



OPEN

A trajectory-based loss function to learn missing terms in bifurcating dynamical systems

Rahel Vortmeyer-Kley^{1,2}✉, Pascal Nieters^{1,2} & Gordon Pipa¹

Missing terms in dynamical systems are a challenging problem for modeling. Recent developments in the combination of machine learning and dynamical system theory open possibilities for a solution. We show how physics-informed differential equations and machine learning—combined in the Universal Differential Equation (UDE) framework by Rackauckas et al.—can be modified to discover missing terms in systems that undergo sudden fundamental changes in their dynamical behavior called bifurcations. With this we enable the application of the UDE approach to a wider class of problems which are common in many real world applications. The choice of the loss function, which compares the training data trajectory in state space and the current estimated solution trajectory of the UDE to optimize the solution, plays a crucial role within this approach. The Mean Square Error as loss function contains the risk of a reconstruction which completely misses the dynamical behavior of the training data. By contrast, our suggested trajectory-based loss function which optimizes two largely independent components, the length and angle of state space vectors of the training data, performs reliable well in examples of systems from neuroscience, chemistry and biology showing Saddle-Node, Pitchfork, Hopf and Period-doubling bifurcations.

Interacting oceanic flows governing our climate, increasing CO₂ levels causing global warming, competing species for coexistence, dominance or extinction—we live in a changing world. Understanding the impact of these often slow and gradual changes is important. But sometimes, a system can undergo a sudden fundamental change and end up in a totally different dynamical behavior. These points are sometimes called tipping points or regime shifts. Since the middle of the last century, the theory of dynamical systems has been applied to characterize the behavior of changing systems and real world phenomena in particular^{1–5} and examples therein.

The dynamics of the system can be described by the behavior of trajectories in a space defined by the state variables of the system. This state space contains repelling, attracting and separating structures—namely stable and unstable manifolds, and separatrices, and different types of attractors and coherent structures—that can act as organizing structures by governing the dynamical behavior of trajectories^{6,7}.

When modeling real world systems in simple equations we still face difficulties because our knowledge or understanding of the underlying processes is limited (e.g.^{8–10}). Here machine learning can help discover missing knowledge about the dynamical relationships between state variables from observed data by statistical inference. For example, Universal Differential Equations (UDE)¹¹ are a recently proposed method to learn dynamical systems from data with machine learning and can be combined with the Sparse Identification of Nonlinear Dynamics (SInDy) algorithm¹² to estimate an algebraic form of the dynamical system from data (see the next section for details). We show how this approach can be applied to learn missing terms in systems that can undergo sudden fundamental changes in their dynamics called bifurcation.

The goal of this work is to investigate the role of the loss function used to compare the learned dynamics and the training data given as time series data in learning a UDE. In particular, we propose a new loss function for optimization that compares angle and length of vectors in state space independently. Usually, the mean-squared error is used to compare time series for each variable in the system independently. Our idea is that this new Length Difference and Angle Difference (LDA) loss is more reliable when learning a UDE in many bifurcating systems and therefore is better suited to find missing terms in bifurcating systems. The examples we use for this comparison cover on the one hand a broad range of different bifurcation types (Saddle-Node, Pitchfork, Hopf and Period-doubling bifurcation) and on the other hand show the importance of bifurcations in various fields of research (e.g. neuroscience, biology, chemistry). Firstly, we set up a statistical comparison of how well the two loss functions perform in each of these systems in two different parameter regimes representing two different

¹Institute of Cognitive Science, Osnabrück University, Wachsbleiche 27, 49090 Osnabrück, Germany. ²These authors contributed equally: Rahel Vortmeyer-Kley and Pascal Nieters. ✉email: rahel.vortmeyer-kley@uni-osnabrueck.de

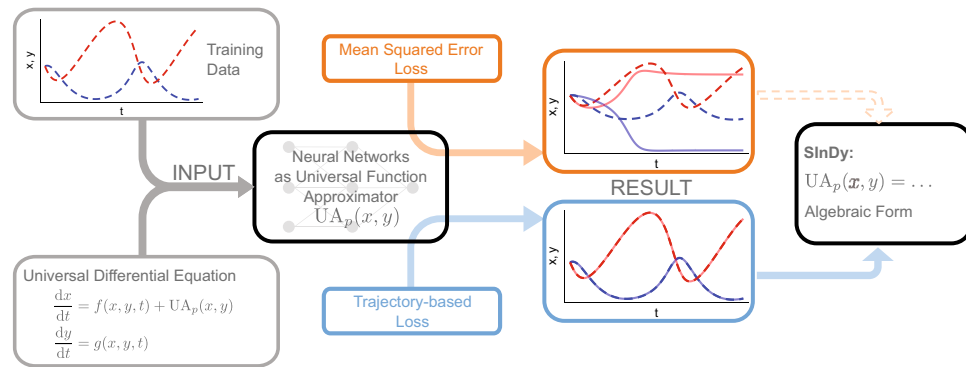


Figure 1. Schematic sketch of the concept of UDE¹¹. Depending on a successful approximation of the training data, the application of SInDy¹² is possible. The time series qualitatively correspond to the Selkov model¹⁵ in Table 2 second row. This figure is plotted using Inkscape (Version 1.0.1, <https://inkscape.org>) and Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>).

dynamics respectively. We then narrow this comparison by focusing on nested intervals around the bifurcation parameter as well as different trajectory starting points in state space. Finally, we show by way of example that an algebraic reconstruction of the missing terms using SInDy is possible if a UDE was trained successfully and discuss our results.

Universal differential equations (UDE) and sparse identification of nonlinear dynamics (SInDy). In this work, we investigate a complete, data-driven pipeline to discover the equations that describe a dynamical system in the context of bifurcating systems. The approach follows recently published work by Rackauckas *et al.* and combines the estimation of a UDE¹¹ from data and the subsequent identification of algebraic terms from the UDE using SInDy¹². We have illustrated the main steps involved in Fig. 1, starting with time series measurements from the system in question which will be used as training data.

Firstly, a Universal Differential Equation generally describes the dynamics of a system as $\dot{\mathbf{x}} = f(\mathbf{x}, t, \text{UA}_p(\mathbf{x}, t))$ where the function UA_p is a parameterized and differentiable universal function approximator. For example, a partially known two dimensional system with variables x and y may be described as:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y, t) + \text{UA}_p(x, y) \\ \frac{dy}{dt} &= g(x, y, t) \end{aligned} \quad (1)$$

where functions f and g describe the known dynamics for x and y respectively, but we want to estimate an additional additive term in the dynamics of x . Given an initial $x(t_0)$ and $y(t_0)$ as well as initial parameters p_{init} the UDE can numerically be solved and trajectories $\hat{x}(t)$ and $\hat{y}(t)$ are estimated. Using a loss function, the Mean Squared Error loss (cf. next section) for each variable of a system as in the original work, the estimated trajectories based on the UDE can be compared against the actual system measured at sample points t_i . The entire program, from the definition of the unknown function UA_p and the UDE through to the numerical solution of the system and the calculation of the loss, can be differentiated with respect to parameters p using automatic differentiation in the Julia programming language^{13,14}. This enables the optimization of these parameters by gradient descent on the loss function, and thereby allows a user to find a UDE with parameterized function $\text{UA}_p^{\text{optimal}}$ that approximates the measured time series of the original system well.

Using differentiable models in machine learning problems is a generalized view of highly successful deep and artificial neural network (ANN) models¹⁶ that have previously been used to solve differential equations^{17,18}. Physics-informed neural networks, for example, aim to combine advantages of data-driven machine learning with knowledge about underlying physical laws in the training process to reconstruct dynamics of a system^{19–21}. In this paper, we use ANNs as the parameterized universal function approximator^{22,23} UA_p . The disadvantage is that neural networks are black boxes and do not allow us to learn about systematic relationships between the variables of the dynamical systems that may be the drivers of underlying dynamics.

Therefore, we also use the SInDy algorithm¹² to identify algebraic terms that can replace the neural network black box. In SInDy, finding the algebraic form of a differential equation is formulated as a linear regression problem. A matrix of non-linear functions Θ applied to the state vector \mathbf{x} of the system multiplied by a matrix of sparse coefficient vectors Ξ recovers the standard form of many non-linear dynamical systems: $\dot{\mathbf{x}} = f(\mathbf{x}) = \Theta(\mathbf{x})\Xi$ (see the original paper¹² for examples). If measurements for both \mathbf{x} and $\dot{\mathbf{x}}$ are available at several sample points in time we can write $\dot{X} = \Theta(X)\Xi + \eta Z$ where X and \dot{X} have rows for each sample point, $\Theta(X)$ is a design matrix of non-linear functions applied to the data, and ηZ is independent and identically distributed Gaussian noise with magnitude η . This is a standard linear regression problem in multiple variables²⁴ where finding the sparse coefficient vectors $\Xi = [\xi_1, \dots, \xi_n]$ for a system with n state variables by using for example the LASSO algorithm²⁵ can find those non-linear functions in \mathbf{x} in the design matrix that best explain the data.

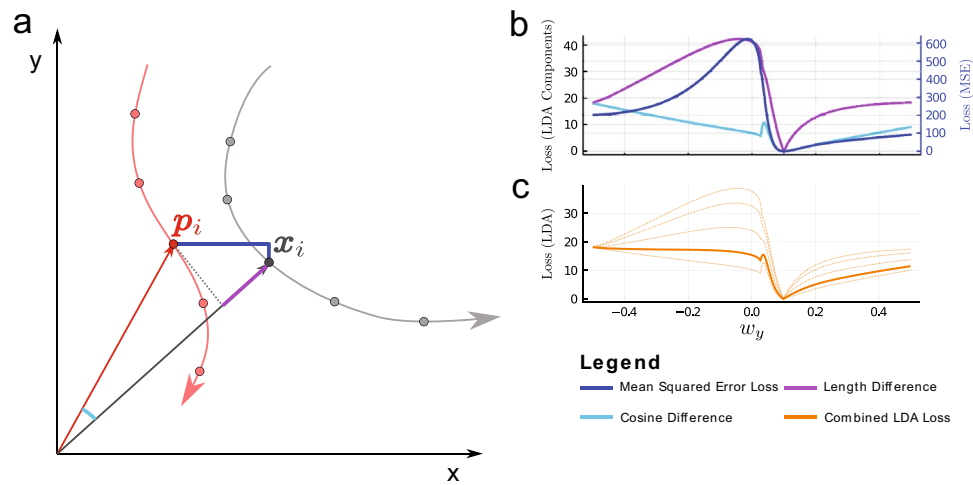


Figure 2. Schematic sketch of the concept of MSE and LDA loss function. (a) The UDE estimates trajectories $P = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$ (red) in state space, the training data is $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ (gray). The MSE measures relative difference in each dimension of the sample point independently (blue lines), whereas the LDA measures the difference in angle (cyan) and difference in vector length (purple). (b) Comparing trajectories for the Selkov model¹⁵ (cf. Table 2 second row) based on MSE (blue), angular difference (cyan) via the cosine similarity and length difference (purple) all show a maximum that splits the loss landscape into two regions, one where the correct minimum can be reached and one where it cannot. Trajectories are compared based on a simple regression model in which only the weight for y : w_y in the Selkov model is unknown and gives the x -axis here. (c) A weighted sum of the angular and length difference gives the LDA loss. Different sums are shown, with $k_1 = 0.25$ (length) and $k_2 = 0.75$ (angle) marked as the thick orange line that gives a good compromise between the two components. This figure is plotted using Inkscape (Version 1.0.1, <https://inkscape.org>) and Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>).

Often, data for \mathbf{x} is not abundant and $\dot{\mathbf{x}}$ is not immediately available. In these cases, we can still estimate the UDE that fits the data. Given a UDE, it is easy to calculate $\dot{\mathbf{x}}$ from any given \mathbf{x} and SInDy can be applied. Doing so will find functions in \mathbf{x} that are much easier to understand and analyze than a neural network and we have successfully removed the black box from the UDE. Furthermore, if partial knowledge about the system is included in the UDE, we can still use SInDy to find algebraic terms for the contribution by solving the regression problem $U\mathbf{A}_p \text{optimal}(\mathbf{x}) = \Theta(\mathbf{x})\Xi + \eta Z$ in isolation and complete the algebraic description of the dynamical system.

Results

The loss function and the trajectories. Generally speaking, the objective function or loss function in an optimization problem must be chosen in accordance with the goal of the optimized model. In neural network regression models, which we use here to find an approximation to missing terms in the UDE, this is typically done by minimization of the Mean Squared Error (MSE) loss function²⁶. However, more specialized practitioners of model optimization have sometimes found that particular choices of one subtly different loss function over another can significantly alter the quality of the model fit, for example in financial modeling of option pricing²⁷.

In the UDE approach, the MSE is used to compare observed dynamics to the estimated dynamics of the UDE by treating each component of the i -th sampled state space vector \mathbf{x}_i (observed) and \mathbf{p}_i (estimated) independently:

$$L_{\text{MSE}} = \sum_{i=1}^n \sum_{j=1}^m (\mathbf{x}_i^j - \mathbf{p}_i^j)^2, \quad (2)$$

where \mathbf{x}_i^j resp. \mathbf{p}_i^j is the j -th component in m dimensional state space, and the i -th point on a trajectory of length n (cf. blue difference vectors in Fig. 2a). This can work well in practice in many different UDE problems^{11,28}. However, when trying to find a missing term in a potentially bifurcating system, this particular choice of loss function can lead to unsatisfactory results.

For example, if the data generating system can undergo a Hopf bifurcation from a stable state solution to an oscillatory solution, a UDE trained with an MSE loss often does not achieve dynamics in the correct regime and does not produce dynamics that fit the original system's structure of nullclines and attractors (see animated state space portrait example of the oscillatory state of the Selkov model¹⁵ in the Supplementary video material S1 panel c).

How can the optimization fail so spectacularly? If we simplify the problem and replace the neural network by a simple linear regression with three parameters, one for each dimension of the state space and a bias parameter, we can visualize the loss function across any axis. If bias and weight for state variable x from the example in Fig. 1 (cf. Selkov model in Table 2 second row) are chosen correctly as 0, the correct solution for the weight for y is 0.1 in this case. Indeed, the MSE loss has its minimum at the correct value (Fig. 2b blue line), but it also has

a significant peak just below 0. For values smaller than 0, the gradient-descent based optimization of the UDE runs into an alternative local minimum. This alternative local minimum acts like a “black hole of loss” in the loss landscape and cannot be escaped. It becomes impossible to find the true solution. Thus, the quality of the approximation is entirely dependent on initial conditions (see results below as well as the animated state space portrait in the Supplementary video material S1 panel c).

In the setup using a neural network, we also observe the phenomenon that trajectories of the estimated and real system must first diverge so that the correct dynamical regime can be reached. While the loss landscape over many ANN parameters is not easily visualized, it can still contain local minima that represent alternative stable solutions and act as “black holes of loss”.

We realized that this problem is unlikely to be solved by a single loss function and were inspired by the view of the dynamics as a trajectory of state space vectors. These can also be described by their angle and length always having the origin in state space as their common reference.

We construct the Length Difference and Angle Difference Loss (LDA):

$$L_{LDA} = \sum_{i=1}^n \left(k_1 \cdot \underbrace{\frac{\sqrt{(|\mathbf{x}_i| - |\mathbf{p}_i|)^2}}{|\mathbf{x}_i| + |\mathbf{p}_i|}}_{\text{length difference}} + k_2 \cdot \underbrace{\frac{1}{2} \cdot \left(1 - \frac{\mathbf{x}_i \cdot \mathbf{p}_i}{|\mathbf{x}_i| \cdot |\mathbf{p}_i|} \right)}_{\text{angle difference}} \right), \quad (3)$$

where the first component describes the normalized length difference between the position vectors \mathbf{x}_i and \mathbf{p}_i of the i -th point on a trajectory of length n and the second component the normalized difference between the angles between the two position vectors (namely a normalized cosine similarity, cf. Fig. 2a).

Each component of LDA individually also has false local minima (Fig. 2b purple and cyan), but, crucially at different points. However, the global minimum is at the same location in both functions. The ratio of these two components can be weighted with the hyperparameter k_1 and k_2 (Fig. 2c) chosen for each problem. In most cases k_1 and k_2 are chosen equal to 0.5 except for the Selkov model, where k_1 equals 0.2 and k_2 equals 0.8. Exceptions to these values are indicated in the text.

This constructed loss function can be used to train the UDE successfully and is less dependent on the initial ANN parameter guess. The UDE now reconstructs the state space portrait of the original describing differential equation faithfully (see the animated state space portrait in the Supplementary video material S1 panel a).

In the simplified example, the combined LDA loss cannot fully eliminate a second erroneous local minimum using a simple weighting of each component. However, when using a neural network as the model, the parameter space has much larger dimensionality. We hypothesized that the basin of attraction for a correct minimum is much larger when using the LDA loss with appropriate weight parameters. This should result in a much higher chance to find a UDE that fits the data given random initial parameters than the MSE loss. Furthermore, we expected this effect to also depend on the bifurcation itself. In the following we tested both hypotheses extensively in computational experiments on four example systems with bifurcations.

The loss functions and different bifurcating systems. What happens if these loss functions are applied to different bifurcating systems from neuroscience, biology and chemistry? We considered the following four systems as examples of four different types of bifurcation:

- Saddle-Node bifurcation: The FitzHugh–Nagumo model^{29,30} describes resting and excited states of neurons. Specific parametrizations of the system can lead to a bistable behavior (“excitable”) with three fixed points where two are stable and one unstable (saddle). When the bifurcation parameter changes, the left stable fixed point and the saddle merge and disappear in a Saddle-Node bifurcation; the right stable fixed point remains and the system is monostable. (cf. Table 1)
- Pitchfork bifurcation: The Gardner model^{31,32} describes a genetic toggle switch in *Escherichia coli*. In one parametrization, the system is monostable with a stable fixed point but undergoes a Pitchfork bifurcation as the bifurcation parameter decreases, leading to bistable behavior with two stable fixed points and one saddle where each fixed point has its own basin of attraction. (cf. Table 1)
- Hopf bifurcation: The Selkov model¹⁵ describes oscillatory behavior in the enzyme reactions of the glycolysis process. The system shows either steady state behavior with a stable fixed point or—undergoing a Hopf bifurcation—ends up in oscillatory behavior with a stable limit cycle. (cf. Table 2)
- Period-doubling: The Rössler model^{33,34} can describe chemical reactions, but is often chosen as example for a simple period-doubling cascade. Here we focus on the transition from a period-one limit cycle to a period-two limit cycle with changing parametrization. (cf. Table 2)

We wanted to evaluate the performance of a UDE trained with the LDA loss function against a UDE trained with the MSE loss function for each model with two different parametrizations leading to the two different dynamical behaviors of the systems named above. Furthermore, we considered three different levels of normal distributed noise added to the training data to test the robustness of our results under more realistic conditions.

The universal function approximator $UA_p(x, y)$ (Equation 1) in all examples was a neural network with a single hidden layer and 16 neurons with *tanh*-activation function to accommodate a wide variety of possible non-linear functions with a limited number of ANN parameters. The initial weights of the Neural Network are chosen randomly using the Glorot initializer³⁵ with a Normal Distribution as its basis. We used the ADAM optimizer³⁶ and added weight decay to train the UDE (see Supplementary Table S1 for additional details).

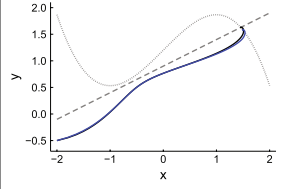
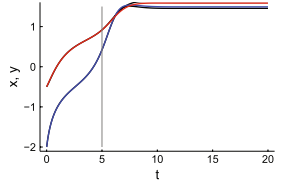
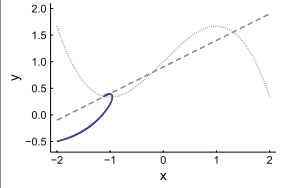
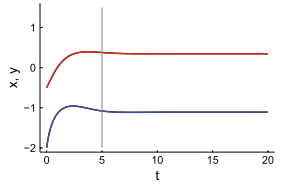
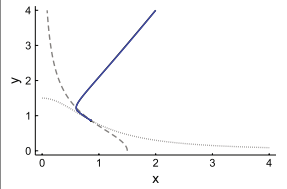
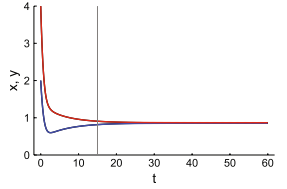
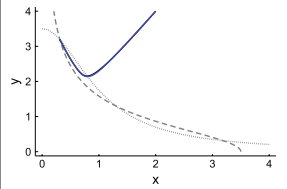
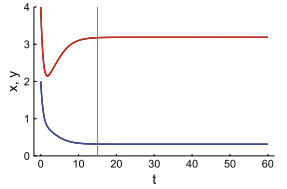
	UDE	State space	Time series	SInDy estimate	
Monostable FitzHugh–Nagumo	$\frac{dx}{dt} = x - \frac{x^3}{3} - y + 1.2$ $\frac{dy}{dt} = 1.25 \cdot (0.9 - y) + UA_p(x, y)$			$UA_p(x, y) = 0.625481 \cdot x,$ cf. missing term = $0.625 \cdot x$	Saddle-Node bifurcation
Bistable FitzHugh–Nagumo	$\frac{dx}{dt} = x - \frac{x^3}{3} - y + 1.0$ $\frac{dy}{dt} = 1.25 \cdot (0.9 - y) + UA_p(x, y)$			$UA_p(x, y) = 0.624665 \cdot x,$ cf. missing term = $0.625 \cdot x$	Saddle-Node bifurcation
Monostable Gardner	$\frac{dx}{dt} = UA_p(x, y) - x$ $\frac{dy}{dt} = \frac{1.5}{1+x^2} - y$			$UA_p(x, y) = \frac{1.499501}{1+y^2},$ cf. missing term = $\frac{1.5}{1+y^2}$	Pitchfork bifurcation
Bistable Gardner	$\frac{dx}{dt} = UA_p(x, y) - x$ $\frac{dy}{dt} = \frac{3.5}{1+x^2} - y$			$UA_p(x, y) = \frac{3.498237}{1+y^2},$ cf. missing term = $\frac{3.5}{1+y^2}$	Pitchfork bifurcation

Table 1. Overview of the models used with changing number of fixed points: FitzHugh–Nagumo model and Gardner model. First column: UDE for the systems in a general x and y notation with $UA_p(x, y)$ as machine learnable blank; second column: example of a trajectory’s longterm behavior trained with LDA (blue) in state space in comparison to the true solution’s trajectory (black). The nullcline given by the known part of the differential equation is dotted gray and the second nullcline of the true solution is gray dashed.; third column: corresponding time series of the longterm behavior of trained x (blue) and y (red) variable in comparison to the true solution (black, mainly masked by the predicted time series). The vertical gray line marks the end of the training data; fourth column: corresponding SInDy estimate of $UA_p(x, y)$ based on the approximation of the training data in comparison to the true missing term. The figures are plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

Tables 1 and 2 show representative results of the longterm prediction of the UDE model trained with LDA, where four times the length of the training data is used.

To compare estimated trajectories of the UDE trained with LDA or MSE loss objectively, we calculated the trajectory difference (TD) as the sum over the length of difference vectors over the sample points of the UDE and samples from the full differential equation:

$$TD = \frac{1}{n} \cdot \sum_{i=1}^n ||\mathbf{x}_i - \mathbf{p}_i||, \tag{4}$$

where the true trajectory is described by the position vectors of the n trajectory points x_i and the approximated position vectors of the n trajectory points p_i .

Because the initial weights w_{init} of the machine learning model can considerably determine the success of UDE training (cp. Fig. 2), we independently drew 100 different sets of initial weights w_{init} per model, model parametrization and loss function and calculated TD for each independent experiment.

In Fig. 3 we show the distribution of a normalized version of TDs we get over all these 100 trained UDEs to compare the quality of the UDE solutions across different systems, parametrizations and loss function used in one figure. The general form of these distributions remains stable across all noise levels. All results for different noise levels can be found in Supplementary Figs. S1 and S2.

Systems with a changing number of fixed points. The Pitchfork and Saddle-Node bifurcations lead to a changing number of fixed points in the Gardner and FitzHugh–Nagumo systems, respectively. A good approximation of

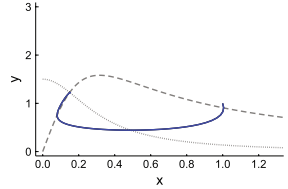
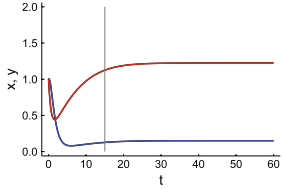
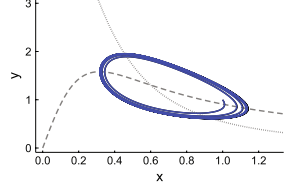
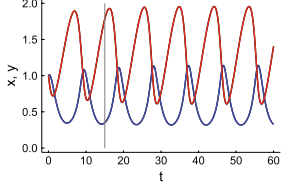
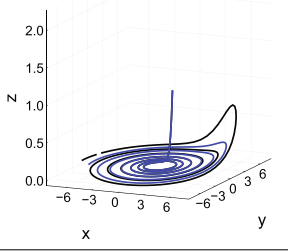
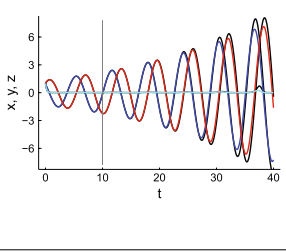
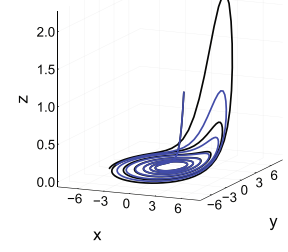
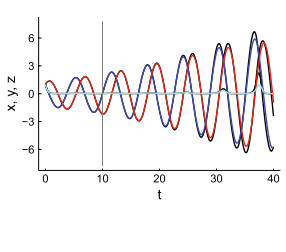
	UDE	State space	Time series	SInDy estimate	
Steady state Selkov	$\frac{dx}{dt} = -x + UA_p(x, y) + x^2 \cdot y$ $\frac{dy}{dt} = 0.15 - 0.1 \cdot y - x^2 \cdot y$			$UA_p(x, y) = 0.100014 \cdot y,$ cf. missing term = $0.1 \cdot y$	Hopf bifurcation
Oscillatory Selkov	$\frac{dx}{dt} = -x + UA_p(x, y) + x^2 \cdot y$ $\frac{dy}{dt} = 0.6 - 0.1 \cdot y - x^2 \cdot y$			$UA_p(x, y) = 0.099735 \cdot y,$ cf. missing term = $0.1 \cdot y$	Hopf bifurcation
Period-one Rössler	$\frac{dx}{dt} = -y - z$ $\frac{dy}{dt} = x + UA_p(x, y, z)$ $\frac{dz}{dt} = 0.1 + z \cdot x - 4.0 \cdot z$			$UA_p(x, y, z) = 0.100181 \cdot y,$ cf. missing term = $0.1 \cdot y$	Period-doubling
Period-two Rössler	$\frac{dx}{dt} = -y - z$ $\frac{dy}{dt} = x + UA_p(x, y, z)$ $\frac{dz}{dt} = 0.1 + z \cdot x - 6.0 \cdot z$			$UA_p(x, y, z) = 0.100183 \cdot y,$ cf. missing term = $0.1 \cdot y$	Period-doubling

Table 2. Overview of the models used with oscillatory behavior: Selkov model and Rössler model. First column: UDE for the systems in a general x and y resp. x, y and z notation with $UA_p(x, y)$ resp. $UA_p(x, y, z)$ as machine learnable blank; second column: example of a trajectory's longterm behavior trained with LDA (blue) in state space in comparison to the true solution's trajectory (black). In case of the Selkov model the nullcline given by the known part of the differential equation is dotted gray and the second nullcline of the true solution is gray dashed.; third column: corresponding time series of the longterm behavior of trained x (blue) and y (red) and in case of Rössler z (cyan) variable in comparison to the true solution (black, mainly masked by the predicted time series). The vertical gray line marks the end of the training data; fourth column: corresponding SInDy estimate of $UA_p(x, y)$ resp. $UA_p(x, y, z)$ based on the approximation of the training data in comparison to the true missing term. The figures are plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

the system's longterm behavior, indicated by small TD between the approximated and the true solution's trajectory, is in general possible with both loss functions (see Fig. 3).

However, for the FitzHugh–Nagumo model, the results using the MSE as the loss function show a higher median compared to the LDA loss and a larger distribution width in case of the monostable parametrization (cf. Fig. 3a). The reason why is that the predicted trajectories vary more around the true solution's trajectory which leads to a number of trajectories ending in alternative stable states along the given nullcline (see Supplementary Fig. S3c,d). This effect is also visible in the outliers of the TD distribution for the bistable case for the MSE loss function, which make up around 12% of all trajectories end up in alternative stable states along the nullcline given by the known part of the differential equation in the region of the second fixed point (cf. Fig. 3b and Supplementary Fig. S3b). This second stable fixed point is also the only stable fixed point in case of the monostable parametrization (cf. Tabel 1). The LDA loss function does not lead to this behavior, because the angular difference between trajectories favors the correct curvature of the trajectory compared to absolutely correct values and guides the trajectory during training towards the correct fixed point. This reflects more precisely what we hope to achieve with the learned UDE, and leads to a compact TD distribution for the longterm prediction.

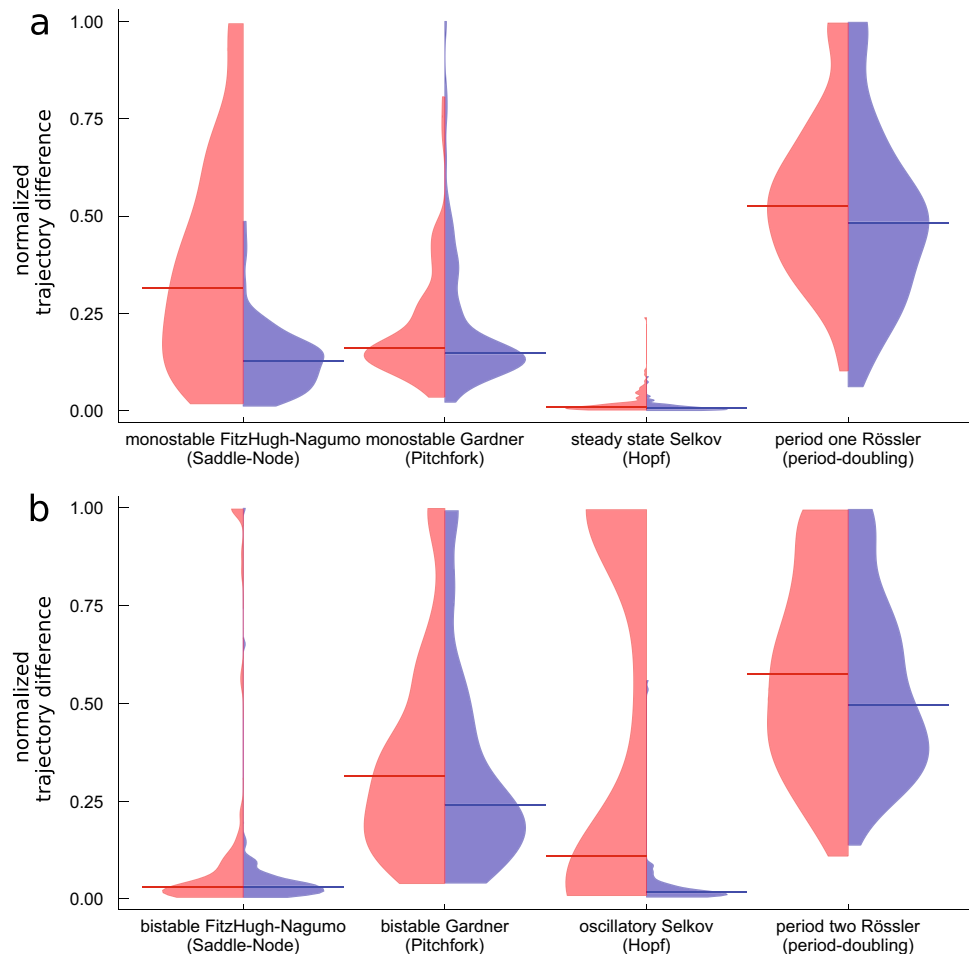


Figure 3. Distribution of the trajectory difference (TD) for the longterm prediction of the bifurcating systems over 100 neural network initializations trained using MSE loss function (red) or LDA loss function (blue). The horizontal bar indicates the median of the respective distribution. The training data contain additive normal distributed noise of noise level 0.0001. The trajectory difference is normalized by $TD = 1.0$ for FitzHugh–Nagumo, by $TD = 0.1$ for the Gardner, by $TD = 1.0$ for the Selkov and by $TD = 2.0$ for the Rössler model. All normalized values larger than one are clipped to one. The starting positions of the trajectories are chosen as for the examples in Tables 1 and 2. **(a)** Monostable, steady state and period-one parametrization **(b)** bistable, oscillatory and period-two parametrization. This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

Only around 2% of the trajectories in the bistable case end in the alternative stable state (cf. Fig. 3b and Supplementary Fig. S3a).

The same but somewhat weakened effect is observed for the bistable Gardner model (cf. Fig. 3b). Here, the results using the MSE loss function show 6–7% of the predicted longterm trajectories terminating in the alternative stable state on the given nullcline close to the fixed point the true solution’s trajectory ends up in (cf. Table 1 and Supplementary Fig. S4b). The results using the LDA loss function only show up to 5% of the trajectories ending in alternative stable states (see Supplementary Fig. S4a).

The main difference in the bistable parametrization of the FitzHugh–Nagumo compared to the bistable Gardner model is, that the alternative states in the Gardner model are not linked to either the second stable fixed point or the stable fixed point in the monostable parametrization. Rather, they only occur spuriously in the functions permitted by the UDE in its current parametrization. The underlying dynamical structure of the Gardner system separates the state space into regions that specify to which fixed point all trajectories starting in this region converge. This dynamical behavior cannot be broken during the training. The alternative stable states can likely be linked to a new parametrization of the bistable Gardner system that co-evolves during training of the UDE.

In case of the monostable parametrization of the Gardner system, both loss functions lead to approximately equally good approximations of the longterm behavior of the system (cf. Fig. 3a and Supplementary Fig. S4c,d).

Systems with oscillatory behavior. The Hopf and period-doubling bifurcation lead to changing dynamical behavior in the Selkov and Rössler systems, from non-oscillatory to oscillatory behavior or from period one to period two oscillatory behavior respectively.

For steady state behavior of the Selkov system, both loss functions lead to good approximations of the longterm behavior of the system without strong dependence on the noise level, with a median TD of around 0.006 for the LDA loss function and around 0.012 for the MSE (cf. Fig. 3a and Supplementary Fig. S5c,d). By contrast, for the case of the oscillatory parametrization of the Selkov system, the results using the MSE loss function have a bimodal TD distribution, whereas the trajectory-based LDA loss function still leads to a narrow TD distribution and consistently good approximations (cf. Fig. 3b).

This is because many initial conditions in the MSE case lead to an alternative steady state solution during optimization which is a local minimum on the loss function that cannot be left (cf. Fig. 2b and Supplementary video material S1 panel b). The bimodal nature of the TD distribution reflects this: if the initial condition is favorable, the learning process finds the correct solution, if it is not, it fails completely (see supplemental Fig. S5b). The calculated median of the TD distribution shifts depending on the number of successful approximations and complete failures (see Supplementary Fig. S2b).

The LDA loss function on the other hand benefits from the combination of angular loss and the vector length loss, which allow for detours in state space (cf. Supplementary video material S1 panel a) bringing the approximated solution during training in the region of the training data. The solutions reliably reproduce the correct oscillations of the original system in nearly all cases (see Supplementary Fig. S5b).

For the case of the Rössler system, the approximations of the training data for both parametrizations show median values of TD around 0.04 for both parametrizations for the MSE and around 0.03 for both parametrizations for the trajectory-based loss function. The distribution is compact around the median with a few outliers (see Supplementary Fig. S6). By contrast, the longterm prediction of both used loss functions show median values of TD around 1.04 (period-one parametrization) resp. 1.22 (period-two parametrization) for MSE and around 0.95 (period-one parametrization) resp. 1.10 (period-two parametrization) for the LDA loss function (cf. Fig. 3a,b). Only a few runs yield good approximations. The reason why is that the training data do not include the typical excursions of the Rössler system in z-direction in state space which are included in the longterm prediction (cf. Table 2). These excursions are not approximated well by either of the trained networks. Longer training data could improve these results. Nevertheless, the accurate approximation of the short term training data enables a proper reconstruction of the algebraic form of the missing term as in Table 2.

The loss functions around the bifurcation. In the previous section we observed very different distributions for the trajectory difference depending on the chosen loss function and the parametrizations of the systems. To investigate if the distributions show a gradual change from one type to the other across the bifurcation, we performed 50 training runs for the FitzHugh–Nagumo and the Selkov model each with 14 different parametrization chosen from an interval nesting of the bifurcation parameter around the bifurcation. The bifurcation parameter for the case of FitzHugh–Nagumo is the 4th term in the first equation in Table 1 and in case of Selkov the first term in the second equation in Table 2. The FitzHugh–Nagumo and the Selkov model were chosen because they show the most different distributions of the trajectory difference TD for the two different dynamical behaviors (cf. Fig. 3). The results are presented in Fig. 4.

Firstly, we see a change of the shape of the MSE loss distribution as soon as the curvature of the trajectory increases (FitzHugh–Nagumo model Fig. 4a) or as soon as strongly damped oscillations start (Selkov model Fig. 4b between the light gray and the gray bar). The median of the LDA TD distribution is only sensitive to this in a narrow band of parameters shortly after the bifurcation in the FitzHugh–Nagumo model (cf. inset in Fig. 4a). Thus, the distributions reflect the bifurcation, too. As the curvature of the trajectory changes drastically across the bifurcation for both systems, the difference in the quality of the approximation using either loss function is markedly different. The reason for this behavior is that MSE only optimizes the difference in each variable in state space independently. This approach misses differences in the curvature of the training data trajectory and the current approximated solution. Thus, the MSE is often not suited to approximate a proper solution for a randomly chosen first guess of the approximated solution that belongs to a different dynamical regime than the training data and must be transferred to another dynamical regime during training by temporarily changing the curvature of the approximated solution drastically.

In contrast, the LDA loss function indirectly contains information about the curvature difference in its angle part of the loss function at a specific position in state space (length difference part of the loss function). Therefore, the LDA loss can capture changes in curvature earlier and permits the randomly chosen first guess UDE in the wrong dynamical regime to change to the dynamical regime of the training data more reliably. On the other hand considering small changes in the direction of a trajectory can immediately cause bad performance of LDA for specific noisy trajectory passing very close to a saddle. These small changes in the direction of the trajectory can lead to a switch of the dynamical behavior and a broader TD distribution (cf. Fig. 4a parameter 1.139 and 1.143).

In sum, there is an abrupt change from one type of distribution to another around the bifurcation reflecting the ability of the loss functions to reliably capture changes in curvature or not.

The loss functions and different starting points of trajectories. To investigate the impact of the starting position of the trajectories in state space on the performance of a UDE trained with the LDA loss function against a UDE trained with the MSE loss function, we again trained 50 UDEs for the FitzHugh–Nagumo and the Selkov model in the two parametrizations used in Tables 1 and 2 respectively. We chose starting points with very different dynamical behavior of the trajectories, namely slow and fast dynamics in state space as well as weak and strong curvatures of the trajectories. The results are presented in Fig. 5.

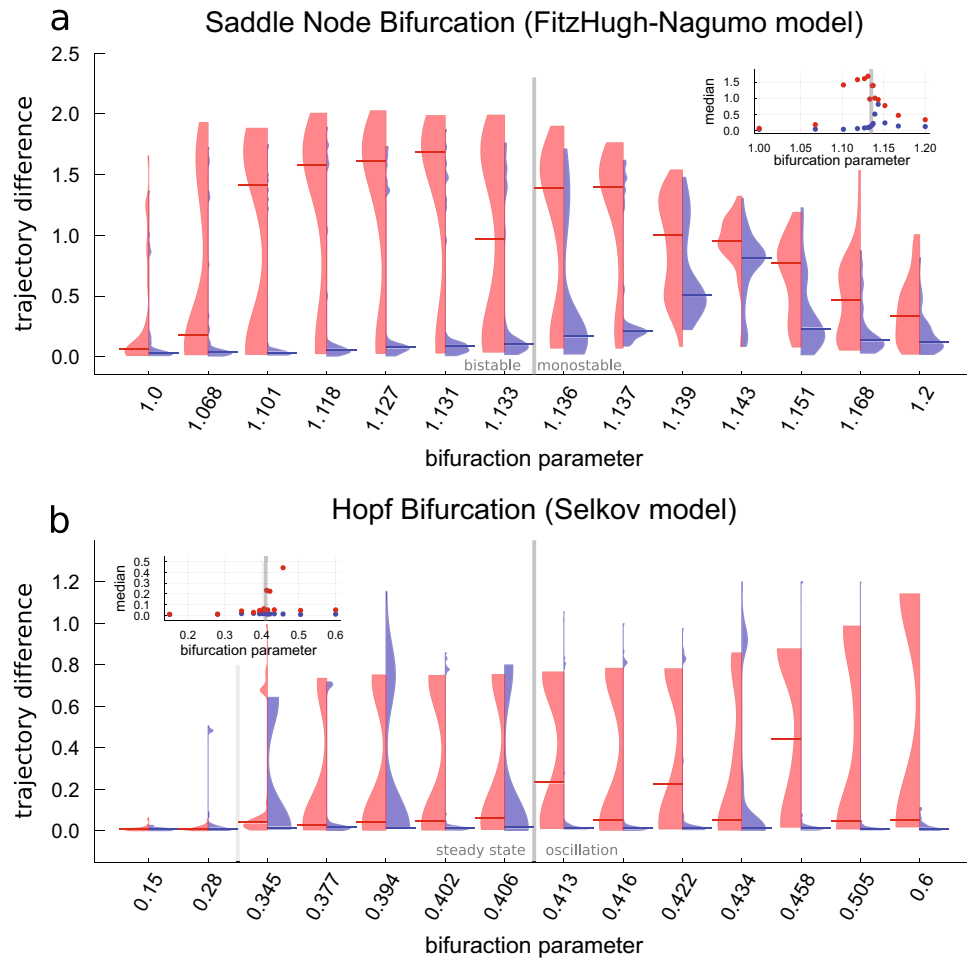


Figure 4. Distribution of the trajectory difference (TD) for the longterm prediction of the bifurcating systems over 50 neural network initializations trained using the MSE loss function (red) or LDA loss function (blue) for FitzHugh–Nagumo (a) and Selkov model (b) around the bifurcation (gray bar). The light gray bar in case of the Selkov model indicates the onset of strongly damped oscillations. The horizontal bar indicates the median of the respective distribution. The median of the distribution is plotted in the inset. The training data contain additive normal distributed noise with the noise level set to 0.0001. The trajectory difference larger than 1.2 are clipped to 1.2 in case of the Selkov model (b). The bifurcation parameter in case of FitzHugh–Nagumo is the 4th term in the first equation in Table 1 and in case of Selkov the first term in the second equation in Table 2. The trajectories start at $(-2.0, -0.25)$ in case of FitzHugh–Nagumo and at $(1.0, 1.0)$ in case of Selkov. This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

For both models we observe a strong dependence of the success of the training on the starting position in state space when using MSE. Less successful approximations are often linked with strong curvatures or oscillatory behavior. In case of the LDA loss function for the monostable FitzHugh–Nagumo model we observe less successful training for starting position B, C and E (cf. Fig. 5b). B and C share the common feature that they pass very slowly close to the saddle and the training data trajectory ends in this region, thus small changes of the trajectory due to noise can lead to a broadening of the distribution of the longterm prediction. Starting position E corresponds to a trajectory with a fast dynamics in case of the monostable parametrization and an extremely curved trajectory in case of the bistable parametrization. While we observe a wider distribution of trajectory differences in the monostable case, the LDA loss still captures the correct dynamical regime in case of the bistable parametrization and does not converge to a different fixed point as is possible for the MSE (cf. Fig. 5c).

When learning the Selkov model, the MSE shows a bimodal distribution for the oscillatory parametrization as expected, but this is also the case for starting point E in the steady state parametrization (cf. Fig. 5e). E is very close to second nullcline in state space which is unknown for the UDE. One possible explanation is that during training two different solution classes (steady state and oscillatory) are realized. We even observe a bimodal distribution in case of the LDA for the steady state and the oscillatory parametrization if we apply the standard values for the factors k_1 and k_2 used in the experiments above. If we adjust k_1 to 0.1 and k_2 to 0.9, the bimodal distribution vanishes and successful training is possible (cf. E in Fig. 5e,f). The same effect occurs for D in case

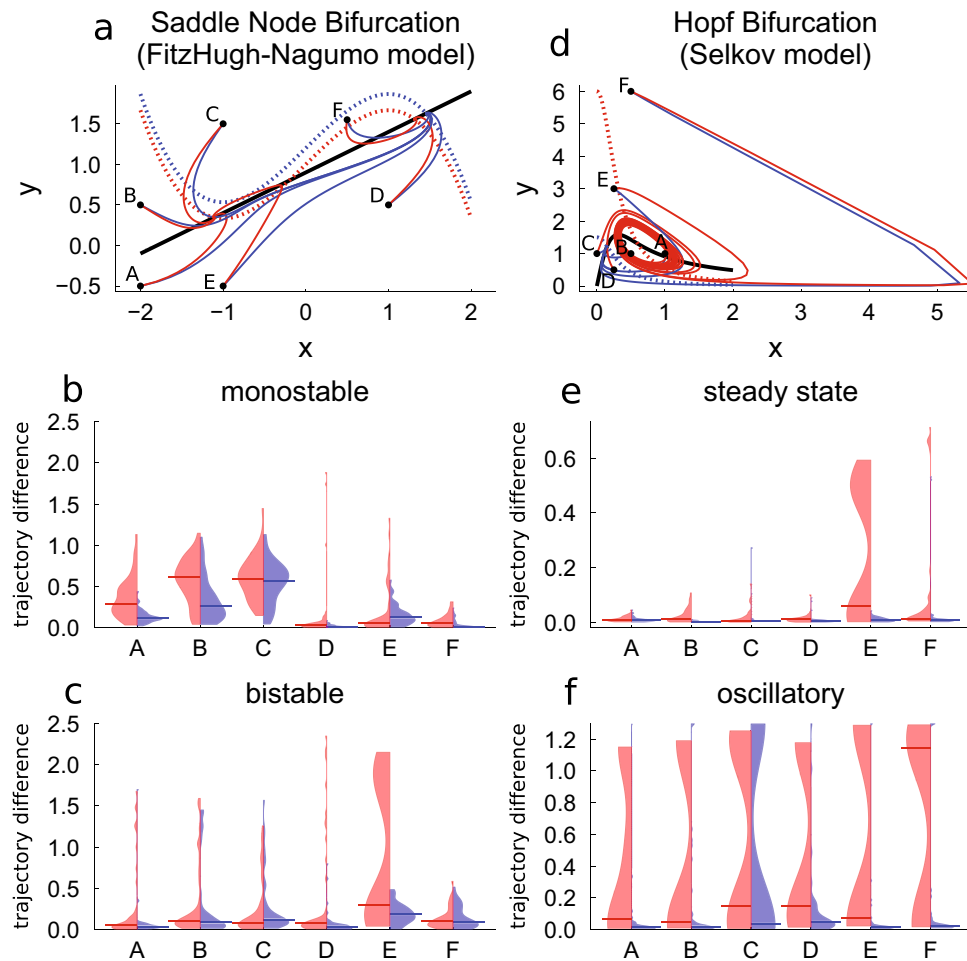


Figure 5. Distribution of the trajectory difference (TD) for the longterm prediction of the FitzHugh–Nagumo model (b, c) and the Selkov model (e, f) over 50 neural network initializations trained using the MSE loss function (red) or LDA loss function (blue) for starting position A to F of the trajectories (a, d) using two different parametrizations. The horizontal bar indicates the median of the respective distribution. The training data contain additive normal distributed noise with the noise level set to 0.0001. The red trajectories correspond to the bistable case (a) resp. oscillatory case (d), the blue one to the monostable case (a) resp. steady state (d). The black curve is the nullcline given by the known part of the differential equation, the dashed red (blue) curve is the second nullcline corresponding to the bistable resp. oscillatory state (to the monostable resp. steady state) in (a) resp. (b). This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

of the oscillatory parametrization (cf. Fig. 5f), whereas a change of k_1 and k_2 does not impact the results for C (cf. Fig. 5f).

To sum up, the starting position of the trajectory and its resulting properties can impact the success of the training. This is especially the case for the MSE, where there are no further hyperparameters to tune the solution.

SInDy's answer: algebraic form of the missing term. Finally, to get an idea of the algebraic form of the missing term, an approach for sparse identification of nonlinear dynamical systems (SInDy)¹² by Brunton et al. was applied to examples trained with the LDA loss function. This approach was chosen to show by way of example that a successful trained UDE system can be translated into an algebraic form. The examples are chosen from the statistical analysis with the lowest noise level such that the trajectory difference between the approximated solution of the training data and the training data itself is smallest. We investigated one example for each parametrization and each model. To improve the performance of SInDy, outliers in the beginning and the end of the approximated guess of the dynamics of the missing term are removed before SInDy is applied. The optimizer used by SInDy is STRRidge¹². The basis functions from which SInDy can select its guess are polynomials up to order of 5, except for the case of the Gardner model where polynomials up to order of 3 are combined with terms of the form $\frac{1}{1+z^n}$ with z being either x or y and $n = 1, 2, 3$. This exception was chosen to identify the term more clearly and not hidden it in a polynomial expansion of the function.

The results are shown in Tables 1 and 2. All in all, the algebraic form of the missing terms as well as the system parameters are approximated well.

The approach is not limited to the examples shown. It is possible to learn other terms or several terms at the same time. A crucial precondition for a successful reconstruction of the missing term with SInDy is an accurate approximation of the dynamics of the missing term with the UDE approach, which is directly linked with the successful reconstruction of the dynamical behavior of the training data. Thus, the success of the SInDy approach is directly linked with the accuracy of the results of the UDE approach. Furthermore, an accurate reconstruction of the missing term with SInDy based only on the approximation of the training data might give better results for the longterm prediction of the dynamics than the UDE, because the sparsity criterion of SInDy suppresses numerical fluctuations present in the UDE that might interfere with the longterm prediction.

Discussion and conclusion

In summary, the LDA loss function leads to more stable and accurate approximation of the longterm dynamical behavior in systems with a changing number of fixed points or a change in oscillatory behavior due to a bifurcation.

Often, trained networks using MSE as loss function lead to a solution which is only slightly worse than the networks using the LDA loss function, but there is a risk of complete failure of the approximation depending on the network initialization. Local minima in the MSE loss function due to different dynamical regimes of the UDE can represent completely different dynamical solutions, which gradient-based learning cannot recover from by adapting the gradient-descent procedure with, for example, momentum³⁷ or adaptive learning rates³⁶.

We used the LDA loss, which optimizes two components, the length and angle of state space vectors, with different local minima. A weighted combination of both components allowed us to estimate the dynamics of bifurcating systems more accurately, and avoided the catastrophic scenario of local minima leading to different dynamic regimes. While some randomly chosen initial parameters for the neural network can still lead to a bad fit of the UDE, we found that in examples where the MSE loss often finds local minima in the wrong dynamic regime, the basin of attraction of the optimization process was indeed much larger when we used the LDA loss instead. In some cases the choice of the factors weighting the two components can improve the results of the training as shown in case of the study of different starting points. We demonstrated that in applications, the loss function has to be chosen with great care based on the dynamics of the measured trajectories. This is because there is no gradual change from the possibility of successful training to failure across the bifurcation.

A more general open questions of UDE is the impact of the training data length and its temporal resolution on the successful approximation of the system. A preliminary survey of this question for examples of the LDA loss does not indicate a clear effect for the temporal resolution (cf. supplementary Fig. S7) but as expected the trajectory differences between the approximated and the true solution trajectory decreases with increasing training data length (cf. supplementary Fig. S8). A more comprehensive study should be the content of further research, because in any real data set of measured time series the length and temporal resolution is fixed.

In real-world applications, physics-informed neural network approaches have successfully been applied to, among others, fluid dynamical problems^{38,39}. SInDy has also shown good results in complex data-driven settings^{40–42}. Universal Differential Equations¹¹ elegantly bring together two important aspects of these approaches: the advantages of knowledge about the underlying physics in the form of a differential equation, and the statistical estimation of dynamics via machine learning. SInDy¹² can be used to open the black box of machine learning and reconstruct algebraic terms to complete a differential equation. This approach to knowledge discovery is powerful. It implies that some interactions in the system of interest are not yet discovered, which is particularly true for—for example—ecosystems that cannot completely be examined in the lab or for systems for which the underlying governing equation is unknown. In such systems tipping points and potentially unknown bifurcations are of particular interest.

We have demonstrated, that UDEs can be used to estimate and reconstruct missing terms even in bifurcating systems. However, all examples shown here are simple compared to real-world applications. But even at this basic level we only have a rough idea of the interaction of the learning process and the bifurcation properties of the system. The interaction of both the dynamics of gradient-descent optimizing a loss function and the dynamics of the system of interest itself is highly fascinating and not yet understood. At this stage more fundamental research is necessary.

Particular care has to be taken in how the loss function is constructed for the optimization process, because the optimization process itself can lead to bifurcations. Nonetheless, our results suggest that dynamics in real-world, not yet fully described systems can be discovered using the LDA loss function.

Data availability

All experiments have been implemented in the Julia programming language¹³ based on packages developed particularly to fit UDEs with neural networks (<https://github.com/SciML/DiffEqFlux.jl>) and SInDy (<https://github.com/SciML/DataDrivenDiffEq.jl>). The code to produce all data presented in the paper is available publically at <https://github.com/pnieters/GeneralizedDynamicsFromData>.

Code availability

Accession codes The code to produce all data presented in the paper is available publically at <https://github.com/pnieters/GeneralizedDynamicsFromData>.

Received: 28 May 2021; Accepted: 21 September 2021

Published online: 14 October 2021

References

1. Steffen, W. *et al.* Trajectories of the Earth system in the anthropocene. *Proc. Natl. Acad. Sci.* **115**, 8252–8259. <https://doi.org/10.1073/pnas.1810141115> (2018).
2. Simonnet, E., Dijkstra, H. A. & Ghil, M. Bifurcation analysis of ocean, atmosphere, and climate models. In *Handbook of Numerical Analysis*, 187–229. [https://doi.org/10.1016/s1570-8659\(08\)00203-2](https://doi.org/10.1016/s1570-8659(08)00203-2) (Elsevier, 2009).
3. van Nes, E. H., Rip, W. J. & Scheffer, M. A theory for cyclic shifts between alternative states in shallow lakes. *Ecosystems* **10**, 17–28. <https://doi.org/10.1007/s10021-006-0176-0> (2007).
4. Stommel, H. Thermohaline convection with two stable regimes of flow. *Tellus* **13**, 224–230. <https://doi.org/10.1111/j.2153-3490.1961.tb00079.x> (1961).
5. Feudel, U., Pisarchik, A. N. & Showalter, K. Multistability and tipping: From mathematics and physics to climate and brain—Mini-review and preface to the focus issue. *Chaos Interdiscip. J. Nonlinear Sci.* **28**, 033501. <https://doi.org/10.1063/1.5027718> (2018).
6. Hadjighasem, A., Farazmand, M., Blazeovski, D., Froyland, G. & Haller, G. A critical comparison of Lagrangian methods for coherent structure detection. *Chaos Interdiscip. J. Nonlinear Sci.* **27**, 053104. <https://doi.org/10.1063/1.4982720> (2017).
7. Mancho, A. M., Small, D. & Wiggins, S. A tutorial on dynamical systems concepts applied to Lagrangian transport in oceanic flows defined as finite time data sets: Theoretical and computational issues. *Phys. Rep.* **437**, 55–124. <https://doi.org/10.1016/j.physrep.2006.09.005> (2006).
8. Boers, N., Ghil, M. & Rousseau, D.-D. Ocean circulation, ice shelf, and sea ice interactions explain Dansgaard–Oeschger cycles. *Proc. Natl. Acad. Sci.* **115**, E11005–E11014. <https://doi.org/10.1073/pnas.1802573115> (2018).
9. Shimoda, Y. & Arhonditsis, G. B. Phytoplankton functional type modelling: Running before we can walk? A critical evaluation of the current state of knowledge. *Ecol. Model.* **320**, 29–43. <https://doi.org/10.1016/j.ecolmodel.2015.08.029> (2016).
10. Edwards, A. M. & Brindley, J. Oscillatory behaviour in a three-component plankton population model. *Dyn. Stab. Syst.* **11**, 347–370. <https://doi.org/10.1080/02681119608806231> (1996).
11. Rackauckas, C. *et al.* Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385v1*, v2, v3 (2020).
12. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.* **113**, 3932–3937. <https://doi.org/10.1073/pnas.1517384113> (2016).
13. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Rev.* **59**, 65–98. <https://doi.org/10.1137/141000671> (2017).
14. Innes, M. Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951* (2018).
15. Sel'kov, E. E. Self-oscillations in glycolysis. 1. A simple kinetic model. *Eur. J. Biochem.* **4**, 79–86. <https://doi.org/10.1111/j.1432-1033.1968.tb00175.x> (1968).
16. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444. <https://doi.org/10.1038/nature14539> (2015).
17. Lagaris, I. E., Likas, A. & Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**, 987–1000. <https://doi.org/10.1109/72.712178> (1998).
18. Chen, R. T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018). (Accessed 26 April 2021).
19. Raissi, M., Perdikaris, P. & Karniadakis, G. E. 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. <https://doi.org/10.1016/j.jcp.2018.10.045> (2019).
20. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236* (2018). (Accessed 26 April 2021).
21. Psychgiou, D. C. & Ungar, L. H. A hybrid neural network-first principles approach to process modeling. *AIChE J.* **38**, 1499–1511. <https://doi.org/10.1002/aic.690381003> (1992).
22. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2**, 303–314. <https://doi.org/10.1007/BF02551274> (1989).
23. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**, 251–257. [https://doi.org/10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T) (1991).
24. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).
25. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B (Methodol.)* **58**, 267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x> (1996).
26. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep Learning* Vol. 1 (MIT Press Cambridge, 2016).
27. Christoffersen, P. & Jacobs, K. The importance of the loss function in option valuation. *J. Finan. Econ.* **72**, 291–318. <https://doi.org/10.1016/j.jfineco.2003.02.001> (2004).
28. Xiao, T. & Frank, M. Using neural networks to accelerate the solution of the Boltzmann equation. *arXiv preprint arXiv:2010.13649* (2020). (Accessed 26 April 2021).
29. FitzHugh, R. Mathematical models of threshold phenomena in the nerve membrane. *Bull. Math. Biophys.* **17**, 257–278. <https://doi.org/10.1007/bf0247753> (1955).
30. FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophys. J.* **1**, 445–466. [https://doi.org/10.1016/s0006-3495\(61\)86902-6](https://doi.org/10.1016/s0006-3495(61)86902-6) (1961).
31. Gardner, T. S., Cantor, C. R. & Collins, J. J. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**, 339–342. <https://doi.org/10.1038/35002131> (2000).
32. Bose, I. & Ghosh, S. Bifurcation and criticality. *J. Stat. Mech. Theory Exp.* **2019**, 043403. <https://doi.org/10.1088/1742-5468/ab11d8> (2019).
33. Rössler, O. E. An equation for continuous chaos. *Phys. Lett. A* **57**, 397–398. [https://doi.org/10.1016/0375-9601\(76\)90101-8](https://doi.org/10.1016/0375-9601(76)90101-8) (1976).
34. Rössler, O. E. Chaotic behavior in simple reaction systems. *Zeitschrift für Naturforschung A* **31**, 259–264. <https://doi.org/10.1515/zna-1976-3-408> (1976).
35. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (JMLR Workshop and Conference Proceedings, 2010).
36. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015).
37. Nesterov, Y. E. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr* **269**, 543–547 (1983).
38. Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030. <https://doi.org/10.1126/science.aaw4741> (2020).
39. Raissi, M. & Karniadakis, G. E. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.* **357**, 125–141. <https://doi.org/10.1016/j.jcp.2017.11.039> (2018).
40. Champion, K., Zheng, P., Aravkin, A. Y., Brunton, S. L. & Kutz, J. N. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access* **8**, 169259–169271. <https://doi.org/10.1109/ACCESS.2020.3023625> (2020).
41. Champion, K., Lusch, B., Kutz, J. N. & Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proc. Natl. Acad. Sci.* **116**, 22445–22451. <https://doi.org/10.1073/pnas.1906995116> (2019).

42. Rudy, S. H., Brunton, S. L., Proctor, J. L. & Kutz, J. N. Data-driven discovery of partial differential equations. *Sci. Adv.* **3**, e1602614. <https://doi.org/10.1126/sciadv.1602614> (2017).

Acknowledgements

R.V.-K. and P.N. thank Georg Schröter for intensive discussions.

Author contributions

R.V.-K. developed the trajectory-based loss function. P.N. and R.V.-K. designed the numerical experiments. P.N. implemented the experiments. Overall supervision was done by G.P. All authors contributed in preparing this manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-021-99609-x>.

Correspondence and requests for materials should be addressed to R.V.-K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2021