ABSTRACT

Experimental data is often comprised of variables measured independently, at different sampling rates (non-uniform Δt between successive measurements); and at a specific time point only a subset of all variables may be sampled. Approaches to identifying dynamical systems from such data typically use interpolation, imputation or subsampling to reorganize or modify the training data *prior* to learning. Partial physical knowledge may also be available *a priori* (accurately or approximately), and data-driven techniques can complement this knowledge. Here we exploit neural network architectures based on numerical integration methods and *a priori* physical knowledge to identify the right-hand side of the underlying governing differential equations. Iterates of such neural-network models allow for learning from data sampled at arbitrary time points *without* data modification. Importantly, we integrate the network with available partial physical knowledge in "physics informed gray-boxes"; this enables learning unknown kinetic rates or microbial growth functions while simultaneously estimating experimental parameters.

1. Introduction

Dynamical systems arise in all areas of science and engineering, from spatiotemporally chaotic or turbulent flows to transient chemical reactor dynamics, complex biological systems, infectious disease models and beyond. These dynamical systems obey fundamental physical or mathematical governing laws, certain components of which may not be known *a priori*, or may only be known approximately: we may have a known functional form but lack certain parameter values (e.g. kinetic constants); or alternatively, we may not even exactly know the right functional form (e.g. the right kinetics). System identification is the science of using experimental data to extract information/knowledge about the underlying dynamical system or possibly to deduce a useful surrogate model; such models can be used for prediction, extrapolation of the system behavior beyond the training data range, optimization of its performance, or for controlling it to a desired setpoint. This is of course an established and rich branch of systems theory, with many successful applications; yet the recent explosion in machine learning algorithms has caused a resurgence of interest in several issues arising when learning models from time-series data—including the lack of interpretability, ability to successfully generalize, and missing data and partial observations.

Many applications of neural ordinary differential equations (Neural ODEs) and Recurrent Neural Networks (RNNs) focus on learning functionals of entire time-series sequences, e.g. classification of ECG signals to heartbeat types (Saadatnejad et al., 2020; Yildirim, 2018). In this paper our focus is on learning autonomous dynamical systems from partial information, although our approach can be extended to non-autonomous systems.

Before starting, we state here our definitions for certain terms as used in this work: (a) **Full Observations**: At each sampling time, *all* relevant dependent variables are measured; (b) **Partial Observations**: At each sampling time, *a subset of* the variables is measured; (c) **Hidden Variables**: some variables are *never* measured at *any* time; (d) **Data** Δt

* Correspondence to: Department of Chemical and Biomolecular Engineering, Whiting School of Engineering, Johns Hopkins University, 3400 North Charles Street, Baltimore, MD21218, USA.
E-mail address: yannisk@jhu.edu (I.G. Kevrekidis).

https://doi.org/10.1016/j.compchemeng.2023.108343
Received 27 April 2023; Received in revised form 20 June 2023; Accepted 22 June 2023
Available online 17 July 2023

is the time interval between successive sampling times; (e) **Learning** δt is our estimate of an integration time step long enough to be useful, but also short enough to be accurate; (f) **Fixed Frequency Sampling**: Variables are sampled at a consistent sampling frequency (Data Δt is fixed); and (g) **Variable Frequency Sampling**: Variables are sampled at arbitrary sampling frequencies (Data Δt is variable).

System identification is traditionally a data-driven process—using mathematical, statistical or machine learning techniques on input–output data to extract information about the underlying system laws. This branch of systems theory has a long history, from the celebrated Kalman filter (Kalman, 1960) (a linear method); non-linear variants such as the extended Kalman filter (Ljung, 1979), the unscented Kalman filter (Wan and Van Der Merwe, 2000), and the particle filter (Gordon et al., 1993); to new techniques such as neural networks (Nelles, 2001; Wang, 2017; Kuschewski et al., 1993; Chen et al., 1990; Hudson et al., 1990; Rico-Martinez et al., 1992). Empirical data, however, while being an essential component of the task, is not the only source of information typically available about the systems of interest. Governing equations, conservation laws, symmetries, physical theories and/or models, even when imperfect or incomplete, also constitute crucial pieces of information. In our work here we endeavor to take into account all such sources of information, developing a hybrid gray-box model that combines black-box data driven techniques with white-box physics models.

Related approaches are becoming increasingly successful (and popular in recent literature), for example through Physics-Informed Neural Networks (PINNs) (Raissi et al., 2017) where physical knowledge, often in the form of partial differential equations (Raissi et al., 2019)—possibly with unknown parameters, or even with unknown operators (Lu et al., 2021)—is enforced while using deep-learning techniques. However, PINNs are typically used to find solutions consistent with a fully known underlying governing ODE/PDE; our approach centers around using known physically relevant solutions (experimental data) to identify (parts of) the underlying system itself. While the term "PINN" has come to refer to the forwards (problem solution) task, simultaneous work (Raissi et al., 2017) also considered the inverse identification task, albeit without gray-box structure. This was then followed by much more general operator-learning work (Lu et al., 2021), in which general functional operators are learned, including integration; older versions of this operator-learning tasks can be found for example in González-García et al. (1998). Performing such a "black box" task using machine learning, and, in particular, neural networks is a research direction that started in the late 1980s and early 1990s (Farber et al., 1993; Rico-Martinez et al., 1992; Purwar et al., 2007), and has been rejuvenated and much more widely used with the advent of neural ODEs or ODE-Nets (Chen et al., 2018; Krishnapriyan et al., 2022).

When partial information about the right-hand-side of the underlying differential equations is known, the "gray-box" version of this identification process uses the expressivity of neural networks to *correct for incompleteness or inaccuracies* of partially/approximately known physical models, or to impose additional structure or known global laws or constraints. These corrections can be performed in an additive (e.g. Rico-Martinez et al. (1994), Lee et al. (2022), Kemeth et al. (2022), Menesklou et al. (2021), Thompson and Kramer (1994), Shi et al. (2019)) or multiplicative (e.g. Lovelett et al. (2020), Chen et al. (2000)) manner, or the neural network can be integrated with physics in a functional manner (e.g. Kemeth et al. (2022), Psichogios and Ungar (1992), Hagge et al. (2017), De Veaux et al. (1999), Oliveira (2004), Van Can et al. (1997)).

In this paper we will focus on identifying both missing parameters and missing functional terms: known governing principles such as mass/energy balances or couplings are imposed by the formulation, and the black-box neural network learns physically relevant functions such as microbial growth rates or chemical reaction rates, and simultaneously estimates possibly unknown parameter values. Note that this is in contrast to methods such as Hamiltonian Neural Networks (Bertalan et al., 2019; Greydanus et al., 2019) where a structural feature of the dynamics (symplecticity, or their Hamiltonian nature) is imposed *in addition* to the demand for accurate prediction.

In placing our work in the context of current literature, we consider distinct categories of analogous efforts: first those that are focusing on black-box models trained on data with missing or partial information; second on black-box models trained on data at variable sampling rates; and last, those that are focusing on partially known physics (gray-box models).

Black Box Models trained on data with missing or partial information. Measurements of experimental data often are not available in a format ideal for data-driven prediction/learning. Sensors may not operate at regular intervals, manually sampled data may also not be available at regular intervals, and some data points may be corrupted by external noise. This results in a dataset that is sampled at *variable frequencies*. Furthermore, different sensors may operate at different sampling rates; and at each sample point only some analyses may be run. This results in a dataset with *partial observations*—where at each sample point, only a subset of all variables are measured.

To make such training data conducive to statistical analyses and machine learning, pre-processing is often performed; this may be as simplistic as omitting the "anomalous" datapoints (Lyngdoh et al., 2022), but more often some form of imputation is involved. This could can be linear-interpolation or a "last observation carried forward" approach (Gelman and Hill, 2006; Oluwaseye et al., 2022), may involve more robust smoothing techniques such as polynomial or LOESS smoothing (locally estimated scatterplot smoothing) (Honaker and King, 2010), machine learning techniques such as expectation maximization (Nelwamondo et al., 2007), k-nearest neighbors (Bertsimas et al., 2021; Silva-Ramírez et al., 2015; Oluwaseye et al., 2022; Lyngdoh et al., 2022) or artificial neural networks (Silva-Ramírez et al., 2015; Nelwamondo et al., 2007) for smoothing and imputation.

Methodologies that impute missing data *first* and train a model on the cleaned dataset *second* often reinforce the imputed values *as if they were true*: "if you fill the holes in the cheese with peanut butter, you should not pretend to have more cheese!" (Honaker and King, 2010). This can be mitigated by conducting imputation and model training *simultaneously*: using the model to impute missing data, further training the model using the data (but not including the loss at the imputed (not ground-truth) values); using the improved model to impute again, and iterating.

Black Box Models trained on data sampled at variable time points. To train on data sampled at variable sampled time points, we need a method that can naturally allow for arbitrary time stepping. The simplest supervised learning approach involves learning a direct mapping between the current state $x(t)$ and the state $x(t+\Delta t)$ at a time Δt in the future, but this method either requires training on data that is sampled at regular Δt intervals, or requires the Δt itself also be an input of the network (Fig. 1(a)). An alternative (and in our opinion more flexible) approach consists of recognizing that the underlying systems can be represented as differential equations in continuous time. Hence, if we instead have the neural network learn *the form of the right hand side of a differential equation system*, we can leverage well-established numerical integrator methods to generate predictions at any desired time point (Fig. 1(b)). This method was proposed in the literature as far back as the 1990's (Rico-Martinez et al., 1992; Rico-Martinez and Kevrekidis, 1993) and has been extended and popularized in recent work through neural ODEs (Chen et al., 2018). This recent work in Neural ODEs has allowed for compatibility with built-in, robust numerical integrators in Python, with gradients calculated using the adjoint method rather than unrolling, and packages such as *torchdiffeq* (Chen, 2018) and *diffrax* (Kidger, 2022) have become publicly available.

Neural ODEs also provide a natural way of simultaneously learning and imputing, where the imputation is done with the time-evolution model and dynamically changes in every iteration as the model is trained. Neural ODEs have been successfully used on data with irregular

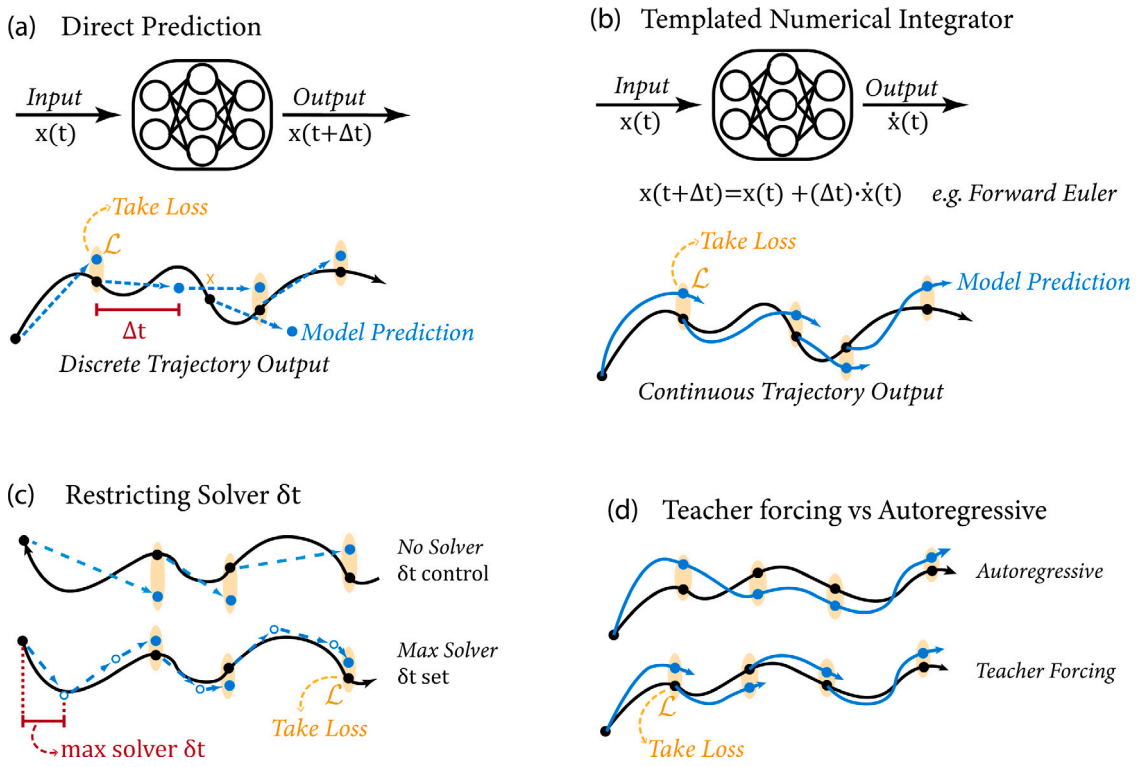


Fig. 1. Comparison of different RNN types. (a) Direct prediction methods typically force the network into a fixed Δt, which prompts the resampling, interpolation, or imputation of data that do not happen to be sampled at this Δt. (b) Incorporating the neural network in a numerical integrator template with a Δt input naturally allows for prediction at varying Δt values. (c) Restricting the max solver δt hyperparameter forces the network to take multiple steps if data Δt is large. (d) We explore both teacher-forcing, where real data is used as input when available, and autoregression, where predictions are fed back into network for the next prediction. During training on partial observations, elements of both teacher forcing and autoregression are used, while during inference, autoregression is used, starting from a known initial condition.

time series by Rubanova et al. (2019); in this work, we "open up" the Neural ODE approach by explicitly templating the neural network on a 4th order Runge–Kutta (RK4) integrator in PyTorch (Rico-Martinez et al., 1992) (Fig. 2). In our work the training gradients are computed by unrolling, rather than through the adjoint method.

Gray Box Models — Learning from data with some physics known. Various previous works in the literature have touched on different combinations of missing data characteristics. Rubanova et al. along with using a neural ODE to learn on irregularly-sampled time series data, also demonstrated learning from data with completely *hidden* variables (Rubanova et al., 2019)—but they do not consider partial observations. They operate on a latent representation that makes gray-boxing difficult, since latent variables need not have straightforward physical interpretations.

Kidger et al. (2020) extend neural ODEs to neural controlled differential equations (neural CDEs), which they view as "the continuous analog of RNNs" in the same way "neural ODEs are the continuous analog of ResNets" (Kidger et al., 2020). This method focuses on non-autonomous dynamical systems, motivated by rough path theory that views stochastic processes as "deterministic equations for a fixed choice of driving path" (Davie, 2008). They demonstrate the neural CDE methodology on a variety of classification tasks rather than system identification from time-series. While their application is very different from the one in this paper, we highlight their approach of using the neural CDE framework for training on irregular time series, and use of channel-specific observational frequency for training on partial observations.

Buisson-Fenet et al. (2022) give a simple demonstration of integrating neural ODEs with prior knowledge to produce a particular type of hybrid gray-box methods for system identification; here the physical knowledge is given in the form of known kinetic equations while the neural network learns a system parameter (the unknown frequency). In contrast, in our work, the neural network learns physically relevant *functions*, such as microbial growth rates or chemical reaction rates, alongside unknown values of fixed experimental parameters. In addition to working with a spectrum of gray-box models by incorporating prior information, a key contribution of Buisson-Fenet et al. (2022) is the use of a synchronized dynamical system based on the (Kazantzis-Kravaris-Luenberger) KKL observer theory to impute missing initial conditions. This addresses training on data with hidden variables, but not with partial observations, as in the current work. Approaches using kernel methods—rather than neural networks—have also been proposed (Bouvrie and Hamzi, 2017; Hamzi and Owhadi, 2021) for learning dynamical systems or for predicting missing initial conditions of LSTM latent dynamics (Kemeth et al., 2021).

The remainder of the paper is organized as follows: Having provided the definitions of the terms we use above, we start by summarizing the key concepts underpinning our approach and the algorithm. We demonstrate our approach first on a well-established illustrative model—the autonomous stirred tank reactor (Uppal et al., 1974); where we incrementally build on the complexity of the training data used to learn the model. We then proceed to a more complex system—a 6-dimensional biological kinetics model involving three microbial species growing in co-culture and exhibiting autonomous oscillations (Baltzis and Frederickson, 1984). In both systems, we demonstrate how gray-boxing may be used to both estimate unknown parameters and simultaneously learn unknown dynamics. We conclude with a summary of our observations, and a discussion of the scope of the approach and of its strengths and weaknesses.

2. Methodology

ODE-nets for learning with arbitrary time sampling. In order for the neural network to learn the right-hand-side (RHS) of a system of ODEs (rather than learning to directly predict the output in discrete

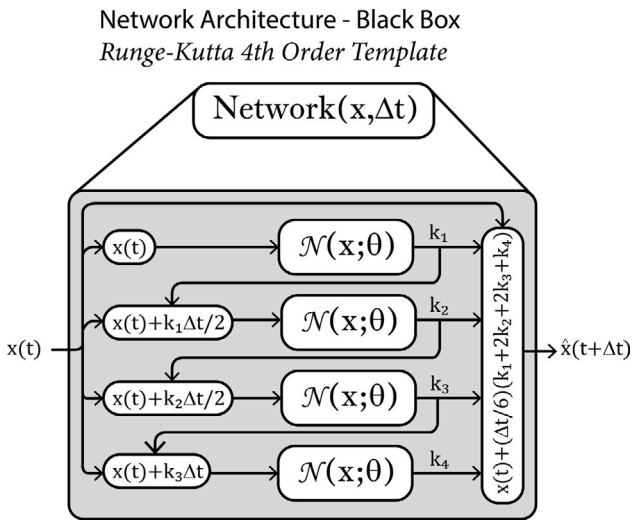


Fig. 2. A black box neural network model with a fourth order Runge–Kutta numerical integrator.

time), it is convenient if the network architecture has been templated on an established numerical integrator framework (Fig. 1(a), Fig. 1(b)) as a "fusion" of neural networks with traditional numerical analysis which we call ODE-nets (but may also be referred to in literature as Neural ODEs (Chen et al., 2018)). In this work we explicitly template the neural network on a 4th order Runge–Kutta (RK4) (Eq. (1), Fig. 2).

$$\hat{\mathbf{x}} = \mathcal{N}(\mathbf{x}(t), \mathbf{p}; \theta) \tag{1a}$$

$$\begin{aligned} \mathbf{k}_1 &= \mathcal{N}(\mathbf{x}(t), \mathbf{p}; \theta) \\ \mathbf{k}_2 &= \mathcal{N}(\mathbf{x}(t) + \mathbf{k}_1 \left(\frac{\Delta t}{2}\right), \mathbf{p}; \theta) \\ \mathbf{k}_3 &= \mathcal{N}(\mathbf{x}(t) + \mathbf{k}_2 \left(\frac{\Delta t}{2}\right), \mathbf{p}; \theta) \\ \mathbf{k}_4 &= \mathcal{N}(\mathbf{x}(t) + \mathbf{k}_3 \left(\Delta t\right), \mathbf{p}; \theta) \\ \hat{\mathbf{x}}(t+\Delta t) &= \mathbf{x}(t) + \left(\frac{\Delta t}{6}\right)(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \end{aligned} \tag{1b}$$

We know from numerical analysis that, generally speaking, the forward error in integrating a known equation improves as step size decreases, motivating the development of adaptive-stepsize integration schemes. Likewise, it can be shown that the backwards error, the inaccuracy in discovering an equation from assumed-true trajectory data, also improves as step size decreases (Zhu et al., 2022). As such, we control the time steps the RK4 integrator takes in order to maintain the precision of the solver, such that multiple iterations through the network may be needed to obtain a prediction at the next observation time step (Fig. 1(c)).

We also note two differing approaches to training: *teacher-forcing* and *autoregression* (Fig. 1(d)): teacher forcing was first introduced in the context of recurrent networks for temporal supervised learning (Williams and Zipser, 1989; Pineda, 1988) and involves replacing the output of the network with the true data (teacher signal), when available, for subsequent iterations. This approach allows for faster convergence of the model during training, particularly during earlier iterations (Benny Toomarian and Barhen, 1992). In contrast, autoregressive training involves feeding the outputs of the network back as the input for the next iteration to yield a full temporal trajectory (Chen et al., 2018). Previous literature has suggested that a balance between the two is optimal (Benny Toomarian and Barhen, 1992), and we take this into account in our training methodology (Fig. 4).

Gray-Boxing for data-driven learning with some physics known. We aim to develop and implement a neural-network based training methodology amenable to coupling with known, "white-box", physical knowledge, and capable of learning from training data sets with variable sampling times and partial observations. Both of these requirements can be addressed by formulating the training problem as the learning of the RHS of a system of ODEs. Often, established physical knowledge is expressed in terms of differential equations in space and time (chemical or microbial kinetics, convection, diffusion, etc.). Expressing the dynamics in the form of differential equations imposes the right inductive bias during the neural network model training to capture the continuous nature of these dynamical systems (Krishnapriyan et al., 2022); and it naturally allows the neural network training to incorporate existing partial physical knowledge. The methods can be extended to PDEs and even SDE/SPDEs (Dietrich et al., 2021; Psarellis et al., 2022), but that is beyond the scope of this paper.

In a simple black-box model, the neural network will have the full onus of learning the law governing the system dynamics:

$$\hat{\mathbf{x}} = \mathcal{N}_{\mathrm{BB}}(\mathbf{x}; \theta) \tag{2a}$$

Alternatively, previously available physical knowledge, which may be incomplete and/or imperfect, may be included in the model framework, so that the neural network is tasked with learning only the unknown parameters/dynamics, or possibly a correction to imperfectly known dynamics. This inclusion can be corrective (either additive (Eq. (2b)) or multiplicative (Eq. (2c))); or it could be a function composition (Eq. (2d)).

$$\hat{\mathbf{x}} = f_{\mathrm{WB}}(\mathbf{x}; \mathbf{p}) + \mathcal{N}_{\mathrm{BB}_{\mathrm{Add}}}(\mathbf{x}; \theta) \tag{2b}$$

$$\hat{\mathbf{x}} = f_{\mathrm{WB}}(\mathbf{x}; \mathbf{p}) \cdot \mathcal{N}_{\mathrm{BB}_{\mathrm{Mult}}}(\mathbf{x}; \theta) \tag{2c}$$

$$\hat{\mathbf{x}} = f_{\mathrm{WB}}(\mathbf{x}, \mathcal{N}_{\mathrm{BB}_{\mathrm{Func}}}(\mathbf{x}; \theta)) \tag{2d}$$

We can also combine corrective gray-boxes and function composition gray-boxes (Eq. (3), Fig. 3)

$$\phi = f_{\phi}(\mathbf{x}; \mathbf{p_f}, \theta) = \begin{cases} f_{\mathrm{WB}}(\mathbf{x}; \mathbf{p_f}) + \mathcal{N}_{\mathrm{BB}_{\mathrm{Add}}}(\mathbf{x}; \theta) & \text{Additive} \\ f_{\mathrm{WB}}(\mathbf{x}; \mathbf{p_f}) \cdot \mathcal{N}_{\mathrm{BB}_{\mathrm{Mult}}}(\mathbf{x}; \theta) & \text{Multiplicative} \end{cases} \tag{3a}$$

$$\hat{\mathbf{x}} = g_{\mathrm{WB}}(\mathbf{x}, \phi; \mathbf{p_g}) \tag{3b}$$

where f_{ϕ} are constitutive relations, or functions modeling kinetic or microbial rates, for which we may have *a priori* postulates f_{WB}. The neural network learns a correction for any imperfections or incompleteness in f_{WB}, or learns the entire f_{ϕ} if there are no *a priori* postulates. ϕ and x are inputs to another white box model g_{WB} which strictly imposes inviolable physical laws, such as conservation laws or other global laws we may want to impose, such as a coupling between reaction rate and the heat of reaction. Note that here, the white box parameters $\mathbf{p_f}$ and $\mathbf{p_g}$ can either be fixed *a priori*, or, if sufficiently identifiable given the data, they can be included as additional training parameters to be fit simultaneously with the learning of the neural network parameters θ.

In our approach, we focus on function composition, where the neural network handles the task of learning the unknown microbial or kinetic rates, and the architecture consists of a white-box structure imposing known physical couplings (e.g. between reaction rate and reactive heat generation, or microbial growth and substrate consumption/product formation (Eq. (4)).

$$\phi = \mathcal{N}_{\mathrm{BB}}(x; \theta) \tag{4a}$$

$$\hat{x} = g_{\mathrm{WB}}(x, \phi; p_g) \tag{4b}$$

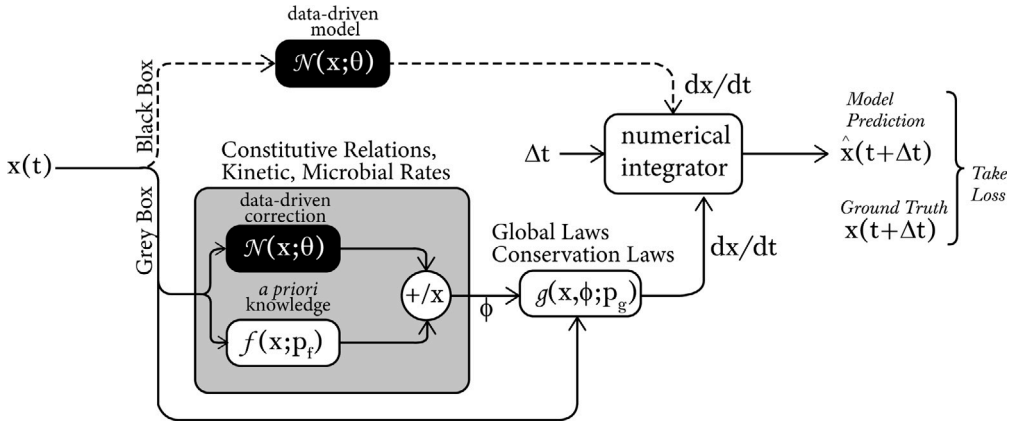


Fig. 3. Network architecture for black and gray box models. In the black box model, the entire onus for learning the RHS is on the neural network. In the gray-box model, physical information is incorporated through constitutive terms or known microbial and reaction rates, and global and conservation laws.

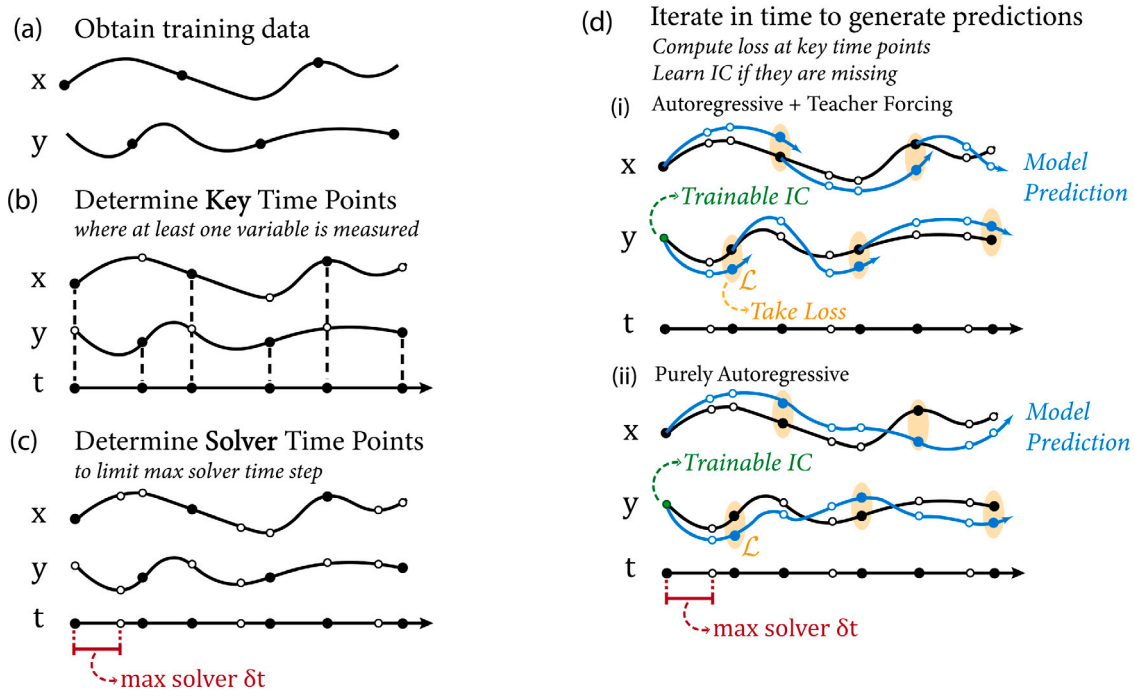


Fig. 4. Training algorithm and batching. (a) Systematically record each trajectory of training data. (b) Determine the **key** time points for each trajectory; key time points are where at least one feature is measured and hence a loss will be computed at these time points. (c) Determine **solver** time points. No loss is computed at these time points, these are included to ensure stability and accuracy of the numerical integrator. (d) Generate predictions by integrating through each trajectory in time; (i) teacher-forcing is used when real data is available for a feature, and if not predictions are generated autoregressively. In this paper this approach is used during *model training*; (ii) model predictions are generated purely autoregressively, starting from a given initial condition. This paper uses this approach during *inference* time. For training, compute loss at the key-time points for each trajectory, back-propagate to compute gradients and perform gradient descent to train the network.

Training algorithm and batching. The training algorithm is summarized in Fig. 4. Within each training trajectory, **key** time points (where the loss is calculated), and **solver** time points (included for stability and accuracy of the templated numerical integrator) are marked, and predictions are generated (with the current vectorfield estimate) by iterating in time through each trajectory. Each training trajectory may have different key and solver time points, and may also be of differing length. To include multiple trajectories in the same training minibatch, a sequence of evaluation time points is included alongside the data input, and padding is added to make all data and time sequences the same length. This allows for simultaneous training on minibatches of data trajectories.

Loss function and error metrics. For the loss function, a mean squared error (MSE) loss is taken between the model predicted trajectory and the ground truth data at the time points at which data is available:

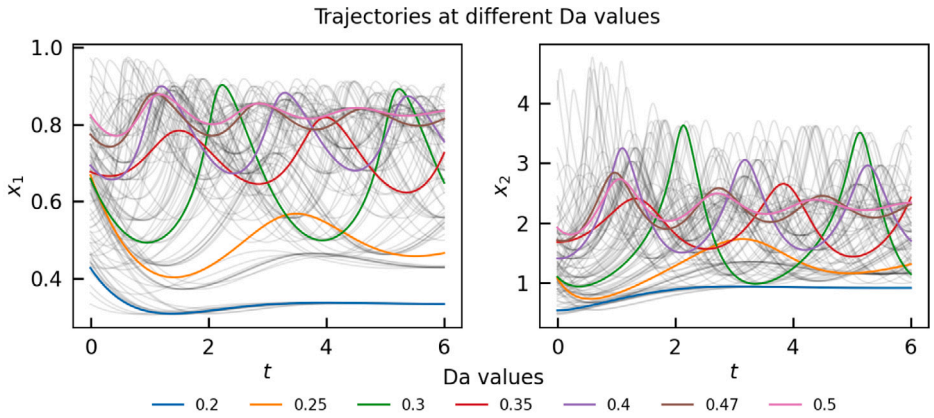
$$L = \frac{1}{N_{\text{data}}} \sum_{i \in \text{trajectories}} \sum_{j \in \text{times}_i} \sum_{k \in \text{channels}(i,j)} (\hat{y}_i^{(k)}(t_j) - y_i^{(k)}(t_j))^2. \tag{5a}$$

$$N_{\text{data}} = \sum_{i \in \text{trajectories}} \sum_{j \in \text{times}_i} \|\text{channels}(i,j)\| \tag{5b}$$

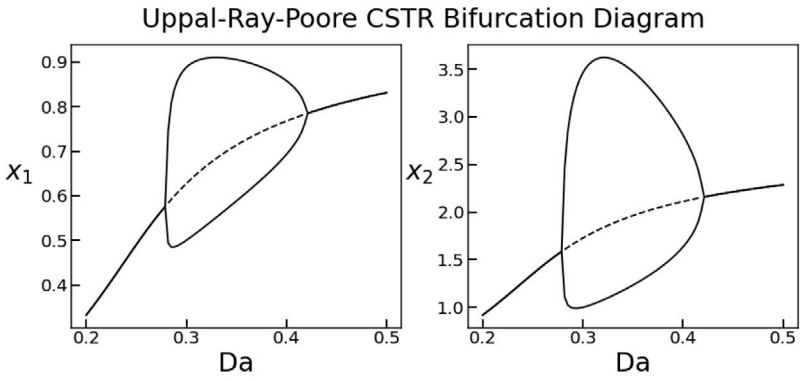
Here, $\text{channels}(i,j)$ is the set of channel indices observed in trajectory i at times j. Refer to Appendix A.2 for details on metrics used to evaluate model performance.

3. Results

Our first illustrative example uses data from simulations of the non-isothermal CSTR model in the classical chemical engineering literature (Uppal et al., 1974) (URP CSTR model), a two dimensional system



(a) Training data trajectories at various Da values



(b) Bifurcation Points at $Da = 0.280275$ and $Da = 0.419548$.
Solid lines: Stable steady state *or* max/min values of stable oscillations;
Dashed line: Unstable steady state

Fig. 5. Visualization of training data for URP CSTR.

of ordinary differential equations (Eq. (6)),

$$\frac{dx_1}{dt} = -x_1 + \mathrm{Da} \cdot (1 - x_1) \cdot \exp(x_2) = f_1(x_1, x_2) \tag{6a}$$

$$\frac{dx_2}{dt} = -x_2 + \mathrm{B} \cdot \mathrm{Da} \cdot (1 - x_1) \cdot \exp(x_2) - \beta \cdot x_2 = f_2(x_1, x_2) \tag{6b}$$

where x_1 is non-dimensional concentration (conversion), x_2 is non-dimensional temperature, and t is non-dimensional time. This system is parameterized by the (usually) fixed parameters $B = 11$ and $\beta = 3.0$, and a variable parameter Da (the Damkohler number) sampled in the range $\mathrm{Da} \in (0.2, 0.5)$. For this choice of B and β, the model exhibits two Hopf bifurcation points wrt. Da: at $\mathrm{Da} = 0.280275$ and $\mathrm{Da} = 0.419548$ respectively (Fig. 5). Stable periodic oscillatory behavior is observed in the range $0.280275 \leq \mathrm{Da} \leq 0.419548$, and a single stable steady state outside of this range. Sample trajectories of training data are visualized in Fig. 5.

The neural network architecture here is a multi-layer perceptron with the following structure: a three-neuron input layer with SiLU activation; two hidden layers of 64 neurons each with SiLU activation; and an output layer with two neurons (black-box formulation) or a single neuron (gray-box formulation) with linear activation.

In the "Black-Box" formulation, no *a priori* knowledge of the system is assumed, and so the neural network must learn the *full* dynamics (Eq. (7)) and its dependence on Da.

$$\hat{\mathbf{x}} = \mathcal{N}(\mathbf{x}, \mathrm{Da}; \theta) \quad \text{where } \mathbf{x} = \{x_1, x_2\} \tag{7}$$

In the gray-box formulation, *some* parts of the system dynamics are known *a priori*, while others are unknown (Eq. (8)). Here, the kinetics of the reaction are assumed to be unknown, but the form of the overall mass and energy conservation laws, as well as the coupling of the reaction rate to reactive heat generation (the heat of reaction, which can be measured independently) are assumed to be known.

$$\begin{aligned} \hat{\phi} &= \mathcal{N}(\mathbf{x}, Da; \theta) \\ \hat{x_1} &= -x_1 + \hat{\phi} \\ \hat{x_2} &= -x_2 + \mathrm{B} \cdot \hat{\phi} - \beta \cdot x_2 \end{aligned} \tag{8a}$$

Here $\hat{\phi}(\mathbf{x}, Da)$ is the neural network approximation of the reaction kinetics (Eq. (8b)).

$$\phi = f_\phi(\mathbf{x}, \mathrm{Da}) = \mathrm{Da} \cdot (1 - x_1) \cdot \exp(x_2) \tag{8b}$$

Here B and β are experimental parameters that can be independently measured, are assumed known *a priori* and are "hardwired" in the network; if they happen to *not* be known, they can of course be considered as *additional* training parameters, to be fitted along with Da and the neural network weights during ANN training.

For simplicity, unless otherwise specified, $\mathcal{N}$ represents the overall network trained to predict $\hat{\mathbf{x}}$, with all gray-box aspects subsumed within this representation.

$$\hat{\mathbf{x}} = \mathcal{N}(\mathbf{x}, \mathrm{Da}; \theta) \tag{8c}$$

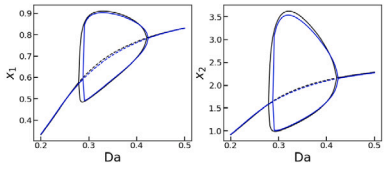
In our demonstration we gradually "build up" the complexity in the training data used by incrementally incorporating the following pathologies: large data sampling Δt; arbitrary data Δt; partial observations; and without initial conditions known/given. All model results are summarized in Tables 1 and 2. Case A (Fig. 6(a)) is the base case, with the training data trajectories densely sampled in time, with full observations and regular time sampling. For this simple base case, we

Table 1
Metrics summary table for URP CSTR training. This Table showcases model performance on data of increasing complexity of representation. Errors are means and standard deviations of 10 training runs. Refer to Appendix A.2 for details on error metrics.

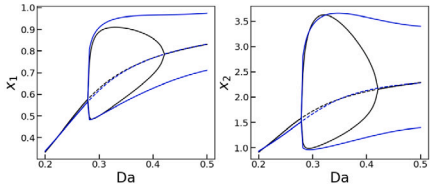
Case	A	B	C	D	E	F	URP_BB
Data $< \Delta t >$	0.1	0.5	0.5	0.5	0.5 (x_1) 0.55(x_2)	0.5	0.5
Solver max δt	0.1	0.5	0.1	0.1	0.1	0.1	0.1
Arbitrary data Δt	–	–	–	✓	–	✓	✓
Partial observations	–	–	–	–	✓	✓	✓
Initial condition known	✓	✓	✓	✓	✓	✓	–
Solution error ($\mathcal{L}_2$)	$(9.68 \pm 1.71) \times 10^{-3}$	$(1.49 \pm 0.12) \times 10^{-2}$	$(8.69 \pm 1.18) \times 10^{-3}$	$(4.23 \pm 0.43) \times 10^{-3}$	$(6.09 \pm 0.56) \times 10^{-3}$	$(8.91 \pm 1.65) \times 10^{-3}$	$(7.92 \pm 1.36) \times 10^{-3}$
RHS error ($\mathcal{L}_2$)	$(2.72 \pm 0.36) \times 10^{-3}$	$(9.71 \pm 0.76) \times 10^{-3}$	$(5.95 \pm 1.21) \times 10^{-3}$	$(3.50 \pm 0.34) \times 10^{-3}$	$(3.66 \pm 0.46) \times 10^{-3}$	$(7.12 \pm 0.81) \times 10^{-3}$	$(7.69 \pm 0.11) \times 10^{-3}$

Table 2
Metrics summary table for URP CSTR data. This table showcases model performance of black-box and gray-box models. All models have arbitrary data Δt, partial observations and unknown initial conditions. Errors are means and standard deviations of 10 training runs. Refer to Appendix A.2 for details on error metrics.

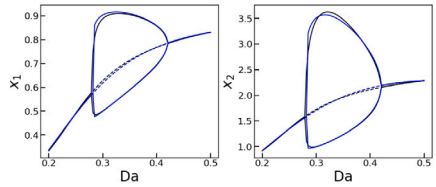
Case	Learnable kinetic functions	Learnable experimental parameters	Solution error ($\mathcal{L}_2$)	RHS error ($\mathcal{L}_2$)	Kinetic function error ($\mathcal{L}_2$)	Experimental parameter error ($\mathcal{L}_2$)
URP_BB	–	–	$(7.92 \pm 1.36) \times 10^{-3}$	$(7.69 \pm 0.11) \times 10^{-3}$	–	–
URP_GB1	✓	–	$(9.91 \pm 1.26) \times 10^{-3}$	$(4.43 \pm 0.51) \times 10^{-3}$	$(5.31 \pm 0.62) \times 10^{-3}$	–
URP_GB2	✓	✓	$(1.22 \pm 0.17) \times 10^{-2}$	$(6.49 \pm 2.08) \times 10^{-3}$	$(7.77 \pm 2.25) \times 10^{-3}$	$(4.45 \pm 2.28) \times 10^{-2}$



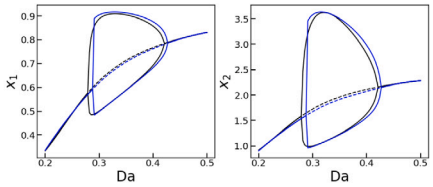
(a) Case A: Base Case: Ideal data (Finely sampled, full observations, fixed time sampling)



(b) Case B: Training with no solver time step control



(c) Case C: Training with solver time step control



(d) Case URP_BB: Final Model (Solver time step control, partial observations, variable time step sampling, trainable initial conditions

Fig. 6. Bifurcation diagram predictions.

observe good reproduction of the bifurcation diagram by the black box trained network. Cases B (Fig. 6(b)) and C (Fig. 6(c)) demonstrate the importance of controlling the solver max δt. In Case B, the solver max $\delta t = 0.5$ is too large for stable/accurate integration with the embedded RK4 integrator, and the model is unable to reproduce the Hopf Bifurcation point near $Da = 0.42$. Reducing the solver max δt to 0.1 in Case C resolves this issue. In Cases D and E, we introduce arbitrary time sampling and partial observations respectively; Case F has both of these data pathologies simultaneously. In all three we observe good training results with low solution and RHS error. Finally, in Case URP_BB (Fig. 6(d)), we let the initial condition vector be trainable, allowing for training on data where the initial condition is unknown/where not all variables are measured at the first sampling time.

In Cases URP_GB1 and URP_GB2 (Fig. 7, Table 2) we introduce physics informed "gray-boxes" into our model. In both, rather than learning the dynamic evolution of variables x_1 and x_2 directly, the model instead learns the kinetic function f_ϕ; the evolution of x_1 and x_2 is reconstructed by mass and energy balances, and coupling of reaction kinetics and energetics. In Case URP_GB1, the coupling constants B and β are known, while in Case URP_GB2 they are unknown and introduced as additional trainable parameters for the model. In both cases, the data has the pathology of having partial observations, arbitrary time sampling, and unknown initial conditions. In Case URP_GB2 (Fig. 7) we observe good reproduction of the bifurcation diagram, accurate prediction of the kinetic function f_ϕ that translates into accurate RHS predictions for both x_1 and x_2, and the coupling constants B and β are well predicted.

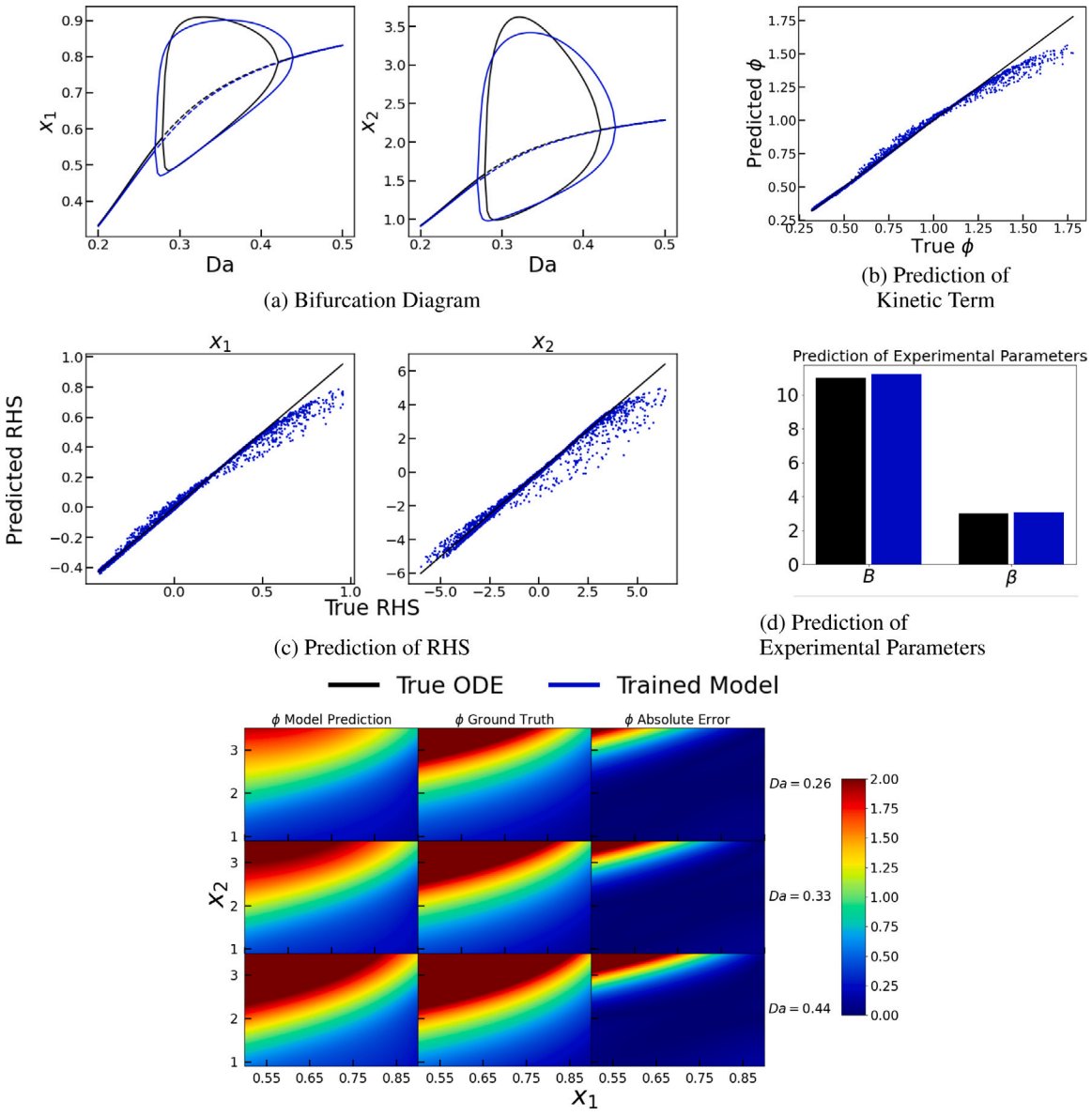


Fig. 7. Trained model performance for case URP_GB2 (Gray-Box with trainable experimental parameters).

A major benefit of training in "gray-box" as opposed to "black-box" is that it retains physical intuition, and in doing so we expect that it will allow for extrapolation to parameter space beyond the training data. This is evidenced in the Case URP_GB1 (Fig. 8); this model was trained on a fixed, known value of B and β, and the neural network was used to learn the kinetic reaction term. This allows the model to be extrapolated to values of B and β well beyond those trained on, allowing for predictions of bifurcation diagrams for B and β that are remarkably close to ground truth despite the model having been trained at *a single* set of parameter values.

For our second illustrative example, we use data from a classical biochemical model involving the coexistence of three microbial species growing in co-culture, coupled through various chemical substrates and cofactors (Baltzis and Frederickson, 1984) (B&F Model). This is a six-dimensional system of coupled ODEs outlined in Eq. (9), with parameter values in Table 3. The training data is visualized in Fig. 9.

$$\begin{aligned}
\frac{dx}{dt} &= -\alpha x + \mu_1 x - \mu_{c_1} x && \text{Host Species}\\
\frac{dy}{dt} &= -\alpha y + \mu_2 y - \mu_{c_2} y && \text{Commensal Species 1}\\
\frac{dz}{dt} &= -\alpha z + \mu_3 z - \mu_{c_3} z && \text{Commensal Species 2}\\
\frac{du}{dt} &= \alpha(u_f - u) - \mu_1 x && \text{Host Substrate}\\
\frac{dv}{dt} &= -\alpha v + \omega\mu_1 x - \mu_2 y - \sigma\mu_3 z && \text{Growth Factor}\\
\frac{dg}{dt} &= -\alpha g + \rho\mu_2 y + \eta\mu_3 z && \text{Inhibition Factor,}
\end{aligned} \tag{9a}$$

where

$$\begin{aligned}
\mu_1 &= f_{\mu_1}(u,g) = \frac{u}{(1+u)(1+g)} && \text{Host Species Growth Rate}\\
\mu_2 &= f_{\mu_2}(v) = \frac{\phi_1 v}{1+v} && \text{Commensal Species 1 Growth Rate}\\
\mu_3 &= f_{\mu_3}(v) = \frac{\phi_2 v}{\sigma+v} && \text{Commensal Species 2 Growth Rate.}
\end{aligned} \tag{9b}$$

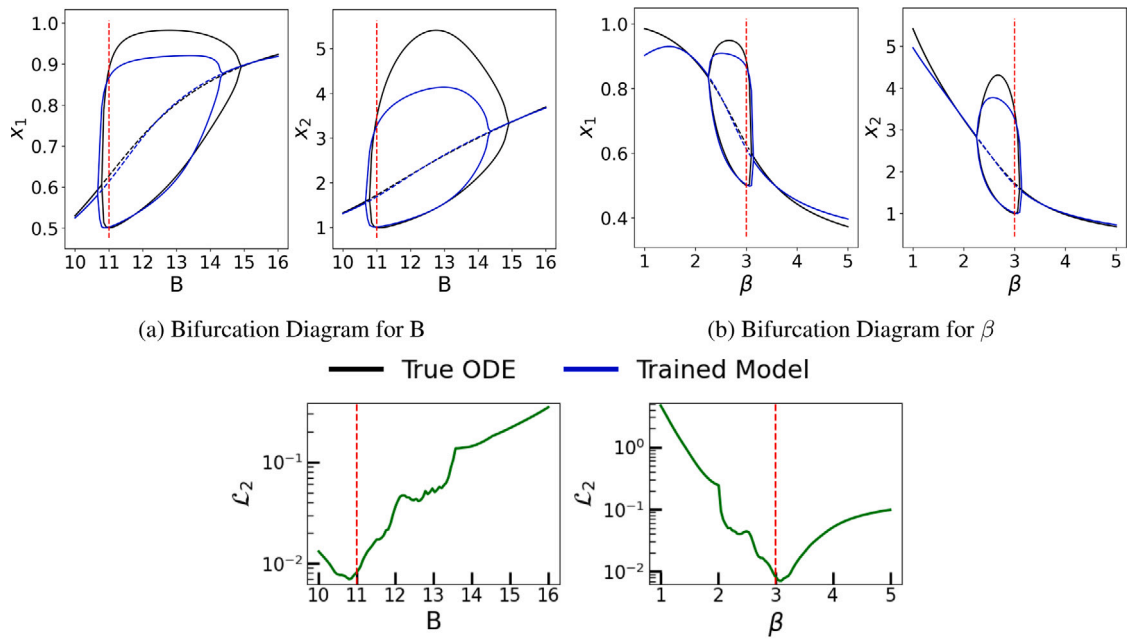


Fig. 8. Case URP_GB1 (Gray-Box with fixed/known experimental parameters): Extrapolating to new parameter values. The dotted red line is at the parameter value of training data; all other parameter values are extrapolations beyond training data.

Table 3
Parameter values for Baltzis & Frederickson (B&F) model.

Parameter	Value
α	1/7.3
u_f	250.0
ω	9.7
σ	10.0
ρ	0.13138686
η	1.29166
ϕ_1	0.2941176
ϕ_2	0.367647
μ_{c_1}	0.367647
μ_{c_2}	0.117647
μ_{c_3}	0.1617647

The host x grows on substrate u and secretes growth factor v. Commensal species y and z consume growth factor v and release inhibition factor g that inhibits growth of the host x. f_{μ_i} are the growth rate functions of each species, and μ_{ci} is a constant parameter dictating maintenance rate of each biomass species.

The neural network architecture is a multi-layer perceptron with a six neuron input layer with SiLU activation; two hidden layers of 32 neurons each with SiLU activation; and an output layer with six neurons (black-box formulation) or three neurons (gray-box formulation) with linear activation. In the "Black-Box" formulation, no *a priori* knowledge of the system is assumed, and so the neural network must learn the *full* system (Eq. (10))

$$\hat{\mathbf{x}} = \mathcal{N}(\mathbf{x};\theta) \quad \text{where } \mathbf{x} = \{x, y, z, u, v, g\}. \tag{10}$$

In the gray-box formulation, some parts of the system dynamics are known *a priori*; others are unknown (Eq. (11)). Here, the microbial growth rates are assumed to be unknown, but the overall mass and energy conservation laws, as well as the coupling of growth rates to substrate and growth/inhibition factor production and utilization are known:

$$\begin{aligned}
\{\hat{\mu}_{1,x}, \hat{\mu}_{2,y}, \hat{\mu}_{3,z}\} &= \mathcal{N}(\mathbf{x};\theta) \\
\dot{\hat{x}} &= -\alpha x + \hat{\mu}_{1,x} - \mu_{c_1} x \\
\dot{\hat{y}} &= -\alpha y + \hat{\mu}_{2,y} - \mu_{c_2} y \\
\dot{\hat{z}} &= -\alpha z + \hat{\mu}_{3,z} - \mu_{c_3} z \\
\dot{\hat{u}} &= \alpha(u_f - u) - \hat{\mu}_{1,x} \\
\dot{\hat{v}} &= -\alpha v + \omega\hat{\mu}_{1,x} - \hat{\mu}_{2,y} - \sigma\hat{\mu}_{3,z} \\
\dot{\hat{g}} &= -\alpha g + \rho\hat{\mu}_{2,y} + \eta\hat{\mu}_{3,z}
\end{aligned} \tag{11a}$$

where $\{\hat{\mu}_{1,x}, \hat{\mu}_{2,y}, \hat{\mu}_{3,z}\}$ are neural network approximations of the microbial growth rates (Eq. (9b)) multiplied by the biomass concentration of the respective microbe; and $\{\omega, \sigma, \rho, \eta\}$ are experimental parameters that are either known *a priori*, or else included as *additional* parameters to be fit during ANN training. For simplicity of expression, unless otherwise specified, let $\mathcal{N}$ represent the overall network to predict $\hat{\mathbf{x}}$, with all gray-box aspects subsumed within this representation.

$$\dot{\hat{\mathbf{x}}} = \mathcal{N}(\mathbf{x};\theta) \tag{11b}$$

We now demonstrate the training methodology, developed and illustrated through the URP CSTR model, on the B&F model; the metrics are summarized in Table 4. All cases have the data pathology of arbitrary time sampling, partial observations and unknown initial conditions. Case BF_BB (Fig. 10) is the black-box case, and we observe good prediction of the RHS of all 6 variables as well as accurate reproduction of the steady limit cycle. In cases BF_GB1 and BF_GB2 (Fig. 11) we introduce physics-informed gray-boxes, with the model learning the microbial growth functions for the species x, y, z, with the overall evolution of all 6 variables specified using mass balances with coupling parameters $\omega, \sigma, \rho, \eta$. In Case BF_GB1 these coupling parameters are known, while in Case BF_GB2 they are unknown and are introduced as additional parameters to be learned during model training. For Case BF_GB2 (Fig. 11), the model predicts accurately the microbial growth rates and the coupling parameters; this translates to accurate prediction of the RHS of all 6 variables and good reproduction of the limit cycle.

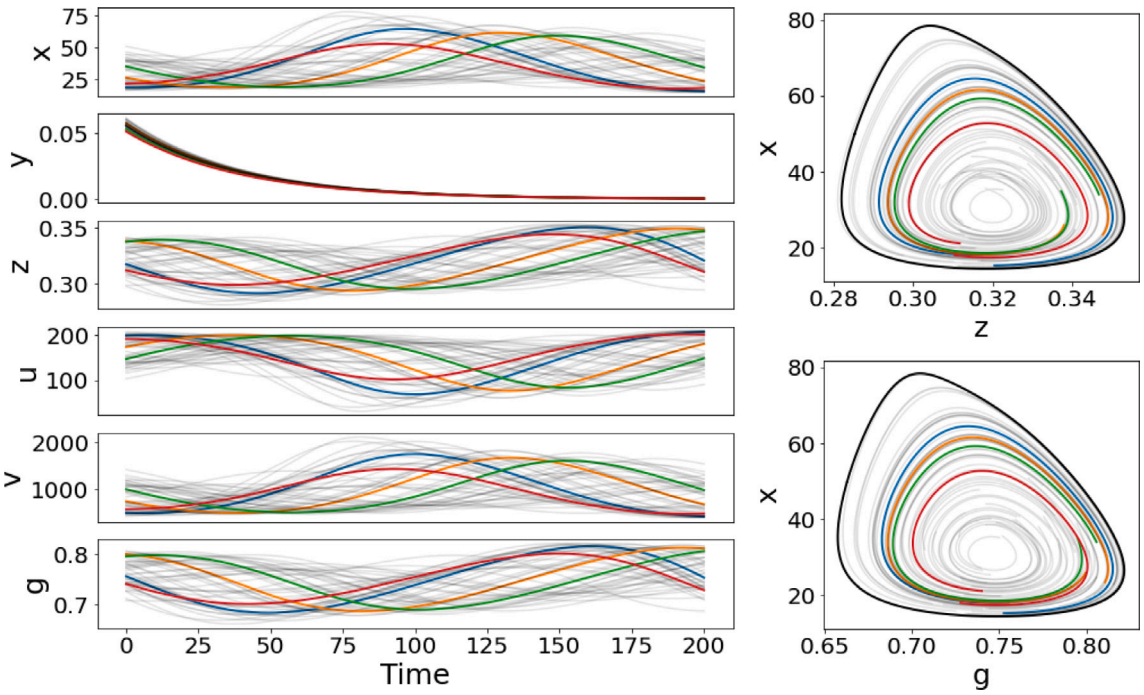


Fig. 9. Transients and limit cycles of B&F training data. The stable Limit Cycle is given in black; the 200 training trajectories represented in light gray with 4 emphasized in colors to show transient behavior.

Table 4
Metrics summary table for B&F data. This table showcases model performance of black-box and gray-box models. All models have arbitrary data Δt, partial observations and unknown initial conditions. Refer Appendix A.2 for details on metrics.

Case	Learnable kinetic functions	Learnable exp. parameters	Solution error from true LC ($\mathcal{L}_2$)	RHS error ($\mathcal{L}_2$)	Kinetic function error ($\mathcal{L}_2$)	Exp. parameter error ($\mathcal{L}_2$)
BF_BB	✗	✗	$(3.25 \pm 0.43) \times 10^{-3}$	$(2.12 \pm 0.45) \times 10^{-3}$	–	–
BF_GB1	✓	✗	$(5.16 \pm 0.68) \times 10^{-3}$	$(5.43 \pm 0.80) \times 10^{-3}$	$(7.66 \pm 1.14) \times 10^{-4}$	–
BF_GB2	✓	✓	$(6.10 \pm 2.72) \times 10^{-3}$	$(7.50 \pm 3.2) \times 10^{-3}$	$(1.09 \pm 0.49) \times 10^{-3}$	$(7.51 \pm 2.1) \times 10^{-2}$

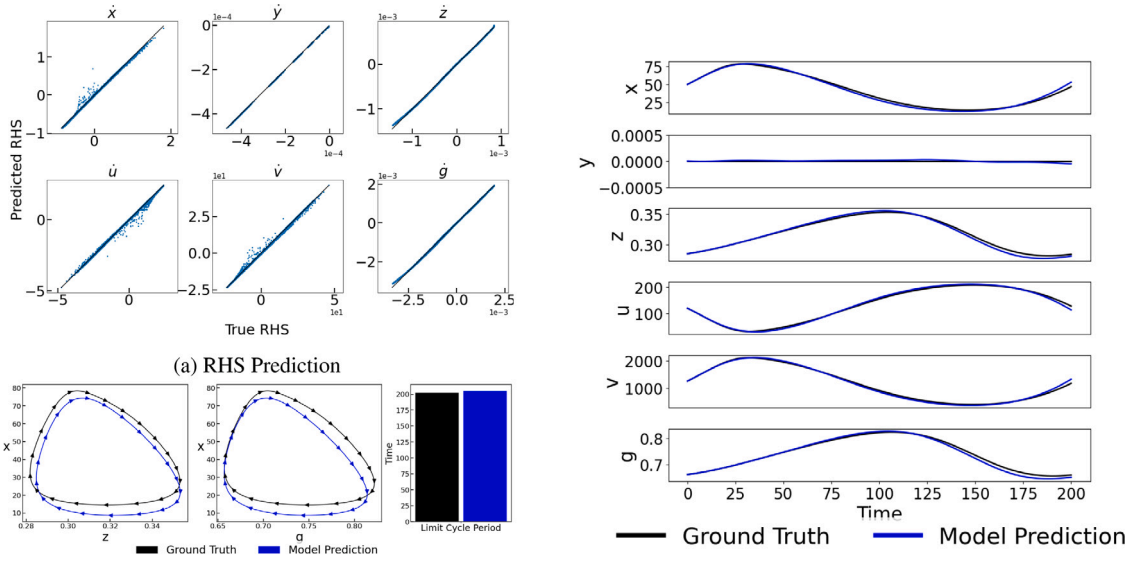


Fig. 10. Trained model performance for case BF_BB.

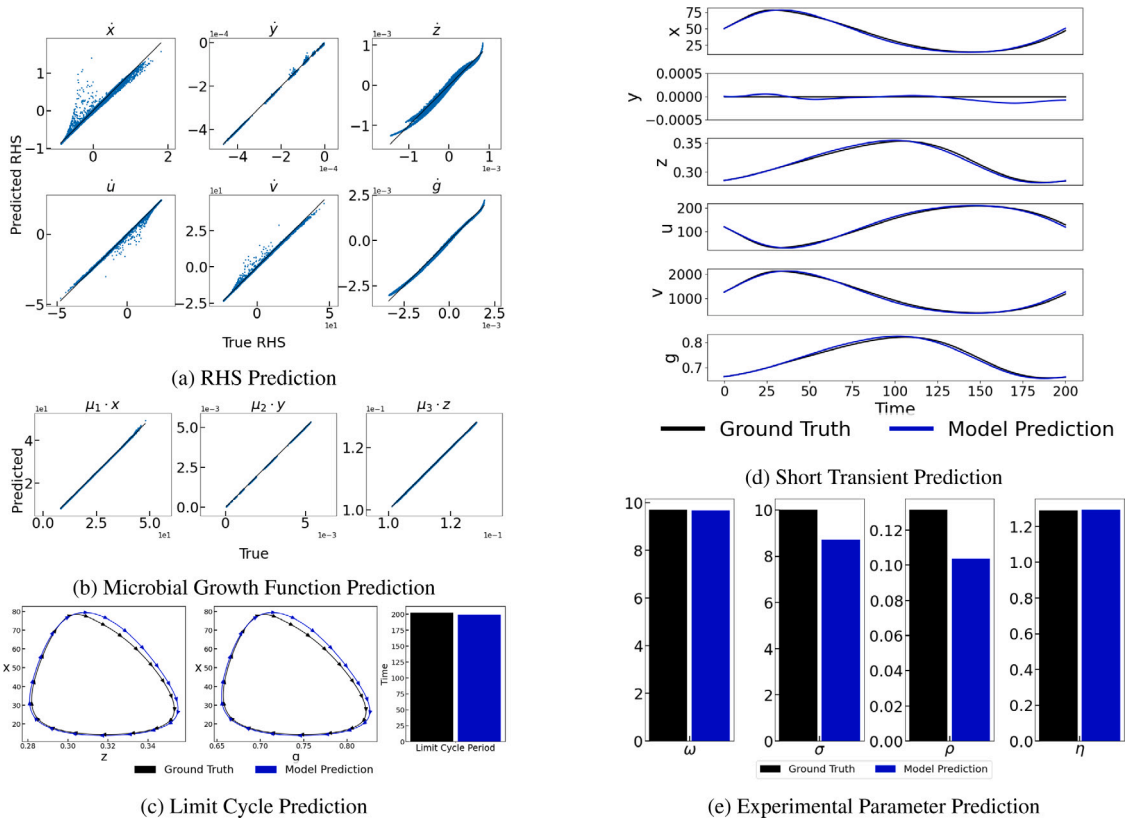


Fig. 11. Trained model performance for Case BF_GB2.

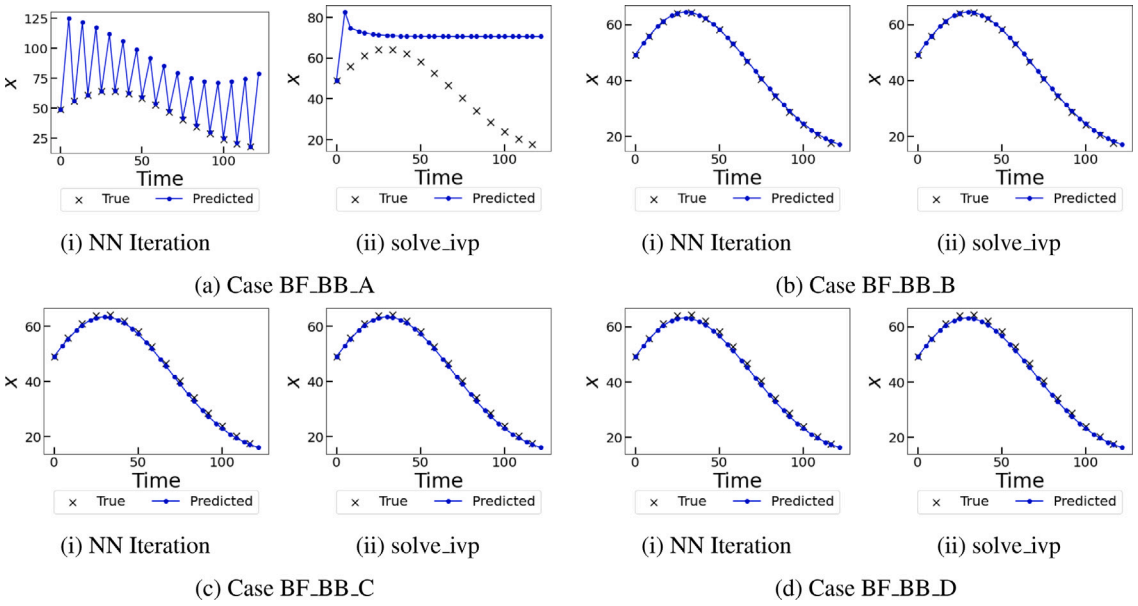


Fig. 12. The performance of the best model on a single representative training trajectory, for cases BF_BB_A through BF_BB_D. This is shown for both (i) the recursive neural network iteration with the time-steps used during training; and (ii) the scipy.integrate.solve_ivp solver, which uses variable time stepping (and thus helps better test the accuracy of integrating the learned RHS).

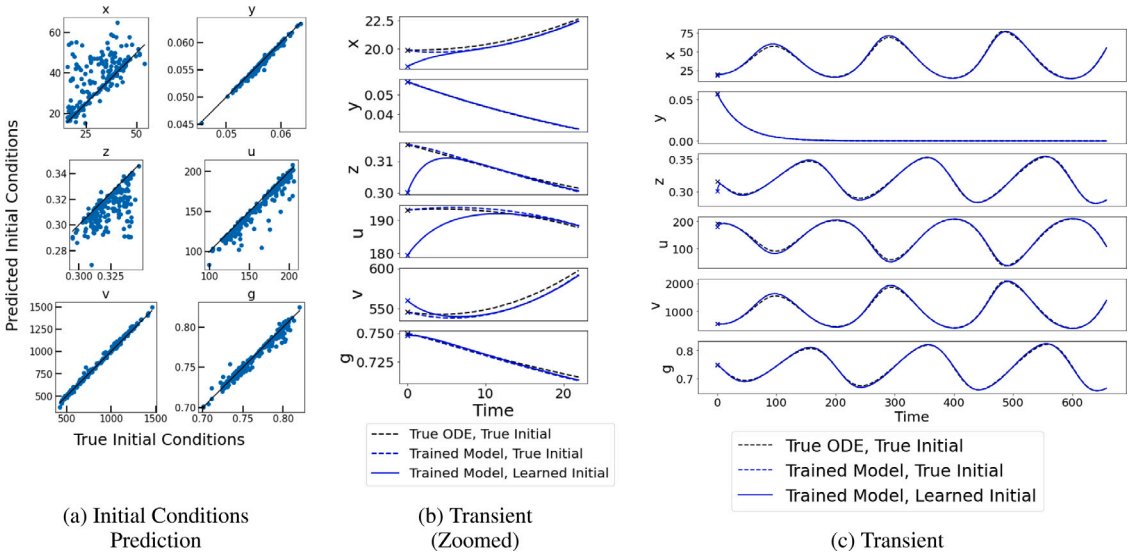
4. Discussion and conclusions

In the course of this work, we observed certain pathologies with the model training that we believe are interesting enough to report and discuss.

The "Resonance" Effect. We observed that, when working with fixed rate time sampling data, depending on the initialization of model weights, training might indeed successfully fit the data by converging to a visibly wrong local minimum. Rather than learning the RHS of the underlying ODE system, it learned a "resonance-type" time-one map: instead of a smooth curve interpolating the data points, two integration steps, with very different slopes (Fig. 12(a)(i), 12(a)(ii)), now take us from each data point to the next; one can describe this as a 2:1 resonance between the integration step and the data sampling step.

Table 5
Metrics summary table for B&F data. This table showcases the resonance effect. Refer Appendix A.2 for details on metrics.

Case	Initial scaling	Randomized data Δt	Randomized solver δt	Solution error from true LC ($\mathcal{L}_2$)	RHS error ($\mathcal{L}_2$)
BF_BB_A	$\times 100^0$	✗	✗	$(4.33 \pm 2.58) \times 10^{-1}$	8.47 ± 1.63
BF_BB_B	$\times 100^{-1}$	✗	✗	$(4.13 \pm 0.53) \times 10^{-3}$	$(2.89 \pm 0.92) \times 10^{-3}$
BF_BB_C	$\times 100^0$	✗	✓	$(8.40 \pm 3.52) \times 10^{-2}$	1.65 ± 0.68
BF_BB_D	$\times 100^0$	✓	✓(Because of data)	$(1.14 \pm 0.17) \times 10^{-2}$	$(2.88 \pm 0.68) \times 10^{-2}$
BF_BB_E	$\times 100^{-1}$	✓	✓(Because of data)	$(4.18 \pm 0.71) \times 10^{-3}$	$(3.10 \pm 0.56) \times 10^{-3}$



(a) Initial Conditions Prediction (b) Transient (Zoomed) (c) Transient

Fig. 13. Initial condition prediction for case BF_GB2. Despite the initial conditions learned not being correct, the *trained* ANN when integrated from the learned initial condition (solid blue line) quickly converges with the trajectory integrated from the true initial condition (dashed blue line), and in long-term dynamics the two are practically indistinguishable. This is due to the separation of time scales (fast-slow nature) of the initial startup.

In the above example, the data was sampled every $\Delta t = 8.333$. For case Bf_BB_A (Fig. 12(a)), this Δt was broken up in the RK4 numerical integrator into two steps of 5 and 3.33 consistently. With "bad" (random) weight initialization, the model training "hard-wires" this time step, essentially learning a time-one map without explicitly learning the RHS of the ODE; when iterated through the network with the same time-steps as during training, we observe a characteristic zig-zag "resonance" style output (Fig. 12(a)(i)). We explored various mitigation techniques to prevent convergence to such a local minimum (Table 5). One approach involved attempts at better initialization of the model weights (here, simply scaling the output layer's weights down by a factor of 100, with the initialization of other layers unchanged) (Case BF_BB_B, Fig. 12(b)). Secondly, we randomize the solver δt steps so that, rather than consistently taking steps of (5+3.333), the $\Delta t = 8.333$ was broken up into randomly sized time chunks, all selected below the max δt of 5 (Case BF_BB_C, Fig. 12(c)). Thirdly, we randomize the data sampling itself, with a gamma distribution with mean $< \Delta t = 8.333 >$, and in doing so the solver δt values were naturally randomized too (Case BF_BB_D, Fig. 12(d)). All three methods allowed the model to escape the bad local minimum, or to even remove it entirely (see Table 5).

The learning of initial conditions. We have observed that when initial conditions were not part of the training set, and needed to be inferred our models did **not** consistently infer the correct initial conditions (Fig. 13). This can be rationalized (and mitigated) by considering what happens in multi-time-scale singularly perturbed dynamic problems (fast-slow systems, systems with large separation of time scales) (Fig. 14, Eq. (12)).

$$\begin{aligned} \dot{x} &= f(x, y) \\ \epsilon \dot{y} &= g(x, y) \end{aligned} \tag{12}$$

In such problems (our 6 ODEs exhibit eigenvalues of the linearization that vary by one or more orders of magnitude (Table 6)), initial conditions very quickly (over the fast time scale) converge to a "slow manifold", on which the long-term dynamics evolve over the slow time scale. This is the same argument as the celebrated Quasi-Steady-State (QSSA) or Bodenstein Approximation in chemical kinetics. This means that *an infinity* of initial conditions will quickly (over the fast time scale) end up at (practically) the same state value on the slow manifold after this short initial integration period, and then remain and evolve on the slow manifold (Gear et al., 2005; Vandekerckhove et al., 2009; Antonios et al., 2012; Zagaris et al., 2009).

In the relevant literature (Balmaseda et al., 2009), this is mitigated by searching for an initial condition that *already* lies approximately on the slow manifold. Say we are interested in an initial condition Δt before our first sampled data point. The trick for constructing such a "mature" or "bred" initial condition one Δt in the past involves estimating an initial condition farther back (say 10 Δt earlier, which can be anywhere on the fast foliation of the slow manifold), and then letting it evolve for $9\Delta t$. This allows the "bad" components of the earlier initial condition to "mature" ("breed") giving us a good estimate for an initial condition *on the slow manifold* at the desired one Δt in the past.

Summary and Outlook. In this paper, we presented a neural-network based identification architecture—inspired by, and based on traditional numerical analysis—to address a number of time series

Table 6
Eigenvalues for the jacobian of the ground truth ODEs and the trained model on the true and learned initial conditions. The fast dynamics associated with the leading eigenvalues (in red) decay quickly at a time scale of $t = 2$ to $t = 2.3$ and quickly converge to the slower manifolds associated with the eigenvalues in blue that are consistent with the true system. The training data used was sampled at an average $\Delta t = 8$, hence the faster dynamics could not be learned (or discouraged) during model training.

Eigenvalue	True ODE True IC	Trained ANN True IC	Trained ANN Learned IC
1	$-0.20506 - 0.020221j$	-0.51397	-0.51752
2	$-0.20506 + 0.020221j$	-0.44970	-0.44258
3	-0.20145	-0.20145	-0.20145
4	-0.025494	-0.026535	-0.026341
5	$0.002033 - 0.03847j$	$-0.000690 - 0.03661j$	$-0.004094 - 0.03534j$
6	$0.002033 + 0.03847j$	$-0.000690 + 0.03661j$	$-0.004094 + 0.03534j$

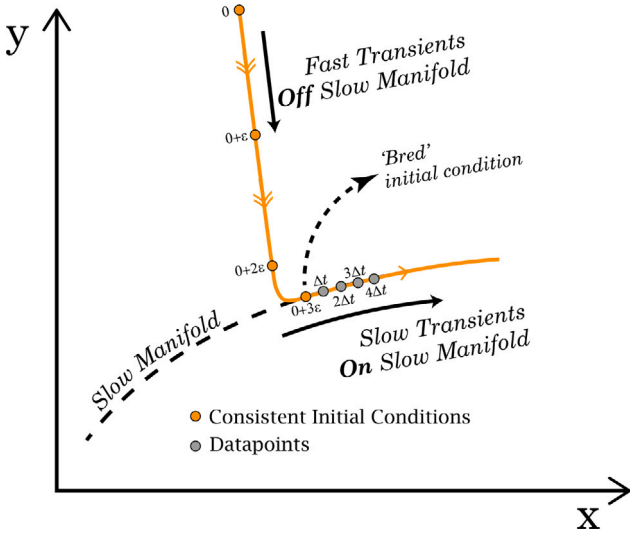


Fig. 14. Separation of fast and slow time-scales, as observed in singular perturbation problems (Eq. (12)), can lead to multiple initial conditions that are *practically* consistent with the same first measured datapoint, as they evolve rapidly on the fast manifold before converging on to the slow manifold. Ideally, we would want the initial condition that has been 'bred' till it lies on the slow manifold as this is the "most likely" candidate, but this can be hard to impose.

data pathologies: uneven sampling rates, missing data (including initial conditions), as well as unknown or partially known physics.

We formulated the systems identification process as the learning of the right-hand-side of a system of ordinary differential equations, which can be combined with numerical integration methods. In each forward pass, we iterate through each training trajectory through time, using teacher-forcing when real data is available, and autoregressive iterations when not, allowing for training on partial observations. We demonstrated how this architecture can be integrated with white-box prior knowledge, where the neural networks learn physically interpretable functions such as microbial growth rates or chemical reaction rates, as well as experimental parameters; global laws, such as coupling between reaction rates and heats of reaction, or microbial growth rates, substrate consumption and product formation were easy to systematically "hardwire" in the architecture.

We expect that the approach (especially its "gray box" component) will make it useful in industrial applications (e.g. in CHO cell culture biomanufacturing, which we are currently exploring). On the technical/computational side, a natural next task involves the efficient and easily usable implementation of parallel batching, and, more generally the consistent use of parallelization to accelerate computation. We also expect work towards incorporating adaptivity of the approach, in the spirit of adaptive time step selection for traditional initial value solvers. Linking this work with uncertainty quantification is an important direction, and a subject of current applied ML research, whether of parameter uncertainty or of prediction uncertainty. Remarkably, the error between the identified right-hand-sides of the dynamics (termed *Inverse Modified Differential Equations* or IMDEs) and the true dynamics—the so called *Inverse Modified Error Analysis*—appears to be a nascent branch of numerical analysis, born by precisely the type of work we present here (Zhu et al., 2022, 2023).

CRediT authorship contribution statement

Saurabh Malani: Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Tom S. Bertalan:** Software, Writing – original draft. **Tianqi Cui:** Software, Writing – original draft. **José L. Avalos:** Funding acquisition, Writing – original draft. **Michael Betenbaugh:** Conceptualization, Writing – original draft. **Ioannis G. Kevrekidis:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The work of IGK and TSB was partially supported by the US Department of Energy (grant number SA22-0052-S001); and JLA and SM are supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research (Award Numbers DE-SC0019363 and DE-SC0022155). IGK and TSB were also partially supported by the US Air Force Office of Scientific Research (AFOSR) (grant number FA9550-21-0317); and by AMBIC, that partially supported the work of MB. SM was also supported by the National Science Scholarship (NSS) from the Agency for Science, Technology and Research (A*STAR) Singapore.

Appendix A. Supplementary

A.1. Supplementary results

A.1.1. URP model
See Tables A.1 and A.2.

A.1.2. B&f model
See Tables A.3 and A.4.

Table A.1
Metrics Summary Table for URP CSTR data. This table showcases model performance on data of increasing complexity of representation. Refer Section A.2 for details on metrics.

Case	Data $\langle \Delta t \rangle$	Solver max δt	Arbitrary data Δt	Partial observations	Initial condition known	Solution error	RHS error
A	0.1	0.1	✗	✗	✓	$\mathcal{L}_1 : (1.31 \pm 0.24) \times 10^{-2}$ $\mathcal{L}_2 : (9.68 \pm 1.71) \times 10^{-3}$ $\mathcal{L}_\infty : (1.58 \pm 0.27) \times 10^{-2}$	$\mathcal{L}_1 : (3.66 \pm 0.52) \times 10^{-3}$ $\mathcal{L}_2 : (2.72 \pm 0.36) \times 10^{-3}$ $\mathcal{L}_\infty : (4.56 \pm 0.57) \times 10^{-3}$
B	0.5	0.5	✗	✗	✓	$\mathcal{L}_1 : (1.97 \pm 0.17) \times 10^{-2}$ $\mathcal{L}_2 : (1.49 \pm 0.12) \times 10^{-2}$ $\mathcal{L}_\infty : (2.58 \pm 0.22) \times 10^{-2}$	$\mathcal{L}_1 : (1.32 \pm 0.10) \times 10^{-2}$ $\mathcal{L}_2 : (9.71 \pm 0.76) \times 10^{-3}$ $\mathcal{L}_\infty : (1.61 \pm 0.13) \times 10^{-2}$
C	0.5	0.1	✗	✗	✓	$\mathcal{L}_1 : (1.14 \pm 0.15) \times 10^{-2}$ $\mathcal{L}_2 : (8.69 \pm 1.18) \times 10^{-3}$ $\mathcal{L}_\infty : (1.50 \pm 0.24) \times 10^{-2}$	$\mathcal{L}_1 : (7.90 \pm 1.64) \times 10^{-3}$ $\mathcal{L}_2 : (5.95 \pm 1.21) \times 10^{-3}$ $\mathcal{L}_\infty : (1.03 \pm 0.21) \times 10^{-2}$
D	0.5	0.1	✓	✗	✓	$\mathcal{L}_1 : (5.56 \pm 0.60) \times 10^{-3}$ $\mathcal{L}_2 : (4.23 \pm 0.43) \times 10^{-3}$ $\mathcal{L}_\infty : (7.44 \pm 0.80) \times 10^{-3}$	$\mathcal{L}_1 : (4.67 \pm 0.44) \times 10^{-3}$ $\mathcal{L}_2 : (3.50 \pm 0.34) \times 10^{-3}$ $\mathcal{L}_\infty : (5.97 \pm 0.62) \times 10^{-3}$
E	0.5 (x_1) 0.55 (x_2)	0.1	✗	✓	✓	$\mathcal{L}_1 : (8.08 \pm 0.79) \times 10^{-3}$ $\mathcal{L}_2 : (6.09 \pm 0.56) \times 10^{-3}$ $\mathcal{L}_\infty : (1.04 \pm 0.09) \times 10^{-2}$	$\mathcal{L}_1 : (4.91 \pm 0.63) \times 10^{-3}$ $\mathcal{L}_2 : (3.66 \pm 0.46) \times 10^{-3}$ $\mathcal{L}_\infty : (6.27 \pm 0.75) \times 10^{-3}$
F	0.5	0.1	✓	✓	✓	$\mathcal{L}_1 : (1.16 \pm 0.22) \times 10^{-2}$ $\mathcal{L}_2 : (8.91 \pm 1.65) \times 10^{-3}$ $\mathcal{L}_\infty : (1.59 \pm 0.30) \times 10^{-2}$	$\mathcal{L}_1 : (9.22 \pm 1.05) \times 10^{-3}$ $\mathcal{L}_2 : (7.12 \pm 0.81) \times 10^{-3}$ $\mathcal{L}_\infty : (1.27 \pm 0.15) \times 10^{-2}$
URP_BB	0.5	0.1	✓	✓	✗	$\mathcal{L}_1 : (9.91 \pm 1.71) \times 10^{-3}$ $\mathcal{L}_2 : (7.92 \pm 1.36) \times 10^{-3}$ $\mathcal{L}_\infty : (1.45 \pm 0.26) \times 10^{-2}$	$\mathcal{L}_1 : (9.99 \pm 1.41) \times 10^{-3}$ $\mathcal{L}_2 : (7.69 \pm 0.11) \times 10^{-3}$ $\mathcal{L}_\infty : (1.37 \pm 0.18) \times 10^{-2}$

Table A.2
Metrics Summary Table for URP CSTR data. This table showcases model performance of Black-box and Gray-box models. All models have arbitrary data Δt, partial observations and unknown initial conditions. Refer Section A.2 for details on metrics.

Case	Learnable kinetic functions	Learnable experimental parameters	Solution error	RHS error	Kinetic function error	Experimental parameter error
URP_BB	✗	✗	$\mathcal{L}_1 : (9.91 \pm 1.71) \times 10^{-3}$ $\mathcal{L}_2 : (7.92 \pm 1.36) \times 10^{-3}$ $\mathcal{L}_\infty : (1.45 \pm 0.26) \times 10^{-2}$	$\mathcal{L}_1 : (9.99 \pm 1.41) \times 10^{-3}$ $\mathcal{L}_2 : (7.69 \pm 0.11) \times 10^{-3}$ $\mathcal{L}_\infty : (1.37 \pm 0.18) \times 10^{-2}$	-	-
URP_GB1	✓	✗	$\mathcal{L}_1 : (1.37 \pm 0.18) \times 10^{-2}$ $\mathcal{L}_2 : (9.91 \pm 1.26) \times 10^{-3}$ $\mathcal{L}_\infty : (1.53 \pm 0.18) \times 10^{-2}$	$\mathcal{L}_1 : (6.24 \pm 0.72) \times 10^{-3}$ $\mathcal{L}_2 : (4.43 \pm 0.51) \times 10^{-3}$ $\mathcal{L}_\infty : (6.87 \pm 0.80) \times 10^{-3}$	$(5.31 \pm 0.62) \times 10^{-3}$	-
URP_GB2	✓	✓	$\mathcal{L}_1 : (1.68 \pm 0.24) \times 10^{-2}$ $\mathcal{L}_2 : (1.22 \pm 0.17) \times 10^{-2}$ $\mathcal{L}_\infty : (1.90 \pm 0.26) \times 10^{-2}$	$\mathcal{L}_1 : (8.97 \pm 2.87) \times 10^{-3}$ $\mathcal{L}_2 : (6.49 \pm 2.08) \times 10^{-3}$ $\mathcal{L}_\infty : (1.04 \pm 0.34) \times 10^{-2}$	$(7.77 \pm 2.25) \times 10^{-3}$	$(4.45 \pm 2.28) \times 10^{-2}$

Table A.3
Metrics Summary Table for B&F data. This table showcases model performance of Black-box and Gray-box models. All models have arbitrary data Δt, partial observations and unknown initial conditions. Errors are means and standard deviations of 10 training runs. Refer to Section A.2 for details on error metrics.

Case	Learnable kinetic functions	Learnable exp. parameters	Solution error from true LC	RHS Error	Kinetic function error	Exp. parameter error
BF_BB	✗	✗	$\mathcal{L}_1 : (6.26 \pm 0.81) \times 10^{-3}$ $\mathcal{L}_2 : (3.25 \pm 0.43) \times 10^{-3}$ $\mathcal{L}_\infty : (1.36 \pm 0.17) \times 10^{-2}$	$\mathcal{L}_1 : (4.00 \pm 0.77) \times 10^{-3}$ $\mathcal{L}_2 : (2.12 \pm 0.45) \times 10^{-3}$ $\mathcal{L}_\infty : (7.13 \pm 1.28) \times 10^{-3}$	-	-
BF_GB1	✓	✗	$\mathcal{L}_1 : (1.04 \pm 0.15) \times 10^{-2}$ $\mathcal{L}_2 : (5.16 \pm 0.68) \times 10^{-3}$ $\mathcal{L}_\infty : (1.90 \pm 0.24) \times 10^{-2}$	$\mathcal{L}_1 : (9.94 \pm 1.41) \times 10^{-3}$ $\mathcal{L}_2 : (5.43 \pm 0.80) \times 10^{-3}$ $\mathcal{L}_\infty : (1.68 \pm 0.23) \times 10^{-2}$	$\mathcal{L}_1 : (1.02 \pm 0.14) \times 10^{-3}$ $\mathcal{L}_2 : (7.66 \pm 1.14) \times 10^{-4}$ $\mathcal{L}_\infty : (1.24 \pm 0.17) \times 10^{-3}$	-
BF_GB2	✓	✓	$\mathcal{L}_1 : (1.22 \pm 0.57) \times 10^{-2}$ $\mathcal{L}_2 : (6.10 \pm 2.72) \times 10^{-3}$ $\mathcal{L}_\infty : (2.33 \pm 1.09) \times 10^{-2}$	$\mathcal{L}_1 : (1.37 \pm 0.60) \times 10^{-2}$ $\mathcal{L}_2 : (7.50 \pm 3.2) \times 10^{-3}$ $\mathcal{L}_\infty : (2.47 \pm 1.48) \times 10^{-2}$	$\mathcal{L}_1 : (1.48 \pm 0.71) \times 10^{-3}$ $\mathcal{L}_2 : (1.09 \pm 0.49) \times 10^{-3}$ $\mathcal{L}_\infty : (1.79 \pm 1.17) \times 10^{-3}$	$(7.51 \pm 2.1) \times 10^{-2}$

A.2. Metrics used

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{p}) && True\ ODE \\ \hat{\dot{\mathbf{x}}} &= \begin{cases} \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta) & Learned\ ODE\ (Black\ Box) \\ g(\mathbf{x}, \mathbf{p}, \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta)) & Learned\ ODE\ (Gray\ Box) \end{cases} \end{aligned} \tag{A.1}$$

A.2.1. Short-term prediction error

Let $\mathbf{x}_j^{LC} = \{x_i^{LC}\}_j$ be the steady-state solution if it is stable, and a point on the limit cycle (obtained by solving a Poincaré map) if the steady-state solution is unstable. Let N_v be the number of variables, N_n the number of testing points, and T the time period for short-term transients.

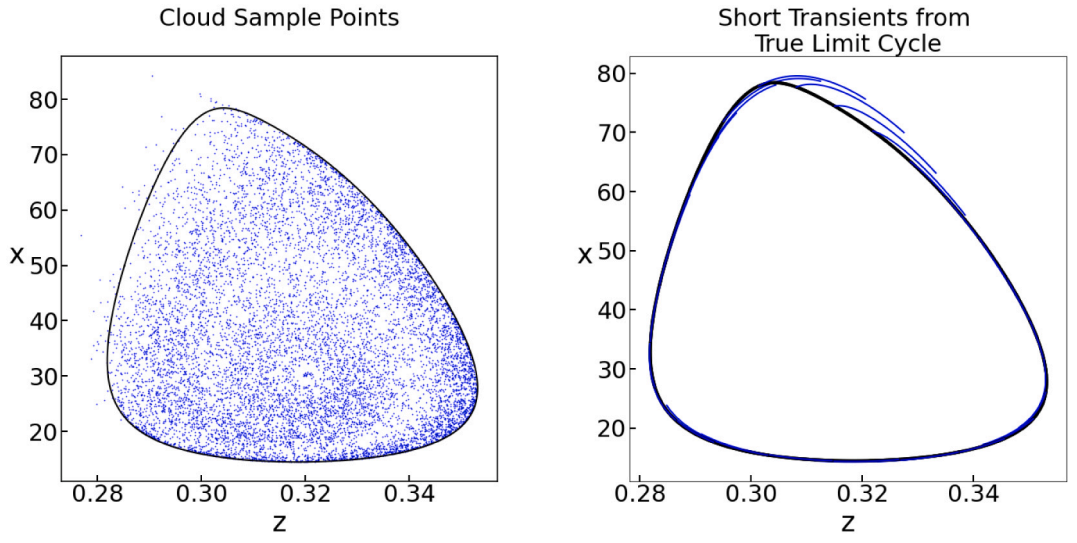
Starting from $\mathbf{x}_j^{LC}$, the true ODEs and the ANN learned ODEs are integrated for $t = T$ (Fig. A.1 (b)).

$$\begin{aligned} \mathbf{x}_j(t=0) &= \mathbf{x}_j^{LC}, \quad 1 \le j \le N_n \\ \mathbf{y}_j &= \int_0^T f(\mathbf{x}, \mathbf{p}) dt, \quad 1 \le j \le N_n \\ \hat{\mathbf{y}}_j &= \begin{cases} \int_0^T \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta) dt, \quad 1 \le j \le N_n & (Black\ Box) \\ \int_0^T g(\mathbf{x}, \mathbf{p}, \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta)) dt, \quad 1 \le j \le N_n & (Gray\ Box) \end{cases} \end{aligned} \tag{A.2}$$

$\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ are the true and predicted matrices. These matrices are condensed into an error metric using various norms defined in Appendix A.2.5.

Table A.4
Metrics Summary Table for B&F data. This table showcases the resonance effect. Refer Section A.2 for details on metrics.

Case	Initial scaling	Randomized data Δt	Randomized solver δt	Solution error from true LC	RHS error
BF_BB_A	$\times 100^{0}$	✗	✗	$\mathcal{L}_1 : (8.02 \pm 4.02) \times 10^{-1}$ $\mathcal{L}_2 : (4.33 \pm 2.58) \times 10^{-1}$ $\mathcal{L}_\infty : 2.03 \pm 1.43$	$\mathcal{L}_1 : 14.6 \pm 2.56$ $\mathcal{L}_2 : 8.47 \pm 1.63$ $\mathcal{L}_\infty : 38.3 \pm 9.43$
BF_BB_B	$\times 100^{-1}$	✗	✗	$\mathcal{L}_1 : (7.84 \pm 0.92) \times 10^{-3}$ $\mathcal{L}_2 : (4.13 \pm 0.53) \times 10^{-3}$ $\mathcal{L}_\infty : (1.72 \pm 0.24) \times 10^{-2}$	$\mathcal{L}_1 : (5.29 \pm 1.50) \times 10^{-3}$ $\mathcal{L}_2 : (2.89 \pm 0.92) \times 10^{-3}$ $\mathcal{L}_\infty : (9.23 \pm 2.49) \times 10^{-3}$
BF_BB_C	$\times 100^{0}$	✗	✓	$\mathcal{L}_1 : (1.63 \pm 0.70) \times 10^{-1}$ $\mathcal{L}_2 : (8.40 \pm 3.52) \times 10^{-2}$ $\mathcal{L}_\infty : (3.50 \pm 1.43) \times 10^{-1}$	$\mathcal{L}_1 : 2.77 \pm 1.22$ $\mathcal{L}_2 : 1.65 \pm 0.68$ $\mathcal{L}_\infty : 7.59 \pm 3.38$
BF_BB_D	$\times 100^{0}$	✓	✓ (Because of data)	$\mathcal{L}_1 : (2.29 \pm 0.39) \times 10^{-2}$ $\mathcal{L}_2 : (1.14 \pm 0.17) \times 10^{-2}$ $\mathcal{L}_\infty : (4.46 \pm 0.66) \times 10^{-2}$	$\mathcal{L}_1 : (4.57 \pm 1.00) \times 10^{-2}$ $\mathcal{L}_2 : (2.88 \pm 0.68) \times 10^{-2}$ $\mathcal{L}_\infty : (1.21 \pm 0.31) \times 10^{-1}$
BF_BB_E	$\times 100^{-1}$	✓	✓ (Because of data)	$\mathcal{L}_1 : (7.99 \pm 1.33) \times 10^{-3}$ $\mathcal{L}_2 : (4.18 \pm 0.71) \times 10^{-3}$ $\mathcal{L}_\infty : (1.74 \pm 0.30) \times 10^{-2}$	$\mathcal{L}_1 : (5.69 \pm 1.01) \times 10^{-3}$ $\mathcal{L}_2 : (3.10 \pm 0.56) \times 10^{-3}$ $\mathcal{L}_\infty : (1.02 \pm 0.18) \times 10^{-2}$



(a) Sampling for Sections A.4 and A.5

(b) Short Term Transients for Section A.3

Fig. A.1. (a) Sampling points were obtained by randomizing initial conditions, integrating in time and discarding the early transients, and sub-sampling from 1000 different trajectories. (b) Short term transients obtained by taking as initial conditions 50 points evenly spaced around the true limit cycle and integrating for $t = 20$.

A.2.2. RHS error

Visualization of sampling points is given in Fig. A.1 (a).

$$\mathbf{y}_j = f(\mathbf{x}, \mathbf{p}), \quad 1 \le j \le N_n$$
$$\hat{\mathbf{y}}_j = \begin{cases} \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta), & 1 \le j \le N_n \quad (Black\ \ Box) \\ g(\mathbf{x}, \mathbf{p}, \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta)), & 1 \le j \le N_n \quad (Gray\ \ Box) \end{cases} \tag{A.3}$$

$\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ are the true and predicted matrices. These matrices are condensed into an error metric using various norms defined in Appendix A.2.5.

A.2.3. Gray-box function error

For Gray-Box models, the neural network learns a subset of the full ODE function. Visualization of sampling points is given in Fig. A.1 (a).

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{p}) = g(\mathbf{x}, \mathbf{p}, \phi(\mathbf{x}, \mathbf{p})) \quad True\ \ ODE$$
$$\hat{\dot{\mathbf{x}}} = g(\mathbf{x}, \mathbf{p}, \hat{\phi}(\mathbf{x}, \mathbf{p})) \quad Learned\ \ ODE \tag{A.4}$$

$$\mathbf{y}_j = \phi(\mathbf{x}, \mathbf{p}), \quad 1 \le j \le N_n$$
$$\hat{\mathbf{y}}_j = \hat{\phi}(\mathbf{x}, \mathbf{p}) = \mathcal{N}(\mathbf{x}, \mathbf{p}; \theta), \quad 1 \le j \le N_n \tag{A.5}$$

$\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ are the true and predicted matrices. These matrices are condensed into an error metric using various norms defined in Appendix A.2.5.

A.2.4. Experimental parameter error

Some Gray-Box models include additional parameters that have physical and experimental significance, and these are fit during ANN training. For these parameters, relative error is used.

Let $\boldsymbol{\kappa} = \{\kappa\}_k$ be the experimental parameters, with $\hat{\boldsymbol{\kappa}} = \{\hat{\kappa}\}_k$ the ANN predictions, and N_P the number of learnable experimental parameters.

$$\mathcal{L}(\boldsymbol{\kappa}, \hat{\boldsymbol{\kappa}}) = \frac{1}{N_P} \sum_{k=1}^{N_P} \left| \frac{\kappa_k - \hat{\kappa}_k}{\kappa_k} \right| \tag{A.6}$$

A.2.5. Norms

The error matrices $\mathbf{y}_j$ and $\hat{\mathbf{y}}_j$ are first normalized with min–max scaling per variable. $\hat{\mathbf{y}}_j$ is scaled with the min/max of $\mathbf{y}_j$.

$$\tilde{\mathbf{y}}_j = \frac{\mathbf{y}_j - \min(\mathbf{y})}{\max(\mathbf{y}) - \min(\mathbf{y})}$$
$$\tilde{\hat{\mathbf{y}}}_j = \frac{\mathbf{y}_j - \min(\hat{\mathbf{y}})}{\max(\mathbf{y}) - \min(\mathbf{y})} \tag{A.7}$$

These normalized error matrices are condensed into a single metric using various norms:

$$\mathcal{L}_1(\tilde{\mathbf{y}}_j, \tilde{\hat{\mathbf{y}}}_j) = \frac{1}{N_n N_V} \sum_{j=1}^{N_n} \sum_{i=1}^{N_V} |\tilde{y}_{i,j} - \tilde{\hat{y}}_{i,j}|$$
$$\mathcal{L}_2(\tilde{\mathbf{y}}_j, \tilde{\hat{\mathbf{y}}}_j) = \frac{1}{N_n N_V} \sum_{j=1}^{N_n} \sqrt{\sum_{i=}^{N_V} (\tilde{y}_{i,j} - \tilde{\hat{y}}_{i,j})^2} \tag{A.8}$$
$$\mathcal{L}_\infty(\tilde{\mathbf{y}}_j, \tilde{\hat{\mathbf{y}}}_j) = \frac{1}{N_n} \sum_{j=1}^{N_n} \max_i |\tilde{y}_{i,j} - \tilde{\hat{y}}_{i,j}|$$

References

Antonios, Zagaris, Vandekerckhove, Christophe, William Gear, C., Kaper, Tasso J., Kevrekidis, Loannis G., 2012. Stability and stabilization of the constrained runs schemes for equation-free projection to a slow manifold. Discrete Contin. Dyn. Syst. 32 (8), 2759–2803.

Balmaseda, Magdalena A., Alves, Oscar J., Arribas, Alberto, Awaji, Toshiyuki, Behringer, David W., Ferry, Nicolas, Fujii, Yosuke, Lee, Tong, Rienecker, Michele, Rosati, Tony, Stammer, Detlef, 2009. Ocean initialization for seasonal forecasts. Oceanography 22 (SPL.ISS. 3), 154–159.

Baltzis, B.C., Frederickson, A., 1984. Coexistence of two microbial populations competing for a renewable resource in a non-predator-prey system. Bull. Math. Biol. 46 (1), 155–174.

Benny Toomarian, Nikzad, Barhen, Jacob, 1992. Learning a trajectory using adjoint functions and teacher forcing. Neural Netw. 5 (3), 473–484.

Bertalan, Tom, Dietrich, Felix, Mezić, Igor, Kevrekidis, Ioannis G., 2019. On learning Hamiltonian systems from data. Chaos 29 (12).

Bertsimas, Dimitris, Orfanoudaki, Agni, Pawlowski, Colin, 2021. Imputation of clinical covariates in time series. Mach. Learn. 110 (1), 185–248.

Bouvrie, Jake, Hamzi, Boumediene, 2017. Kernel methods for the approximation of nonlinear systems. SIAM J. Control Optim. 55 (4), 2460–2492.

Buisson-Fenet, Mona, Morgenthaler, Valery, Trimpe, Sebastian, Di Meglio, Florent, 2022. Recognition Models to Learn Dynamics from Partial Observations with Neural ODEs. Technical Report.

Chen, Ricky T.Q., 2018. torchdiffeq.

Chen, L., Bernard, O., Bastin, G., Angelov, P., 2000. Hybrid modelling of biotechnological processes using neural networks. Control Eng. Pract. 8 (7), 821–827.

Chen, S., Billings, S.A., Grant, P.M., 1990. Non-linear system identification using neural networks. Internat. J. Control 51 (6), 1191–1214.

Chen, Ricky T.Q., Rubanova, Yulia, Bettencourt, Jesse, Duvenaud, David, 2018. Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, vol. 2018-Decem, pp. 6571–6583.

Davie, A.M., 2008. Differential equations driven by rough paths: An approach via discrete approximation. Appl. Math. Res. Express. 2008.

De Veaux, Richard D., Bain, Rod, Ungar, Lyle H., 1999. Hybrid neural network models for environmental process control: (The 1998 Hunter Lecture). Environmetrics 10 (3), 225–236.

Dietrich, Felix, Makeev, Alexei, Kevrekidis, George, Evangelou, Nikolaos, Bertalan, Tom, Reich, Sebastian, Kevrekidis, Ioannis G., 2021. Learning Effective Stochastic Differential Equations from Microscopic Simulations: linking Stochastic Numerics to Deep Learning. Technical Report.

Farber, Robert M., Lapedes, Alan S., Rico-Martínez, Ramiro, Kevrekidis, Ioannis G., 1993. Identification of Continuous-Time Dynamical Systems: Neural Network Based Algorithms and Parallel Implementation. In: Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Vol. 1. pp. 287–291.

Gear, C.W., Kaper, T.J., Kevrekidis, I.G., Zagaris, A., 2005. Projecting to a slow manifold: Singularly perturbed systems and legacy codes. SIAM J. Appl. Dyn. Syst. 4 (3), 711–732.

Gelman, Andrew, Hill, Jennifer, 2006. Data Analysis using Regression and Multilevel/Hierarchical Models. Cambridge University Press.

González-García, R., Rico-Martínez, R., Kevrekidis, I.G., 1998. Identification of distributed parameter systems: A neural net based approach. In: Computers and Chemical Engineering, vol. 22, (no. SUPPL.1), pp. 965–968.

Gordon, N.J., Salmond, D.J., Smith, A.F.M., 1993. Novel approach to nonlinear/non-gaussian Bayesian state estimation. IEE Proc., Part F: Radar Signal Process. 140 (2), 107–113.

Greydanus, Sam, Dzamba, Misko, Yosinski, Jason, 2019. Hamiltonian neural networks. In: Advances in Neural Information Processing Systems, vol. 32.

Hagge, Tobias, Stinis, Panos, Yeung, Enoch, Tartakovsky, Alexandre M., 2017. Solving differential equations with unknown constitutive relations as recurrent neural networks.

Hamzi, Boumediene, Owhadi, Houman, 2021. Learning dynamical systems from data: A simple cross-validation perspective, part I: Parametric kernel flows. Physica D 421, 132817.

Honaker, James, King, Gary, 2010. What to do about missing values in time-series cross-section data. Am. J. Political Sci. 54 (2), 561–581.

Hudson, J.L., Kube, M., Adomaitis, R.A., Kevrekidis, I.G., Lapedes, A.S., Farber, R.M., 1990. Nonlinear signal processing and system identification: Applications to time series from electrochemical reactions. Chem. Eng. Sci. 45 (8), 2075–2081.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. J. Fluids Eng., Trans. ASME 82 (1), 35–45.

Kemeth, Felix P., Alonso, Sergio, Echebarria, Blas, Moldenhawer, Ted, Beta, Carsten, Kevrekidis, Ioannis G., 2022. Black and Gray Box Learning of Amplitude Equations: Application to Phase Field Systems. Technical report.

Kemeth, Felix P., Bertalan, Tom, Evangelou, Nikolaos, Cui, Tianqi, Malani, Saurabh, Kevrekidis, Ioannis G., 2021. Initializing LSTM internal states via manifold learning. Chaos 31 (9), 093111.

Kidger, Patrick, 2022. On Neural Differential Equations.

Kidger, Patrick, Morrill, James, Foster, James, Lyons, Terry, 2020. Neural controlled differential equations for irregular time series. In: Advances in Neural Information Processing Systems, vol. 2020-Decem, pp. 6696–6707.

Krishnapriyan, Aditi S., Queiruga, Alejandro F., Erichson, N. Benjamin, Mahoney, Michael W., 2022. Learning continuous models for continuous physics.

Kuschewski, John G., Żak, Stanislaw H., Hui, Stefen, 1993. Application of feedforward neural networks to dynamical system identification and control. IEEE Trans. Control Syst. Technol. 1 (1), 37–49.

Lee, Seungjoon, Psarellis, Yorgos M., Siettos, Constantinos I., Kevrekidis, Ioannis G., 2022. Learning Black- and Gray-Box Chemotactic PDEs/closures from Agent Based Monte Carlo Simulation Data. Technical Report.

Ljung, Lennart, 1979. Asymptotic behavior of the extended Kalman filter as a parameter estimator for linear systems. IEEE Trans. Automat. Control 24 (1), 36–50.

Lovelett, Robert J., Avalos, José L., Kevrekidis, Ioannis G., 2020. Partial observations and conservation laws: Gray-box modeling in Biotechnology and optogenetics. Ind. Eng. Chem. Res. 59 (6), 2611–2620.

Lu, Lu, Jin, Pengzhan, Pang, Guofei, Zhang, Zhongqiang, Karniadakis, George Em, 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat. Mach. Intell. 3 (3), 218–229.

Lyngdoh, Gideon A., Zaki, Mohd, Krishnan, N.M. Anoop, Das, Sumanta, 2022. Prediction of concrete strengths enabled by missing data imputation and interpretable machine learning. Cem. Concr. Compos. 128, 104414.

Menesklou, Philipp, Sinn, Tabea, Nirschl, Hermann, Gleiss, Marco, 2021. Grey box modelling of decanter centrifuges by coupling a numerical process model with a neural network. Minerals 11 (7), 755.

Nelles, Oliver, 2001. Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. p. 785.

Nelwamondo, Fulufhelo V., Mohamed, Shakir, Marwala, Tshilidzi, 2007. Missing data: A comparison of neural network and expectation maximization techniques. Current Sci. 93 (11), 1514–1521.

Oliveira, R., 2004. Combining first principles modelling and artificial neural networks: A general framework. In: Computers and Chemical Engineering, vol. 28, (no. 5), Pergamon, pp. 755–766.

Oluwaseye, Luke, A1, Joel *, Doorsamy, Wesley, Paul, Sena, 2022. A review of missing data handling techniques for machine learning. Int. J. Innov. Technol. Interdiscipl. Sci. www.IJITIS.org 5 (3), 971–1005.

Pineda, Fernando J., 1988. Dynamics and architecture for neural computation. J. Complexity 4 (3), 216–245.

Psarellis, Yorgos M., Lee, Seungjoon, Bhattacharjee, Tapomoy, Datta, Sujit S., Bello-Rivas, Juan M., Kevrekidis, Ioannis G., 2022. Data-driven discovery of chemotactic migration of bacteria via machine learning.

Psichogios, Dimitris C., Ungar, Lyle H., 1992. A hybrid neural network-first principles approach to process modeling. AIChE J. 38 (10), 1499–1511.

Purwar, S., Kar, I.N., Jha, A.N., 2007. Nonlinear system identification using neural networks. IETE J. Res. 53 (1), 35–42.

Raissi, Maziar, Perdikaris, Paris, Karniadakis, George Em, 2017. Physics informed deep learning (Part II): Data-driven discovery of nonlinear partial differential equations.

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

Rico-Martinez, R., Anderson, J.S., Kevrekidis, I.G., 1994. Continuous-time nonlinear signal processing: A neural network based approach for gray box identification. In: Neural Networks for Signal Processing - Proceedings of the IEEE Workshop. IEEE, pp. 596–605.

Rico-Martinez, Ramiro, Kevrekidis, Ioannis G., 1993. Continuous time modeling of nonlinear systems: A neural network-based approach. In: 1993 IEEE International Conference on Neural Networks. Publ by IEEE, pp. 1522–1525.

Rico-Martinez, R., Krischer, K., Kube, M.C., Hudson, J.L., Kevrekidis, I.G., 1992. Discrete- Vs. Continuous-time nonlinear signal processing of Cu electrodissolution data. Chem. Eng. Commun. 118 (1), 25–48.

Rubanova, Yulia, Chen, Ricky T.Q., Duvenaud, David, 2019. Latent ODEs for irregularly-sampled time series. In: Advances in Neural Information Processing Systems, vol. 32.

Saadatnejad, Saeed, Oveisi, Mohammadhosein, Hashemi, Matin, 2020. LSTM-based ECG classification for continuous monitoring on personal wearable devices. IEEE J. Biomed. Health Inf. 24 (2), 515–523.

Shi, Guanya, Shi, Xichen, O'Connell, Michael, Yu, Rose, Azizzadenesheli, Kamyar, Anandkumar, Animashree, Yue, Yisong, Chung, Soon Jo, 2019. Neural lander: Stable drone landing control using learned dynamics. In: Proceedings - IEEE International Conference on Robotics and Automation, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., pp. 9784–9790.

Silva-Ramírez, Esther Lydia, Pino-Mejías, Rafael, López-Coello, Manuel, 2015. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. Appl. Soft Comput. 29, 65–74.

Thompson, Michael L., Kramer, Mark A., 1994. Modeling chemical processes using prior knowledge and neural networks. AIChE J. 40 (8), 1328–1340.

Uppal, A., Ray, W.H., Poore, A.B., 1974. On the dynamic behavior of continuous stirred tank reactors. Chem. Eng. Sci. 29 (4), 967–985.

Van Can, H.J.L., Te Braake, H.A.B., Hellinga, C., Luyben, K.C.A.M., Heijnen, J.J., 1997. An efficient model development strategy for bioprocesses based on neural networks in macroscopic balances. Biotechnol. Bioeng. 54 (6), 549–566.

Vandekerckhove, Christophe, Kevrekidis, Ioannis, Roose, Dirk, 2009. An efficient Newton-Krylov implementation of the constrained runs scheme for initializing on a slow manifold. J. Sci. Comput. 39 (2), 167–188.

Wan, E.A., Van Der Merwe, R., 2000. The unscented Kalman filter for nonlinear estimation. In: IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium. AS-SPCC 2000, Institute of Electrical and Electronics Engineers Inc., pp. 153–158.

Wang, Yu, 2017. A new concept using LSTM neural networks for dynamic system identification. In: Proceedings of the American Control Conference. Institute of Electrical and Electronics Engineers Inc., pp. 5324–5329.

Williams, Ronald J., Zipser, David, 1989. A learning algorithm for continually running fully recurrent neural networks. In: Neural Computation, vol. 1, (no. 2), pp. 270–280.

Yildirim, Özal, 2018. A novel wavelet sequences based on deep bidirectional LSTM network model for ECG signal classification. Comput. Biol. Med. 96, 189–202.

Zagaris, Antonios, Gear, C. William, Kaper, Tasso J., Kevrekidis, Yannis G., 2009. Analysis of the accuracy and convergence of equation-free projection to a slow manifold. Math. Model. Numer. Anal. 43 (4), 757–784.

Zhu, Aiqing, Bertalan, Tom, Zhu, Beibei, Tang, Yifa, Kevrekidis, Ioannis G., 2023. Implementation and (Inverse Modified) error analysis for implicitly-templated ODE-nets.

Zhu, Aiqing, Jin, Pengzhan, Zhu, Beibei, Tang, Yifa, 2022. On numerical integration in neural ordinary differential equations. In: Proceedings of the 39th International Conference on Machine Learning, vol. 162, no. 17–23 Jul. PMLR, pp. 27527–27547.