Engineering Applications of Artificial Intelligence 130 (2024) 107611

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Enabling global interpolation, derivative estimation and model identification from sparse multi-experiment time series data via neural ODEs

William Bradley, Ron Volkovinsky, Fani Boukouvala *

School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA, USA

ARTICLE INFO

Keywords:
Neural ODEs
Derivative estimation
Sparsity
Time-series data

ABSTRACT

Estimation of the rate of change of a system's states from state measurements is a key step in several system analysis and model-building workflows. While numerous interpolating models exist for inferring derivatives of time series when data is disturbed by noise or stochasticity, general-purpose methods for estimating derivatives from sparse time series datasets are largely lacking. A notable weakness of current methods, which are largely local, is their inability to globally fit data arising from non-identical initial conditions (i.e., multiple experiments or trajectories). In this contribution, Neural ODEs (NODEs) are demonstrated to close this gap. Through a series of benchmarks, we show that because of the differential formulation of NODEs, these data smoothers can infer system dynamics of sparse data, even when accurate interpolation by algebraic methods is unlikely or fundamentally impossible. Through the presented case studies for derivative estimation and model identification, we discuss the advantages and limitations of our proposed workflow and identify cases where NODEs lead to statistically significant improvements. In summary, the proposed method is shown to be advantageous when inferring derivatives from sparse data stratified across multiple experiments and serves as a foundation for further model development and analysis methods (e.g., parameter estimation, model identification, sensitivity analysis).

1. Introduction

If truly can it be said that "form follows function" in architecture as well as the biological and materials sciences, in the statistical and mathematical sciences an equally valid paradigm may be that *formulation* follows function. This latter statement is made here to mean that, in choosing the formulation of a mathematical model, foremost among considerations ought to be its functional role. After all, a model's function will ultimately be constrained by its formulation. This principle can be instructive, for example, in surrogate modeling—the task of replacing a complex simulation or experiment with an empirical metamodel. To build a surrogate model whose function includes maximal generalizability, conventional wisdom argues that the surrogate with the simplest formulation that still captures the system response ought to be used (Forrester et al., 2009). Moreover, if the function of the surrogate model is model-based optimization, then the formulation of the surrogate model should have the property of being first and second-order differentiable. As another example, when modeling a system response, if the function of the model is to capture a dynamic (i.e., transient) response, then a formulation based on differential equations may be more appropriate. In contrast, if the situation called for modeling a non-transient response, a formulation of purely algebraic equations will often suffice.

This truism has potential crossover applications for the goal of estimating derivatives of dynamic systems. Estimating derivatives, or the rate of change of a system's states, is of interest in many system analysis workflows including sensitivity analysis, uncertainty quantification, and parameter estimation of dynamic models, spanning problems across nearly every scientific and engineering discipline. However, accurate estimation of derivatives from data is a challenging task due to the need to account for noise, sparsity and nonlinearity in real-world datasets. To meet this challenge, numerous methods for interpolating data and estimating derivatives have been proposed. An early contribution is the work of Varah (1982) who used splines as the means for interpolating data for the purpose of derivative estimation and ultimately parameter estimation of differential equations. Similar studies have used Support Vector Machines (Mehrkanoon et al., 2014), Neural Networks (Dua,

* Corresponding author.
E-mail address: fani.boukouvala@chbe.gatech.edu (F. Boukouvala).

https://doi.org/10.1016/j.engappai.2023.107611
Received 3 June 2022; Received in revised form 2 November 2023; Accepted 23 November 2023
Available online 17 December 2023

Abbreviations

AICc	Akaike Information Criterion (Corrected)
a-NN	algebraic Neural Network
BIC	Bayesian Information Criterion
DE	Differential Equation
FD	Finite Difference
FHN	Fitz-Hugh Nagumo (model)
GP	Gaussian Process
LV	Lotka-Volterra (model)
ML	Machine Learning
NN	Neural Network
MAE	Mean Absolute Error
MSE	Mean Squared Error
MWUT	Mann-Whitney *U* Test
NODE	Neural Ordinary Differential Equation
ODE	Ordinary Differential Equation
SINDy	Sparse Identification of Nonlinear Dynamics
SIR	Susceptible, Infected, Resistant (model)
SSE	Sum of Squared Errors
VDP	Van der Pol (model)

2011), and Gaussian Processes (Liu et al., 2009; Swain et al., 2016) to regress dynamically-sampled data and estimate derivatives. However, a notable link between all these algebraic formulations is their dependence on time as an input to the data-interpolating model. The key motivation of this contribution is the hypothesis that a differential formulation is a better interpolator of a differential response, especially in cases of sparse data.

In a recent publication, we proposed Neural Ordinary Differential Equations (NODEs) as the data-driven means for interpolating state data for the purpose of derivative estimation (Bradley et al., 2021). A NODE can be viewed as the differential form of a standard algebraic Neural Network (a-NN) and differs from standard a-NNs primarily in that it predicts the instantaneous change in a system state rather than the state itself (Chen et al., 2018). To motivate its use over other surrogate models, the NODE's ability to infer state derivatives was compared in our work with that of an a-NN, with the former clearly offering superior results. However, there exist numerous methods for automated interpolation of data which could be considered, many of which are more frequently used, and potentially more accurate than a-NNs at derivative estimation.

The most commonly used interpolating methods for derivative estimation include filter techniques (Butterworth, 1930; Schafer, 2011; Savitzky et al., 1964; Aravkin et al., 2017; Kalman, 1960), optimization methods (Chartrand, 2017; Rudin et al., 1992), local regression (Belytschko et al., 1996; Harrell, 2015) or kernel smoothing (Gardner, 2006), moving-average methods (Hyndman, 2010), splines (Boor, 1978), and numerical differentiation. Formulation of some of these methods varies based on whether the interpolating model should pass through data precisely or regress the data to recover a signal from noisy measurements (e.g., standard splines vs. smoothing splines). These methods have been reviewed for interpolating dynamic signals (Härdle et al., 1997; Alexandrov et al., 2012; Lepot et al., 2017), sometimes as a means for short-term forecasting (Rahardja, 2020; Atluri et al., 2018). Naturally, these methods may be combined, for example, by using numerical differentiation to infer the derivatives of curves fitted from filtering techniques as done in (van Breugel et al., 2020). For many of these techniques, there already exists highly automated, computationally-efficient, user-friendly software that facilitates their implementation.

The goal of this work is to motivate the use of NODEs by making a case for their superior accuracy over algebraic interpolating techniques. Although comparing to all methods listed above is outside the scope of this work, we select two of the most popular and tested methods, namely B-splines and Numerical Differentiation as the baseline approaches. Based on results obtained in our previous work, we form the hypothesis that NODEs will be more accurate than an interpolating method based on splines or finite differences (FDs), if the data is sparsely sampled across multiple experiments. This hypothesis is motivated by a simple fact, namely, that by being formulated as a differential rather than algebraic model, a NODE's inputs are different than other interpolating models. To clarify why this may be important, consider the formula for a B-spline (Eqs. (1)–(3)), a frequently used interpolating method due to its high accuracy (Aguilera et al., 2013; Eilers et al., 2010), flexibility (Sun et al., 2017) and ease of implementation (Perperoglou et al., 2019).

$$S(t) = \sum_{i=0}^{n-1} c_i B_{i,k}(t) \tag{1}$$

$$B_{i,0}(t) = \begin{cases} 1, t_i < t < t_{i+1} \\ 0, otherwise \end{cases} \tag{2}$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(t) \tag{3}$$

A B-spline $S(t)$ of order n is a collection of piecewise polynomial functions of degree k=n-1. The polynomials are connected at N knot points t_i, where i = [1,2, …N], and are defined by the coefficients c_i multiplied by the basis functions $B_{i,k}(t)$ defined by Equation (3), also known as the Cox-de Boor recursion formula (Boor, 1978). The main takeaway from the above equations is that the B-spline (like other interpolating methods) is a function of the independent variable t over which the states are changing (i.e., usually time). As a consequence of this, when interpolating dynamic data the spline offers a local approximation only. A different set of spline functions is required not only for each state variable but also for each distinct state 'trajectory' for which data is available. Generally, datasets with multiple distinct trajectories arise when the states of a dynamic system evolve from a unique set of initial or boundary conditions. These datasets are less common in observational studies (e.g., the observed rise in housing prices) but are nearly unavoidable in an experimental setting (e.g., tracking effective yield of a chemical reaction in response to different reactant feed rates). Indeed, datasets with multiple state trajectories are created whenever multiple experiments are performed on the same system.

In contrast to splines and all other time-dependent (i.e., algebraic, or local) interpolating methods, a NODE's formulation is not an explicit function of time, but rather the states only (Eq. (4)).

$$\frac{dx_k}{dt} = NN(x_k, \boldsymbol{w}) \tag{4}$$

In this work, to be considered a NODE, a Neural Network model $NN(\boldsymbol{x_k}, \boldsymbol{w})$, which is a function of parameters $\boldsymbol{w}$ and K states $\boldsymbol{x_k}$, where k = [1, 2, …K], predicts the rate of change of the states (i.e., the state derivatives). Further details on the NODE's architecture can be found in the Appendix. In theory, Equation (4) could be changed to include time as an input. This might be advantageous if the NODE models a homogenous differential equation (DE) system. However, there are obvious advantages of preserving the NODE form of Equation (4). This is because, unlike other interpolating methods, a single NODE model can be used to globally interpolate multi-variate data even if the data is spread across multiple experimental runs. This global interpolation requires the following two assumptions, namely that the dynamics are continuous or can be made continuous (Chen et al., 2021) and are properly scaled. The potential advantages of this fact come critically into play when experiments sample data only sparsely. Data from one experiment can influence the NODE model predictions for a different experiment where no data is available. The idea of developing global interpolators for multiple sparse experiments has been considered in the

past in the form of Bayesian or generalized linear models (Deboeck, 2020; Boker et al., 2010), however the methods of those studies either don't generalize well to irregularly-sampled, multi-variate systems or require too much user intervention to be amenable to automation.

The meaning of sparsity can vary among disciplines and even between applications. What may be a sparse dataset for one application may be excessive for another. Though not considered in this work, when data becomes excessive or high-dimensional, it may be desirable to create a sparse representation of the dataset to remove correlated variables as done in (Chen et al., 2022; Peng et al., 2022). In this work, sparsity is instead used to refer to datasets that have fewer samples than would be needed for local approximation methods. Importantly, this does not mean that the total number of data samples is few. When data from multiple experiments are available, the collated dataset may be large, though data from a single experiment may be few. To make the discussion more concrete, a visual example of a densely sampled system is juxtaposed to datasets representing different modes of sparsity in Fig. 1 for an illustrative system with 3 states.

Reasons for the apparent sparsity depicted in Fig. 1 can vary. For example, missing data could be caused by faulty equipment, outlier removal, limits of sensing techniques, and more. Lower frequency measurements can be due to the sensitivity of the experiment to observation (i.e., sampling changes the experimental conditions or quenches the experiment) or a limitation of the analysis instruments to examine samples at higher volumes or at shorter temporal frequencies. Unobserved states occur when available equipment is unable to track important variables, or the variable does not appear in high enough quantities to be observable. Of course, depending on the sampling, the dataset may be described by multiple Modes outlined in Fig. 1, and this discussion is not intended to define hard boundaries between each sparsity scenario. Several causes of sparsity in chemical process data and machine learning (ML) algorithms to tackle them have recently been reviewed by Thebelt et al. (Thebelt et al., 2022). Among the cases in Fig. 1, only Modes 1–3 are considered in this work. For Mode 4, no state measures are available for the unobserved state and any interpolating method, including the one presented in this work, will fail.

In the work that follows, NODEs will be tested for inferring derivatives under sparse conditions not previously explored in other works. Differing modes of sparsity will be explored with the aims being to demonstrate that 1) NODEs can outperform local interpolating methods at inferring system derivatives and 2) their derivatives are sufficiently accurate for inference tasks such as parameter estimation and model identification. Achieving this second aim can be powerful since it would enable computer-automated model-building for systems where previously this task was computationally intractable. Other works have examined the interpolating potential of NODEs (Chen et al., 2018), but not for inferring derivatives. Moreover, an analysis of a NODE's ability to learn dynamic responses when data is thinly spread across multiple experimental runs has not been systematically studied in the literature so far. Finally, a comparison of NODEs with popular data-driven interpolation methods is lacking.

The results of this study can be broken into two parts. In the first part, NODEs are compared with other data-driven interpolation methods for inferring derivatives on a single benchmark system. The qualitative results therein offer heuristics that may guide a modeler's choice on when to apply NODEs for derivative estimation. In the second part, the conclusions from the first part are tested for their ability to generalize to other systems and data sampling scenarios. In addition, statistical tests are employed to justify the length of NODE integration during training. Finally, the benefits of the superior accuracy of NODE interpolation are demonstrated by solving a problem in model identification.

2. Methods

2.1. Algebraic interpolators

For the first set of studies, NODEs are compared with two other interpolating methods for inferring derivatives. The competing interpolating techniques chosen in this work are finite differences and B-splines, which are the basis of many other local interpolating methods and are thus expected to perform similarly to many state-of-the-art methods for interpolation tasks. For finite difference (FD) approximation, the implementation made available through PySINDy is used (Kaptanoglu et al., 2022). FD approximations ranging from 2nd to 6th

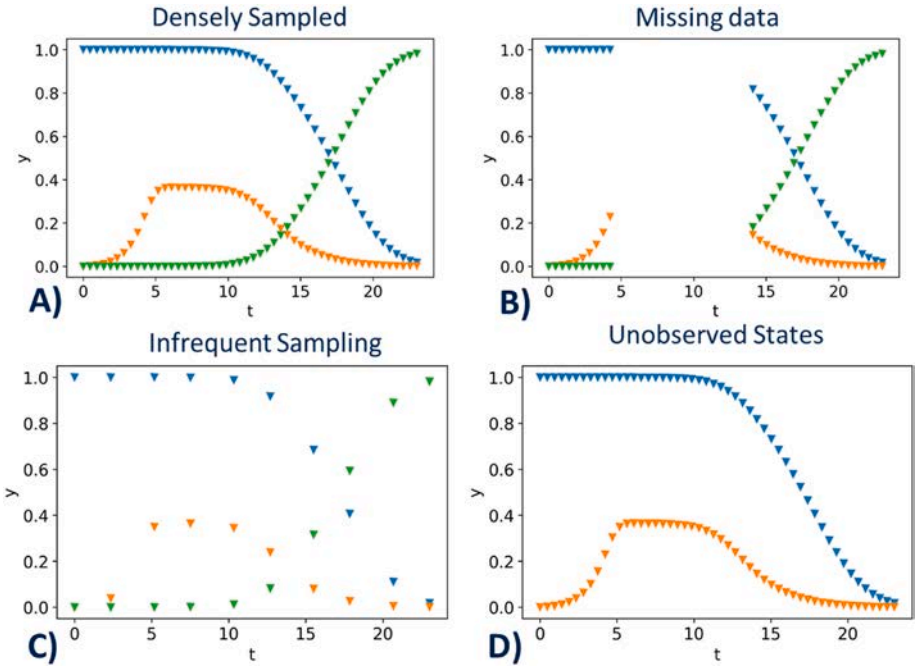

Fig. 1. Four Modes of Sparse Data. (a) Densely Sampled: ideal case of sufficient data for all states/outputs, (b) Missing Data: missing consecutive samples due to failure or disturbance, (c) Infrequent Sampling: equally-timed samples occurring less frequently than in Densely Sampled case, (d) Unobserved States: measurement of one or more states/outputs is entirely missing.

order are considered with the best approximations retained. Splines of degree 3 (4th order) are trained and simulated via SciPy's B-spline packages splrep and splev, respectively (Virtanen et al., 2020). As part of training, the parameters of B-splines are adjusted to reduce the scaled sum of squared errors between data and the interpolating function. When it is assumed that no noise is present in the measured data, the smoothing factor of the spline is fixed at zero (i.e., no smoothing). Otherwise, when noise is present, the smoothing factor is fixed at a value of *n-sqrt(2*n)* according to SciPy's default recommendations, where n is the number of datapoints in each time-series run.

2.2. NODEs

NODEs are formulated by adapting implementations of prior work (Bradley et al., 2021; Chen et al., 2018). Specifically, for examples comparing NODEs with FDs, the Neural Network component of the NODE has a fixed structure with a single hidden layer, 10 hidden nodes and hyperbolic tangent activation function. Euler's method was used to numerically integrate the NODE; the number of Euler steps was manually increased until the integration approximated the dynamics well. More automated and accurate methods for numerical integration of NODEs are becoming increasingly available (Rackauckas et al., 2020; Merkelbach et al., 2022), but these were not required to showcase the methods of this work. Some hyperparameters that were important to the training included the learning rate of the L-BFGS optimizer (Liu et al., 1989) and termination criteria for training. Training of the NODE terminated when the relative improvement in the loss function was less than a fixed tolerance over ten iterations. The chosen learning rate was fixed at 0.1 and the relative tolerance for termination was fixed at 1E-6. Further information about the NODE's formulation and training algorithm can be found in the Appendix.

Prior to training the NODE on this dataset, special attention is given to the methods by which the NODE is integrated during training. Where possible, rather than integrating the NODE across a single time interval spanning all data from an initial time t_o to a final time t_f, the NODE is instead integrated across shorter, overlapping time intervals as depicted in Fig. 2. As discussed in previous work (Bradley et al., 2021), integrating over shorter, overlapping intervals has the dual benefit of reducing the training time while also minimizing the probability of the differential data-driven model "over-smoothing" the data and converging to a local minimum. Further discussion on choice of NODE integration interval can be found in the Appendix and is further analyzed in results Section 4.

A comment should also be made on how the trained interpolators are simulated. As the finite difference and spline methods offer only local approximations, the states and derivatives are evaluated by calling the function at the time t where system estimates are desired. In contrast, the NODE could be simulated from any starting condition. This opens the question whether NODEs should use as inputs the states predicted by the NODE after being simulated from initial conditions to predict derivatives (i.e., their global approximation) or use the training data as inputs when predicting derivatives (i.e., their local approximation). In this work, the states predicted by the NODE (not the training data) are used as inputs to the NODE to globally approximate the derivatives. This makes the method more robust to noise or outliers in the training data. However, this also makes the NODE susceptible to poor performance when simulated over long time intervals. This idea is explored in more detail in the examples that follow.

3. Visualization of interpolation and derivative estimation accuracy using NODEs via a motivating toy problem

To illustrate how the fidelity of interpolating methods varies with various modes of sparsity, the Lotka-Volterra (LV) equations will be used to simulate the underlying system dynamics. The LV equations used in this study are shown in Equations (5) and (6) with the parameter values $a = 3.0$, $b = 0.6$, $c = 4.0$, and $d = 0.5$.

$$\frac{dx}{dt} = ax - bxy \tag{5}$$

$$\frac{dy}{dt} = -cy + dxy \tag{6}$$

These equations offer an ideal test case to investigate the approximation properties of data-driven interpolators due to the highly nonlinear, yet stable oscillation profile of the system states. In the forthcoming examples, the ability of interpolating methods to infer derivatives will be tested for densely sampled, infrequently sampled, and incomplete datasets with missing data.

3.1. Single trajectory, dense sampling

As a first example, system rates are inferred when state data is dense. To generate a dense data-set, the LV equations are simulated with the initial conditions $x_o = 10$ and $y_o = 3$ over the time interval $t = [0,4]$. The system states are measured at increments of dt = 0.08, yielding a dataset of 50 samples for each state. In this example, only a single state trajectory (i.e., one experiment or run) is used. For the densely-sampled data, the length of integration during training is equal to the span of 4 training samples. Graphical representations of the densely sampled data and its estimation via the three interpolating methods is shown in Fig. 3.

Due to the high quality and quantity of data, all interpolating methods perform well at smoothing the data. Moreover, the derivative estimates also follow closely the true system trajectory. The mean absolute error (MAE) of the derivative estimates of the NODEs, splines, and finite differences (FD) were 0.452, 0.385, and 0.369, respectively. Since no noise is present in the data, if a local approximation of the NODE derivatives is calculated using the training data as inputs, the MAE

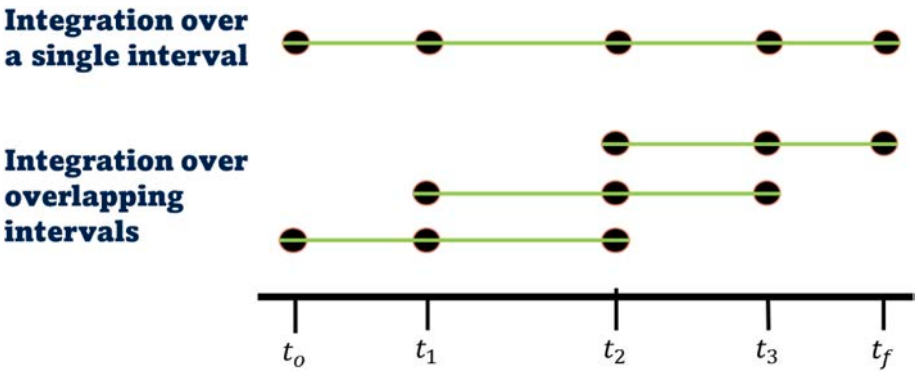

Fig. 2. (Top) Integration from a single timepoint spanning all sampling times or (bottom) integration from multiple timepoints using overlapping intervals spanning three sampling times.

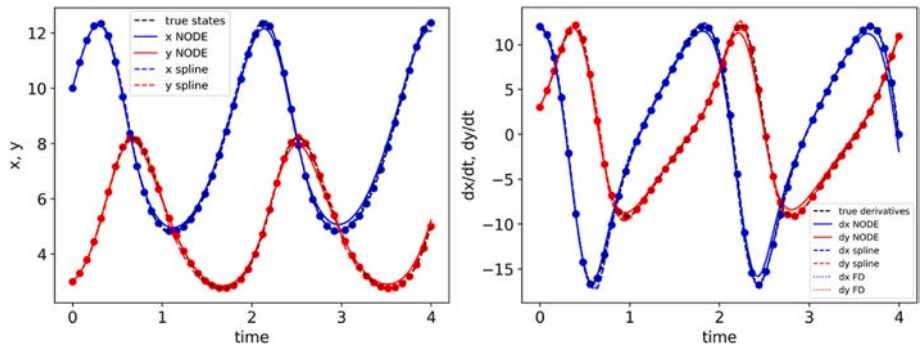

Fig. 3. State (left) and derivative (right) estimation from interpolating densely-sampled data. Fitted state data (left) and matching derivatives (right) represented by dots.

reduced slightly to 0.217. However, this error would substantially increase if the state measurements were less reliable. Regardless which approximation is used, clearly, under the conditions of abundant data, no interpolating method offers a significant advantage for estimating derivatives. However, real systems are rarely so well-sampled. Subsequent examples of this study will differentiate the methods based on their ability to infer derivatives on sparser systems.

3.2. Multiple trajectories, missing data

Next to consider is the mode of sparsity incurred when data is absent in important dynamic regions. An example of this dataset is depicted in the upper graphs of Fig. 4. In this dataset, entire oscillations of the state profile are not captured, and local interpolation of the data results in large errors. If the NODE were trained on only the datapoints of the sparsely sampled experiment, it would likewise fail to capture the dynamics. However, this can be overcome by combining datasets from multiple experiments. To demonstrate this, a second experiment is performed that captures more of the oscillatory profile not captured in the first experiment. The NODE is trained using all data from both experiments and during training the NODE is integrated over shorter intervals for the densely-sampled trajectory and over the entire interval for the sparely-sampled trajectory. The result of training the NODEs on the combined datasets is shown in Fig. 4. Clearly, the dynamics captured by the NODE from the second experiment have informed the predictions of the NODE on the first dataset. This idea of combining data sources for more generalized prediction is analogous to the idea of transfer learning in machine learning (Zhuang et al., 2021; Weiss et al., 2016). However, unlike some transfer learning applications, the data-driven interpolating model is not trained sequentially on the different datasets. Rather, all the data is used to train the NODE at once.

Obtaining the spline and finite-difference approximations is also done by fitting all the data simultaneously. However, unlike the NODE, no transfer learning occurs. As seen in Fig. 4, the derivatives inferred by the algebraic interpolating methods cannot capture the dynamics of the first experiment, though they interpolate the second experiment well. Imaginably, if the two experiments were the result of the same system conditions (i.e., the initial conditions were the same), then the two datasets could be trivially combined into a single experiment. However, barring this trivial case, any method employing an algebraic data-driven

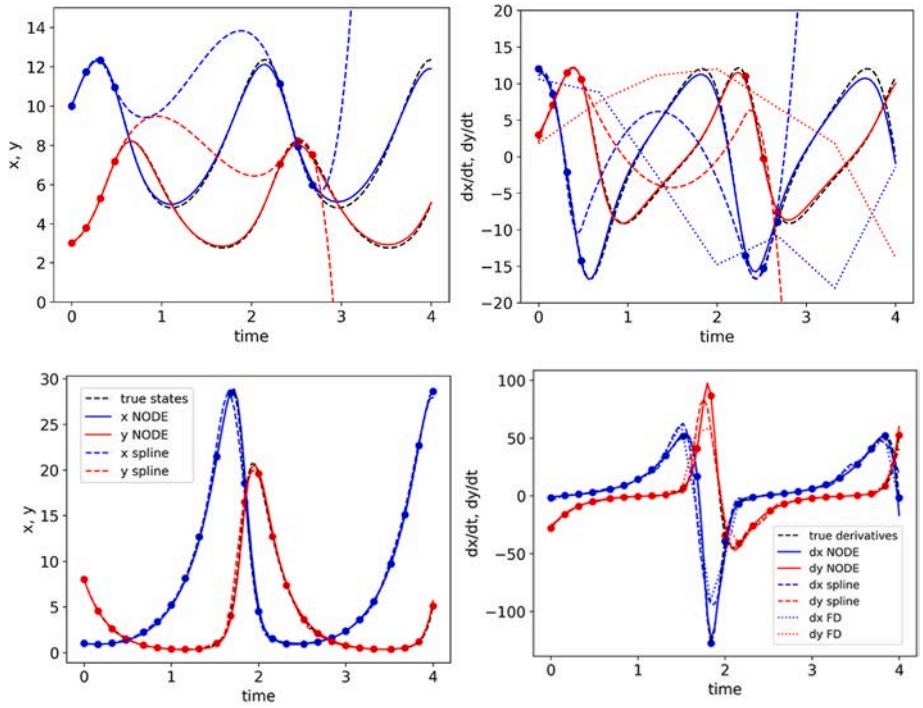

Fig. 4. Interpolation of 2 experimental datasets. Experiment 1 (top) sparsely sampled and experiment 2 (bottom) densely sampled. Fitted state data (left) and matching derivatives (right) represented by dots.

model using time as an input must use multiple models to interpolate data from different experiments. When the different experiments include orthogonal pieces of information (as a well-designed set of experiments should), the NODE is anticipated to outperform the algebraic interpolating methods.

Another interesting observation can be made if the datasets are reversed. Shown in Fig. 5 is the attempt to infer the states and derivatives of the LV system when experiment 1 is well-sampled but experiment 2 is not. Although the NODE is trained on data from both experiments and interpolates the densely-sampled experimental data well, the poorly sampled profile is not simulated accurately by the NODE in its most nonlinear region. Indubitably, this discrepancy is the result of experiment 2 having a greater variance in the values of its response and therefore more information for the NODE to learn on. The profile in experiment 1 is an interpolation of the behavior captured in experiment 2 though the reverse is not true. This highlights the importance of capturing the extrema of a system's response prior to data-smoothing, a principle that also bears weight when samples are evenly, but infrequently measured.

3.3. Multiple trajectories, Infrequent Sampling

As a third comparison between NODEs and its competitors, the methods are used to interpolate data with infrequently collected samples. For this mode of sparsity, the states are sampled at distant though evenly-spaced time intervals. An example dataset consisting of seven samples for the LV system is displayed in Fig. 6. The true state and derivative profiles are superimposed by the interpolated estimates. The interpolating methods are trained as in previous cases, except the length of time used for the NODE's method overlapping intervals spanned fewer (3 instead of 4) to avoid convergence to an over-smoothing local minimum. Once again, neither the spline nor FD methods could offer an excellent approximation of the state profile, resulting in poor derivative approximation.

Somewhat surprising, however, is the excellent interpolation provided by the NODE. Not every extremum is sampled in the training data, yet all the extrema are captured by the NODE interpolation. This result is believed to be a consequence of the symmetric response of the LV system. Since the state response of the LV system is cyclic, the oscillating pattern captured by the first three data points helps the NODE learn the oscillating pattern of the second cycle. Admittedly, this behavior may not fully generalize to other systems where the response does not strictly follow a cyclic pattern. Nevertheless, this example offers a useful contrast between the local approximations offered by the algebraic smoothing methods and the benefits of the global approximation provided by a NODE. The NODE is able to transfer learning from one part of the system response to infer behavior in a different part, even within a single experiment.

To further test the limits of the NODE's ability to transfer learning, the sample size is further reduced to 5 (initial conditions included). Once more, the NODE is trained on data from a single experiment, the distance of integration spanning the length between two points. However, this test was found to be far more difficult for the NODE. Fig. 7 juxtaposes the true state and derivative profiles with those estimated by the NODE. This time, the NODE finds the complete opposite of the true oscillation profile, yielding estimates that are clearly wrong. Attempts to improve the approximation by integration over longer intervals or modifying the learning rate or other training hyperparameters either diverged or converged to same local minimum. In summary, the dataset is too sparse to identify the true state profile.

To improve the approximation of the NODEs, additional experimental trajectories were added to the training dataset until the MAE of the NODE derivative estimates resemble the mean absolute error of the densely sampled data. Each additional dataset includes the same number of equally-spaced datapoints as the first experiment (i.e., five) within the same time range, but the initial conditions of the two states x and y vary randomly within the range 1 to 14. With each additional dataset, the NODE is trained from the same initial parameter values on the entirety of the new dataset. The result of successively augmenting the number of experiments used to fit the NODE is captured in Fig. 8. The MAE of the first column corresponds to the fit of the NODE in Fig. 7, wherein only a single experiment was used for training. The critically informative number of experiments was found to be 5, consisting in total of 25 data points (including initial conditions). Beyond this number, additional experiments did not substantially improve the estimation accuracy of the NODE.

At least two noteworthy observations can be made from the trend in Fig. 8. First, the interpolation worsened when a fourth experiment was

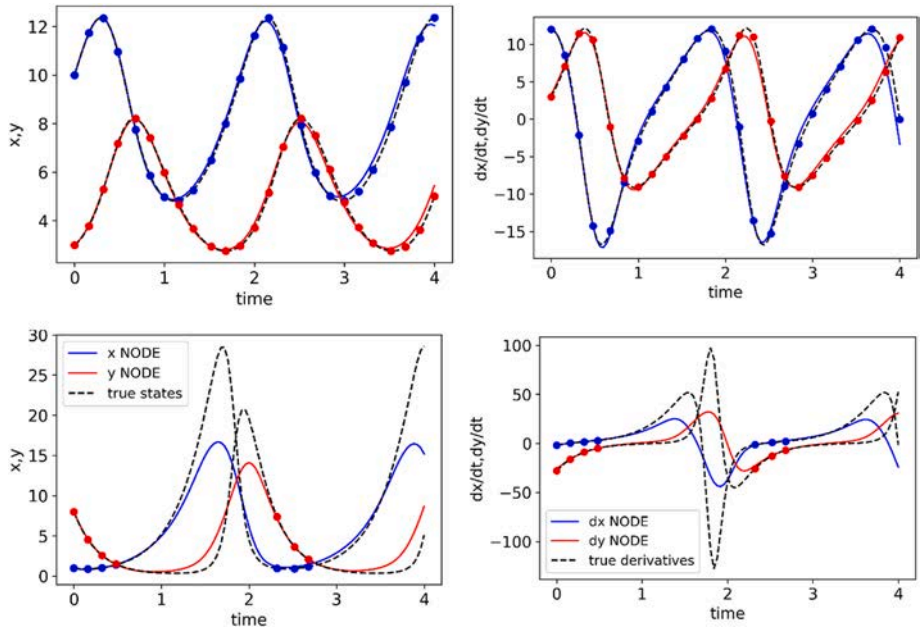

Fig. 5. Interpolation of 2 experimental datasets. Experiment 1 (top) densely sampled and experiment 2 (bottom) sparsely sampled. Fitted state data (left) and matching derivatives (right) represented by dots.

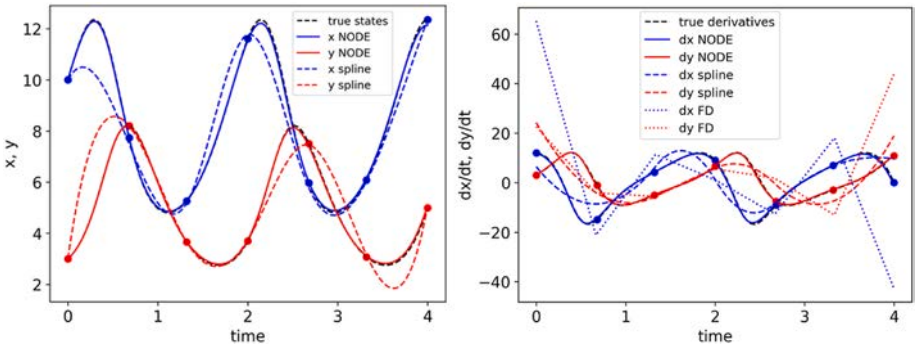

Fig. 6. Interpolation of data available at 7 evenly-spaced times from a single experiment. Fitted state data (left) and matching derivatives (right) represented by dots.

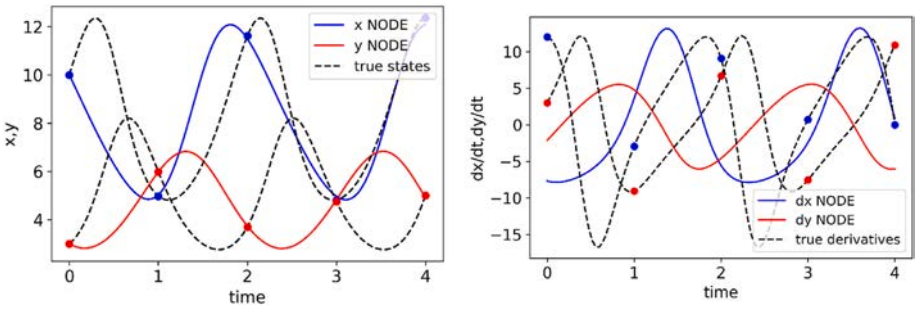

Fig. 7. NODE predictions after training with 5 datapoints from a single experiment. Fitted state data (left) and matching derivatives (right) represented by dots.

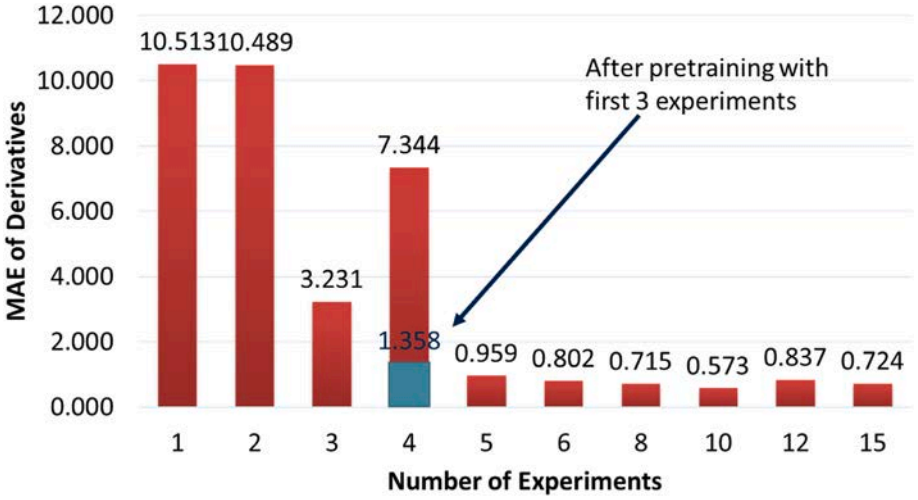

Fig. 8. Interpolation error of NODEs for a varying number of experiments. Also, improved interpolation of 4 experiments after pre-training.

added to the training data. This is unexpected since the four experiments include the same data as was present in the three-experiment case plus five more data points. Adding a fourth experiment should therefore be more informative than the three-experiment case. Upon visualization of the trained NODE predictions, it was found that the trained model had converged to the opposite oscillation profile than the true system, similar to Fig. 7. To remedy this, it was decided to follow a "transfer-learning" approach, and to pre-train the NODE with the datasets of the 3 experiments before training the NODE on all 4 datasets. This pre-training brought about a far superior approximation of the true profiles with a final MAE of 1.358. In summary, this exercise showed the importance of NODE initialization when data samples are few.

Second, the number of samples required to converge to the true profile was greater than the 7 data points used in the previous task (see Fig. 6). The clarifying explanation for this observation is that the performance of the NODE interpolation depends less on the number of samples and moreso on how well the samples of any one experiment cover the extrema of the dynamic response. As a way to validate this conclusion, the number of samples per experiment was further reduced. In the case of the LV system, if the number of equidistant samples was reduced to less than 5 per experiment, the data of any given experiment no longer covered the extrema of the oscillating peaks. Under these circumstances, no number of experiments no matter how large could induce the NODE to reproduce the LV system's oscillating profile. A way to explain this finding is the breakdown of the method for integration during training. When data no longer covers the curvature of the system response, integration over shorter overlapping intervals no longer is more beneficial than simply integrating across the entire time domain.

Yet integrating the NODE across an oscillating system response ultimately converges to a local "over-smoothed" minimum even for a densely sampled experiment (see examples in the Appendix and in (Bradley et al., 2021)). Based on these observations, we offer the heuristic rule that for a NODE to interpolate a dataset well, regardless of the number of experiments, in at least one experiment the local extrema must be identifiable. While this may seem like a severe limitation of the NODE method, especially if all experiments are sparsely sampled, it's worth noting that not all systems experience the severity of fluctuations observed in the LV system and the interval between samples need not be fixed. The last example of this study uses such an example for illustrating how NODEs can infer not only derivatives but an interpretable mechanistic model.

4. Case studies: NODE interpolation and derivative estimation under noisy data scenarios

In the previous results section, the LV system was used to illustrate qualitatively how NODEs can more accurately interpolate data. Under certain sparsity conditions, NODEs offered more reliable estimates of system dynamics than splines or FDs. However, these examples may not be sufficient to demonstrate how well these conclusions extrapolate to other dynamic systems or when data is encumbered by noise. Additionally, it remains unclear what role the length of integration of the NODE plays in the accuracy of the trained model. To address these concerns, the next section offers a more quantitative comparison of NODE performance versus splines on 3 cases studies. These include the Van der Pol (VDP) oscillator, Susceptible, Infected, Resistant (SIR) model, and FitzHugh-Nagumo (FHN) model. These commonly-used benchmarks represent dynamic relationships used to model a variety of physical and engineering systems from fluid vibrations in a pipe (Xie et al., 2019), rate of transmission of a pathogen within a population (Word et al., 2012), and electric circuits (Ma et al., 2019), respectively. Mathematical formulations of these models can be found in the Appendix of this work.

4.1. Effect of NODE interval of integration

Before comparing NODEs with splines on noisy data, we first focus on the effect of integration interval length on NODE-based interpolation accuracy. Rather than randomly tuning this hyperparameter, we hypothesize that there is a relationship between interval length and NODE accuracy. To ascertain this, the interval of integration of the NODE was varied while being trained on various data scenarios. Specifically, the NODE was fit to each of the benchmark systems assuming the availability of either 5 or 10 experiments (i.e., runs) of data with 10 measurements each. Noise was added to the data randomly from a normal distribution whose variance was equal to 0, 5 or 10% of the range of data measurements. Meanwhile, the length of overlapping integration intervals during training varied to span either 4, 7, 9 or 10 datapoints. To factor in variation from model size, the Neural Network was varied to include 1 or 2 hidden layers with 10, 15, or 20 hidden nodes. Due to the presence of noise, it was decided to include the initial conditions as adjustable parameters during training. The learning rate was fixed at 10^{-6} and training was stopped when NODE predictions did not improve by more than 10^{-6} for 10 consecutive iterations.

After training, the accuracy of the NODEs was calculated by simulating the trained NODE once for each experiment starting at $t = 0$ until the end of measurement data. The state predictions at measured times were then used as inputs to the NODE to obtain derivative estimates. The mean squared error (MSE) between predicted derivatives and true derivatives from NODE state estimates was calculated. Visual examples of the variation in prediction error are provided in the Appendix.

A statistical analysis was performed to determine if the improvement from increasing the integration interval was indeed significant for certain systems or data scenarios. For this analysis, a one-sided Mann-Whitney *U* Test (MWUT) was chosen. The MWUT is the non-parametric equivalent of the *t*-test, preferred for this comparison as the errors among fitted NODE models were found not to be normally distributed. A separate MWUT is used to compare each of the longer integration intervals (7, 9, or 10 data span) with the NODE trained using the shortest interval (4 datapoints per interval). Thus, the null hypothesis is that an interval spanning 7, 9 or 10 datapoints does not improve NODE derivative estimation accuracy over a short interval of 4 datapoints.

The p-values from these MWUTs are presented in Fig. 9. For this study, a p-value under 0.05 is considered statistically significant and these values are highlighted in red. According to Fig. 9, for all 3 case studies (VDP, SIR, and FHN), there is a significant improvement in error when the NODE is integrated over longer intervals when data is encumbered by the highest level of noise. Especially when less data is available (5 runs instead of 10), by integrating across the entire span of 10 measurements during training, the fitted NODE consistently simulates more accurate derivatives than when the shortest integration intervals are used. Again, the reasoning for this is the tendency for the NODE to overfit the noisy data and its failure to learn the longer-range dynamics when the NODE is simulated on shorter time intervals during training.

Somewhat less expected is the improved performance of the NODE when trained with longer intervals for the SIR system even when noise is low. A couple explanations can be offered for this behavior. First, the SIR initial conditions vary over larger orders of magnitude, requiring the NODE to predict state values more precisely, which may be more difficult when NODEs are not fitted to the long-term trajectories during training. In a similar vein, a second unique feature of the SIR system is the asymptotic dynamics. Due to the potential for the rate of change to approach zero, the NODE must predict derivative values near zero, a difficult task for Neural Networks in general. Training over shorter intervals results in a failure to learn the asymptotic dynamics and therefore a failure to capture those dynamics when simulating the NODEs over longer-ranges. Further work is needed to establish the capabilities of NODEs when trying to capture asymptotic or 'stiff' dynamics.

As a final check on the above conclusions, another MWUT was also performed, using as the alternative hypothesis that a shorter interval of integration (4 datapoints) offered an improvement in error over longer integration intervals. This is merely a reverse of the previous statistical test using the same data-fitting scenarios as before. Of all scenarios tested only a single case produced a p-value below the 0.05 significance threshold (SIR, 10 runs, 5% noise). However, this result is believed to be more due to random chance than indicative of a contrasting trend. We therefore reject the idea that shorter integration intervals would reduce overfitting.

Put together, the above statistical analyses lead us to conclude that NODE overfitting is reduced by training NODEs over longer intervals, in particular when data is sparse and noisy. These results are not surprising in light of previous work demonstrating superior accuracy when NODEs are trained to learn the entire time-trajectory of a system's response (Rico-Martínez et al., 1992). A key difference between the current study and previous work is demonstrating this trend remains true even in the presence of noise. Especially when data is noisy, integration over longer intervals prevents overfitting and enables the NODEs to better learn the dynamics. Thus, so long as the integration interval is not so long as to over-smooth the data (a scenario described further in the Appendix), the modeler should opt for training the NODE over longer intervals to increase NODEs' ability to discover the true dynamics.

4.2. Effect of noise on NODE interpolation

Having confirmed the importance of training NODEs with longer integration intervals, we now investigate their potential to more accurately estimate derivatives versus algebraic interpolation for a broad collection of dynamic systems in the presence of noise. For this comparison, NODEs and splines are used to regress data simulated from the

VDP

Data pts/interval	5 runs			10 runs		
	0% Noise	5% Noise	10% Noise	0% Noise	5% Noise	10% Noise
7	0.757	0.196	0.0075	0.242	0.242	0.154
9	0.650	0.154	0.331	0.089	0.705	0.294
10	0.879	0.534	0.046	0.268	0.531	0.089

SIR

Data pts/interval	5 runs			10 runs		
	0% Noise	5% Noise	10% Noise	0% Noise	5% Noise	10% Noise
7	0.590	0.879	0.066	0.020	0.803	0.409
9	0.154	0.012	0.062	0.025	0.979	0.242
10	0.757	0.0010	0.020	0.0205	0.468	0.004

FHN

Data pts/interval	5 runs			10 runs		
	0% Noise	5% Noise	10% Noise	0% Noise	5% Noise	10% Noise
7	0.705	0.531	0.032	0.757	0.590	0.046
9	0.953	0.066	0.007	0.910	0.242	0.032
10	0.590	0.089	0.004	0.590	0.196	0.020

Fig. 9. List of p-values from calculated using MWUT based on likelihood of a using an interval spanning 7, 9, or 10 datapoints to improve NODE accuracy vs an interval spanning 4 datapoints for VDP (top), SIR (middle), and FHN (bottom) systems and data scenarios. p-values less than 0.05 highlighted in red.

VDP, SIR, and FHN systems and to estimate state derivatives. As FDs performed similarly to splines in Section 3, they are omitted from this final comparison study.

To maximize the breadth of this assessment, the conditions of training data were varied to include differing amounts of sparsity and noise in the data. Specifically, the number of experiments included in the training dataset varied between 1 and 10. Each experiment included 3, 5, or 10 equally spaced measured datapoints. Moreover, normally distributed noise equal to 0, 5, or 10% of the range of the measured data was added to the data.

The length of integration of the NODE during training was selected to balance overfitting caused by shorter intervals of integration, while minimizing the chance of over-smoothing when longer intervals are used. Based on our prior analysis, the recommended default setting (or heuristic) that balances overfitting vs. over-smoothing is is to fix the integration interval to the time spanned by the number of measurements minus one. Although this heuristic worked for the current systems, it may not be appropriate for other systems, whose length of integration may need to be adjusted based on the system's characteristics (i.e., nonlinearity, noise level). The size of the NODE was varied to include 10, 15 or 20 hidden nodes in either 1 or 2 hidden layers. Each of these NODE structures was fitted, and the NODE with the lowest state prediction error for a given data scenario was assumed to be the 'best' NODE to estimate the true dynamics. Other hyperparameters such as learning rate and stopping criteria are the same as in the previous section. Same as the previous studies, after training, the NODE is simulated forward in time starting from the initial conditions of the experiments used for training. Thus, a NODE fitted to three experiments estimates only the derivatives of the three experiments it was trained on and at measurement times.

Depicted in Fig. 10 is a summary of the mean squared derivative errors of the two methods on three benchmark systems for data encumbered by increasing noise and number of runs. Highlighted in each table are the data scenarios where the NODE or spline estimates are more accurate (lower error) than the alternative interpolating model. Several key conclusions can be readily made from Fig. 10. First, increasing the number of runs (i.e., experiments) does not necessarily translate to a lower average error for either data-driven interpolation method. This may seem counterintuitive as increasing runs means more data is being fed to the interpolation method. However, it should be kept in mind that each run differs in its initial conditions and the complexity of the true dynamics. Increasing the number of runs therefore introduces the possibility of adding a dynamic trajectory that differs significantly from previous experiments or may be more difficult to capture, which would lead to a higher average error. However, equally important, as the number of runs increases, Fig. 10 shows a clear trend of NODEs more frequently offering better estimates than splines of the system derivatives across all systems. This trend holds even when there is a significant amount of noise in the data. In direct contrast, when only a single run of data is available for fitting, in less than half of the data scenarios do NODEs offer a better estimate of the dynamics. In other words, the likelihood of improvement by NODEs over splines in these low-run, sparse scenarios is worse than random chance. Thus, it can be concluded that the greater accuracy of the NODE is strongly tied to the presence of multiple time-series for it to learn from.

In addition to highlighting trends across systems, some comments should be made about the individual benchmarks. Noticeably, the NODEs generally do not offer better results than splines for the VDP benchmark when only 3 measurements are made, regardless of the number of runs. As with the LV system investigated in Section 3, the extreme sparsity of measurements when only 3 measurement per run are available means no single experiment captures the oscillatory dynamics of the data, and the NODE cannot identify the true dynamics. Another anomaly noticeable in the VDP results is the extraordinarily high MSE for one of the cases with a single run of fitting data. For this particular case (5% noise, 1 run, 5 measurements), all six of the fitted NODEs diverged during training, making even the best NODE's estimates

VDP - spline

Runs	Noise % 0	5	10	Dps/Run
1	4.89	6.22	7.13	3
	5.59	1.56	2.03	5
	3.72	0.99	1.03	10
3	9.90	10.05	12.66	3
	6.71	3.00	2.27	5
	3.15	2.98	3.07	10
5	11.03	9.86	12.67	3
	7.92	7.53	7.30	5
	4.88	2.88	2.88	10
10	10.09	11.55	16.99	3
	7.14	8.78	7.86	5
	4.36	3.80	3.32	10

VDP - NODE

Runs	Noise % 0	5	10	Dps/Run
1	4.67	6.32	7.54	3
	9.41	8.73E+05	4.59	5
	2.64	1.22	83.15	10
3	13.71	11.48	22.22	3
	2.45	8.05	8.49	5
	0.78	0.97	2.19	10
5	7.35	15.06	18.56	3
	1.52	2.24	10.75	5
	0.68	1.06	2.09	10
10	14.68	12.88	20.61	3
	3.32	2.28	3.54	5
	1.03	2.58	1.62	10

SIR - spline

Runs	Noise % 0	5	10	Dps/Run
1	0.0064	0.0057	0.0063	3
	0.0133	0.0027	0.0020	5
	0.0047	0.0038	0.0038	10
3	0.0081	0.0064	0.0066	3
	0.0129	0.0032	0.0035	5
	0.0091	0.0043	0.0038	10
5	0.0061	0.0065	0.0094	3
	0.0082	0.0026	0.0039	5
	0.0055	0.0029	0.0029	10
10	0.0045	0.0053	0.0085	3
	0.0051	0.0022	0.0039	5
	0.0028	0.0022	0.0029	10

SIR - NODE

Runs	Noise % 0	5	10	Dps/Run
1	0.2312	0.0440	0.0107	3
	0.0128	0.0627	0.0571	5
	0.0051	0.0109	0.9566	10
3	0.0235	0.0062	0.0088	3
	0.0012	0.0050	0.0123	5
	0.0004	0.0010	0.0020	10
5	0.0057	0.0496	0.0202	3
	0.0035	0.0095	0.0217	5
	0.0001	0.0007	0.0019	10
10	0.0090	0.0042	0.0157	3
	0.0006	0.0026	0.0134	5
	0.0008	0.0008	0.0009	10

FHN- spline

Runs	Noise % 0	5	10	Dps/Run
1	14.00	12.41	12.38	3
	7.99	6.68	6.05	5
	3.17	2.10	2.80	10
3	14.44	16.09	21.05	3
	7.97	8.43	14.86	5
	2.31	10.78	18.54	10
5	12.53	12.96	15.37	3
	5.96	7.39	11.32	5
	1.77	9.29	15.11	10
10	8.51	9.44	13.03	3
	3.70	6.23	8.60	5
	1.40	7.09	10.15	10

FHN - NODE

Runs	Noise % 0	5	10	Dps/Run
1	13.74	11.95	12.45	3
	7.83	9.19	9.68	5
	3.70	4.28	29.30	10
3	3.78	17.58	14.75	3
	4.85	7.88	10.34	5
	1.35	2.46	31.78	10
5	5.16	4.81	19.85	3
	3.15	4.75	7.84	5
	0.82	1.68	29.80	10
10	3.18	5.06	22.10	3
	1.19	2.48	6.18	5
	0.74	0.93	4.73	10

Fig. 10. Mean squared error of derivative predictions from 'best' NODE and spline trained on various data scenarios from 3 dynamic systems. Highlighted in red are data scenarios where an interpolating model is more accurate than its competitor.

especially poor. Selecting different hyperparameters could lead to fitting a NODE model that doesn't diverge during training for this data scenario, but the accuracy of the NODE would still be expected to be poor due to the limited training samples.

Highlighting yet another trend, although it is generally the case that both interpolation methods improve as the number of measurements per run increases, this is not universally true. Taking the FHN system for example, at peak levels of noise, more measurements per run sometimes leads to worse estimates on average. Visual inspection of the fitted model's predictions in these cases found a tendency for increased overfitting of the noisy data. Though not explored here, tuning of the NODE's hyperparameters such as the regularization penalty or stopping criteria could help mitigate some of these overfitting issues.

Finally, it is interesting to observe that the NODEs performed better than splines least frequently for the SIR system. A few factors are considered to cause this behavior. For starters, the average errors for this system are orders of magnitude smaller than the other two systems, making it difficult to get a substantial improvement by either method. Second, as previously mentioned, the SIR system is unique due to it having asymptotic and stiff dynamics, which are difficult for NODEs to capture. Only when the training data includes 5 or more runs do the NODEs generally offer an improvement over splines.

5. Enabling model identification via NODEs

Although Fig. 10 in Section 4 demonstrates that NODEs consistently offer better accuracy than algebraic interpolating splines when data is sparse and an increasing number of experiments are available, it is not immediately obvious whether that improvement in accuracy is substantial. In general, inferring derivatives is not the end goal of data interpolation. Even more impactful is when the estimated derivatives are used for parameter estimation, sensitivity analysis or model discovery. In our previous work (Bradley et al., 2021), derivatives estimated via NODEs were used to estimate parameters of nonlinear ODEs

with speed and accuracy that surpassed direct estimation methods. In this work, as a final and more challenging case study, we will illustrate the impact of the proposed method on model identification (i.e., model discovery). In brief, model identification can be viewed as a parameter estimation problem wherein not only the parameters but also the mechanistic terms of the model are unknown.

Efforts to develop optimal model identification algorithms are numerous and ongoing. However, among the frameworks for discovering DEs for dynamic systems, Sparse Identification of Nonlinear Dynamics (SINDy) has received extensive attention in recent years for its ability to identify the correctly formulated differential equation from among a combinatorially large number of candidate models (Brunton Steven et al., 2016). SINDy 'discovers' the true model through a two-stage algorithm. The first step is to estimate the time derivatives of measured data (i.e., by interpolation) and the second is to solve the model identification problem by solving a regularized least-squares problem that penalizes models with an excessive number of terms. Notably, the term 'Sparse' in the SINDy framework refers to the goal of generating an accurate model with the fewest number of terms, which differs from the usage of 'sparse' in this work to refer to datasets with limited samples. Readers interested in further details of the SINDy approach should consult the original publication (Brunton Steven et al., 2016) as well as its numerous extensions (Kaptanoglu et al., 2022; Kaheman et al., 2020, 2022; Mangan et al., 2019; Kaiser et al., 2018).

In the original version of the SINDy algorithm, the derivatives used to solve the model identification problem were acquired through finite difference or spline interpolation methods. This was appropriate when data was densely sampled around a single time trajectory. In contrast, the global interpolation properties of NODEs should allow them to supply more accurate derivatives to the SINDy algorithm when trained on multi-experiment data, increasing the likelihood of SINDy identifying the true model when data is sparse. To prove this point, the derivatives estimated in one of the case studies of the previous section were used to identify the true model using the SINDy algorithm, assuming of course the true model is unknown. To our knowledge, no previous work has applied NODEs as the derivative estimation component of SINDy algorithm when data is stratified across multiple experiments and thus their potential advantage for model identification is an open question.

The VDP system was chosen as the focus for the model identification comparison as the frequency of improvement from NODEs over splines for this system fell in between that seen in the other two benchmark studies. The first step in the SINDy framework requires predefining a library of candidate terms from which SINDy selects which terms to include in the identified differential equation model. To ensure a fair comparison, the same set of candidate terms were selected for consideration for every data collection scenario, which included polynomial terms up to third order. The objective function used to identify the true model is a regularized regression function of the form (7), and is minimized via a sequentially-thresholded least squares regression algorithm available through SINDy's implementation.

$$\min \sum \frac{1}{2}\left(\dot{\mathrm{X}} - \Theta(X)\Xi\right)^2 + \lambda R(\Xi) \tag{7}$$

In Eq. (7), $\dot{\mathrm{X}}$ and X represent the derivatives and states estimated via interpolation, respectively, Θ is a vector of candidate model terms, and Ξ the parameters of those terms. To penalize non-parsimonious models, a regularization penalty $R()$ multiplied by the hyperparameter λ is added. In this work, $R()$ is the l-2 norm and the optimal value of λ was found by sweeping through a range of lambda values, performing the sequentially-thresholded least squares regression each time. Specifically, for each of 10 lambda values in the range 0.1–1.0, Equation (7) was minimized using the derivatives and states predicted via either splines or a NODE. Next, the model selected via Eq. (7) was integrated on the interval $t = [0,10]$, and the sum of squared errors (SSE) between the candidate model predictions and training data was calculated. Using the SSE, the corrected Akaike Information Criteria (AIC_C) was evaluated to rank the models found by the different lambda values.

$$AICc = n\ln\left(\frac{SSE}{n}\right) + 2k + \frac{2k^2+2k}{n-k-1} \tag{8}$$

In Equation (8) n is the number of training samples and k is the number of terms in the discovered model.

To provide a rich dataset for SINDy to fit to, the fitted interpolating model simulated derivatives for all runs used for training at both measured and unmeasured times. Specifically, states and derivatives used in SINDy are those estimated by the interpolating model at 100 equidistant points within the time interval of training data, t = [0,10]. In addition, there is concern that data-driven interpolation may perform poorly at estimating derivatives at initial conditions. This was noticeable source of error in our previous work for NODEs and has been described for splines at the edges of interpolation (Gauthier et al., 2020). To combat this, model identification with SINDy was also attempted using 90 equidistant predictions, eliminating the initial 10 estimates of each experiment (i.e., using the interval spanning t = 1 and t = 10). Thus, all in all, for every data scenario, SINDy algorithm is called twenty times (10 lambdas times 2 sets of training data) and will produce twenty models. The model with the lowest AICc score is selected as the 'true' model, finishing off the model selection workflow.

An example of the relative AIC_C scores are plotted in Fig. 11 for SINDy models found from 10 different lambda values for the VDP system, for the case where interpolating models were fit to data with 5% noise, 5 runs and 10 measurements per run. Also plotted in Fig. 11 are the number of false positive and false negative terms in the SINDy-identified models using each lambda value and fitting dataset. A model is reported to have a false positive term if a term is present in the SINDy-selected model that is not present in the true model and a model is reported to have a false negative term if a term present in the true model is not present in the discovered model. Notable in Fig. 11 are some cases where no AICc value is reported, which occurs whenever the SINDy-selected model diverges when simulated. Under these circumstances, no AICc value can be calculated.

For the case presented in Fig. 11, the AICc criterion assigns the lowest score when $\lambda = 0.5$ and 100 NODE estimates per run are used, which corresponds to the true model with no false positive or negative terms. In contrast to the selection from the NODE estimates, the best model obtained by the SINDy algorithm using the spline estimates is when a lambda threshold of 0.1 is used. However, this model contains incorrect terms not a part of the true model (false positives) and is missing terms that should be present in the true model (false negatives). Further increasing the regularization simply leads to elimination of terms that should be included while decreasing the regularization results in a model with additional terms not present in the true model. The derivatives estimated by the splines are simply not accurate enough to extract the true simulating model from the sparse data.

To verify how well these conclusions generalize, the same analysis depicted in Fig. 11 was performed for other data scenarios and case studies. These results are summarized in Fig. 12. In Fig. 12, for each data scenario for the VDP system, the number of false positives and false negatives in the 'best' model found through the SINDy algorithm are reported. For example, highlighted in red are the cases where a "0/0" is reported, which indicates the selected model has zero false positives and zero false negatives.

Overall, the small number of data cases where the true model is selected attests to the difficulty of solving the model selection problem when data is especially sparse. To select the true model, SINDy must select the correct 4 terms from the simulating VDP model from a library of 22 candidate terms. Data provided to SINDy's regression algorithm from a single run is insufficient to differentiate the true model, regardless which data-driven model is used. This does not contradict results of other studies, wherein a single time-series dataset was sufficient to

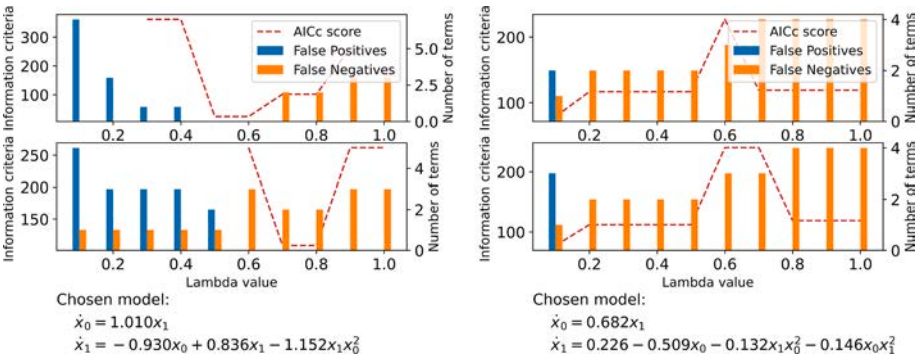

Fig. 11. Number of incorrect terms and information criterion score for SINDy-selected models using either 100 (top) or 90 (bottom) estimates per run from a fitted NODE (left) or spline (right) model.

SINDy-Selected Model from Neural ODE Estimates

Runs	Noise % 0	5	10	Dps/Run
1	0/2	0/2	2/2	3
1	0/4	Diverged	0/2	5
1	0/3	0/3	0/3	10
3	1/1	0/1	0/1	3
3	0/0	3/1	1/2	5
3	0/4	0/2	0/0	10
5	0/2	0/4	1/2	3
5	0/0	0/2	5/0	5
5	0/0	0/0	0/1	10
10	0/1	1/1	1/2	3
10	0/0	0/0	0/2	5
10	0/0	0/0	0/0	10

SINDy-Selected Model from Spline Estimates

Runs	Noise % 0	5	10	Dps/Run
1	0/4	0/4	0/4	3
1	0/4	3/2	0/2	5
1	1/0	1/1	1/1	10
3	0/4	0/1	0/1	3
3	0/4	0/4	0/4	5
3	0/0	1/0	4/0	10
5	0/4	0/4	0/4	3
5	0/4	2/2	1/2	5
5	0/1	3/1	0/1	10
10	0/4	0/4	0/4	3
10	0/2	1/2	0/2	5
10	0/1	0/0	0/1	10

Fig. 12. Models found by the SINDy algorithm using fitted NODE (left) or spline (right) estimates for a variety of data scenarios.

capture all system dynamics. However, whereas those studies assumed data was densely sampled, due to the sparsity imposed by the current study, not enough measurements are available to capture the complete dynamics in a single run.

Only for two data scenarios does the model selection algorithm select the true model when spline-fitted estimates are used. For these cases, even though spline estimates are not as accurate as NODE predictions, they are accurate enough to select the true model. Nevertheless, these successes occur only when each experiment contains the maximum amount of data coverage for each run of 10 datapoints/experiment. In direct contrast, the estimates from the NODE can identify the true model for several cases where the spline estimates cannot, most especially when multiple experiments of data are available to train the NODE. The NODE captures the true dynamics by transferring knowledge of the system response learned across multiple experiments, even when measurements are highly encumbered by noise. For these data scenarios, not only are the NODE estimates more accurate than splines, but their accuracy is sufficiently superior to discover the true underlying model when splines cannot. This offers a palpable example of the benefits of having more accurate derivative estimation that comes from multi-experiment interpolation with a single interpolating model.

6. Discussion

The examples in this work have illustrated the benefits of using a global interpolator such as NODEs when inferring derivatives from data which is sparsely sampled across multiple experiments. However, like any tool, NODEs are not a panacea for all problems and some discussion of their shortcomings and potential for improvement is merited.

The choice in this work to approximate the right-hand side of an ordinary differential equation with a Neural Network is motivated by a Neural Network's high interpolating potential, scalable to high dimensions, but there is no rule requiring that the data-driven approximator of a DE be a Neural Network. The differential form of the NODE enables it to be trained on data from multiple experimental datasets, a situation for which there is no straightforward way to train a single algebraic model if derivative estimation is the goal. However, this conclusion is not meant to exclude the possibility that other data-driven models may approximate a differential equation equally well. Especially interesting would be a differential data-driven approach that better approximates derivatives that vary over several orders of magnitude, which may more precisely capture system dynamics than NODEs.

Interestingly, if data cannot be sampled or combined so that the extrema are present in a single experiment, it may still be possible to capture the extrema by leveraging domain knowledge. One way to encode knowledge would be to manipulate the structure of the Neural Network such as its activation function or layer connections. A related method is to include known mechanistic terms in the DE when training the NODEs as done in (Psichogios et al., 1992) and more recently in (Rackauckas et al., 2020; Sorourifar et al., 2023). A third avenue for influencing the NODE interpolation is to impose constraints on the predicted profile (for example, see (Wilson et al., 2017)). None of these methods were pursued in this work in part to demonstrate the generalizability of NODE interpolation even when no such knowledge is available. Whether any of these methods become generalizable will depend on how suitable they are to automation.

As mentioned earlier, when a state is entirely unobserved any interpolating method will fail. Overcoming this limitation will likely require incorporating domain knowledge, possibly through similar strategies as proposed for detecting unsampled extrema. Although this work attempts to illustrate the types of datasets amenable to training NODEs under sparse conditions, admittedly the examples primarily offer heuristics. More rigorous metrics that assess the amount of data required to accurately interpolate across experiments would certainly be welcome as these could lead to more efficient designs of dynamic experiments (Georgakis, 2013). Assessing the accuracy of a NODE can be done more quantitatively than done in this work by applying cross-validation metrics to select hyperparameters. However, it should be kept in mind that such cross-validation can be less straightforward for dynamic data than the non-dynamic, algebraic datasets Neural Networks are more typically trained for.

This study presented an approach to automated model selection using NODEs, which successfully reproduced the true model despite the limited data samples. However, the presented approach is encumbered by several limitations, many of which are due to the inherent assumptions of the SINDy framework. For example, SINDy assumes that the true DE is linear with respect to its parameters, the true model terms are included in the candidate library, and that each term contributes to the model dynamics to a similar degree. The lattermost assumption makes the framework inappropriate for identifying models whose parameters vary over large orders of magnitude, common to stiff DE models. Of course, the SINDy framework was used here merely to showcase how increased accuracy of derivative estimates could lead to solving an increasing number of real-world problems, and the accuracy of the NODE and spline interpolations are not dependent on the limitations of the SINDy framework.

In addition, it is important to note that we have not performed a comparison between the proposed "indirect" NODE method of fitting ODEs from estimated derivatives and the traditional "direct" methods used in ODE literature for parameter estimation of ODEs. A "direct" approach is one that considers the functional form of the ODE to be known, or at least fixed, so that the ODE model can inform numerical integration, collocation schemes or basis functions while fitting the ODE parameters. These direct approaches include nonlinear least squares (NLS) (Hemker, 1972; Bard, 1970; Benson, 1979; Li et al., 2005), principle differential analysis (Ramsay et al., 2007), and direct Bayesian (Huang et al., 2006) and Gaussian Process-based methods (Wenk et al., 2020; Lorenzi et al., 2018; Wang et al., 2014). Physics-informed approaches (Raissi et al., 2017; Sun et al., 2021), which have attracted much interest in recent years, may also be considered direct approaches whenever the PDE/ODE to be fitted is fixed during parameter estimation.

It is well known that these "direct" methods will be more accurate than the "indirect" 2-step approach employed in our work, wherein the latter first estimates derivatives by a data-driven model and then uses those derivatives in the second step to estimate ODE parameters without numerically discretizing the ODE (Varah, 1982). Thus, when computationally tractable, a direct ODE fitting is generally preferred over indirect methods. However, it is also well-known that the 2-step approach can be far more computationally tractable (Varah, 1982). Specifically, these computational gains are most evident when the ODEs to be fit are highly nonlinear with respect to their parameters or when numerous ODEs need to be fit (i.e., a model identification problem). Our previous work explored the former case whereas we highlight the latter case in the current submission. To see a concise comparison between direct and indirect approaches when the latter is better suited, readers may refer to our previous work (Bradley et al., 2021). Because of the opposite advantages of the direct and indirect approaches, they are generally appropriate for different classes of problems and comparing the two would only prove what is already known in the literature—that one method is more accurate and the other more computationally efficient. Thus, in this work, we focus exclusively on comparing our method only with other indirect approaches.

Indeed, the choice to compare NODEs with splines was strategic, as several leading algorithms adopt splines as their basis including kernels smoothing (Dai et al., 2022) and physics informed (Sun et al., 2021) methods, and are thus expected to perform similarly relative to NODEs as splines did in this work. Gaussian Processes (GPs) are also a leading method for interpolation and derivative estimation of data. However, since time-dependent GPs share many similarities to time-dependent, algebraic Neural Networks, which we have previously showed were inferior derivative estimators to NODEs (Bradley et al., 2021), a comparison with GPs we hypothesize will yield similar conclusions. In short, a comparison will all these methods is beyond the scope of this work. However, based on the results in this study one can hypothesize that so far as these indirect methods rely on splines or some other time-dependent model as the basis for interpolation without fixing the ODE structure, the NODE will offer superior results for two-step indirect ODE fitting if data is thinly measured across multiple experiments.

Finally, this work focuses on proving the superior accuracy of NODEs, acknowledging that in other categories, such as interpretability and computational efficiency, NODEs are not expected to offer an advantage over alternative data-driven techniques. Due to the need to integrate the NODE during training, fitting the NODE will require many more function calls during training than most interpolating models. Moreover, during training, the NODE has the potential to converge to a local minimum or diverge on account of poor parameter updating. These issues can be remedied either by breaking up the integration interval in the former case, and in the latter case through testing several hyperparameters of the optimizer and NODE structure. This hyperparameter tuning would invariably increase training time. Nevertheless, the additional computation required to train NODEs can be worthwhile if it leads to more accurate predictions and enables the modeler to solve problems not solvable with faster, but less accurate interpolating methods. Software that automates numerical integration of NODEs are becoming increasingly available (Rackauckas et al., 2020; Merkelbach et al., 2022). As NODE architectures evolve (Rubanova et al., 2019; Kidger et al., 2020), so too are evolving methods to accelerate their training (Cai et al., 2023; Bonnaffé et al., 2023). Future software that enables integrating NODEs over multiple trajectories in parallel would be of great value in accelerating training of NODEs.

7. Conclusions

In conclusion, Neural Differential Equations have successfully been demonstrated to infer system derivatives when data is sparse, noisy, and spread across multiple experiments, including cases when algebraic interpolating methods failed. To communicate the generalizability of the approach, several modes of sparsity were explored, and NODEs were shown to interpolate well the system response provided the extrema were represented in one of the system experiments. In addition, it was shown that the interval of NODE integration becomes an important hyperparameter when being trained on data that is noisy, nonlinear, and sparse. Finally, the method was shown to be sufficiently accurate to discover the mechanistic model generating the system dynamics, especially when multiple experiments of data were available for training. It is anticipated that the methods described in this work will help automate difficult tasks in data analysis and inference of sparse dynamic systems that are beyond the reach of traditional inference methods.

8. Notes

The authors declare no competing financial interest.

CRediT authorship contribution statement

William Bradley: Conceptualization, Methodology, Validation, Investigation, Formal analysis, Writing – original draft, Writing – review

& editing. **Ron Volkovinsky:** Methodology, Validation, Investigation, Writing – original draft. **Fani Boukouvala:** Conceptualization, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, 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

No data was used for the research described in the article.

Acknowledgements

WB and FB gratefully acknowledge funding received from RAPID/NNMI Grant DE-EE0007888-09-03, Georgia Tech start-up grant and NSF CBET grants (1336386 and 1944678).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.engappai.2023.107611.

References

Aguilera, A.M., Aguilera-Morillo, M.C., 2013. Comparative study of different B-spline approaches for functional data. Math. Comput. Model. 58 (7), 1568–1579.
Alexandrov, T., et al., 2012. A review of some modern approaches to the problem of trend extraction. Econom. Rev. 31, 593–624.
Aravkin, A., et al., 2017. Generalized kalman smoothing: modeling and algorithms. Automatica 86, 63–86.
Atluri, G., Karpatne, A., Kumar, V., 2018. Spatio-temporal data mining: a survey of problems and methods. ACM Comput. Surv. 51 (4), 83.
Bard, Y., 1970. Comparison of gradient methods for the solution of nonlinear parameter estimation problems. SIAM J. Numer. Anal. 7 (1), 157–186.
Belytschko, T., et al., 1996. Meshless methods: an overview and recent developments. Comput. Methods Appl. Mech. Eng. 139 (1), 3–47.
Benson, M., 1979. Parameter fitting in dynamic models. Ecol. Model. 6 (2), 97–115.
Boker, S.M., et al., 2010. Generalized local linear approximation of derivatives from time series. In: Statistical Methods for Modeling Human Dynamics: an Interdisciplinary Dialogue. Routledge/Taylor & Francis Group, New York, NY, US, pp. 161–178.
Bonnaffé, W., Coulson, T., 2023. Fast fitting of neural ordinary differential equations by Bayesian neural gradient matching to infer ecological interactions from time-series data. Methods Ecol. Evol. 14 (6), 1543–1563.
Boor, C.d. A., 1978. Practical Guide to Splines. Applied Mathematical Sciences.
Bradley, W., Boukouvala, F., 2021. Two-Stage Approach to Parameter Estimation of Differential Equations Using NODEs. Industrial & Engineering Chemistry Research.
Brunton Steven, L., Proctor Joshua, L., Kutz, J.N., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. USA 113 (15), 3932–3937.
Butterworth, S., 1930. On the theory of filter amplifiers. Experimental Wireless and the Wireless Engineer 7, 536–541.
Cai, L., et al., 2023. Accelerating neural-ODE inference on FPGAs with two-stage structured pruning and history-based stepsize search. In: Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Association for Computing Machinery, Monterey, CA, USA, pp. 177–183.
Chartrand, R., 2017. Numerical differentiation of noisy, nonsmooth, multidimensional data. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP).
Chen, R.T.Q., et al., 2018. Neural Ordinary Differential Equations. arXiv e-prints.
Chen, R.T.Q., Amos, B., Nickel, M., 2021. Learning Neural Event Functions for Ordinary Differential Equations. ArXiv, 03902 abs/2011.
Chen, X., et al., 2022. Local matrix feature-based kernel joint sparse representation for hyperspectral image classification. Rem. Sens. 14 (17), 4363.
Dai, X., Li, L., 2022. Kernel ordinary differential equations. J. Am. Stat. Assoc. 117 (540), 1711–1725.
Deboeck, P.R., 2020. Empirical bayes derivative estimates. Multivariate Behav. Res. 55 (3), 382–404.
Dua, V., 2011. An Artificial Neural Network approximation based decomposition approach for parameter estimation of system of ordinary differential equations. Comput. Chem. Eng. 35 (3), 545–553.
Eilers, P.H.C., Marx, B.D., 2010. Splines, knots, and penalties. WIREs Computational Statistics 2 (6), 637–653.
Forrester, A.I.J., Keane, A.J., 2009. Recent advances in surrogate-based optimization. Prog. Aero. Sci. 45 (1), 50–79.
Gardner, E.S., 2006. Exponential smoothing: the state of the art—Part II. Int. J. Forecast. 22 (4), 637–666.
Gauthier, J., Wu, Q.V., Gooley, T.A., 2020. Cubic splines to model relationships between continuous variables and outcomes: a guide for clinicians. Bone Marrow Transplant. 55 (4), 675–680.
Georgakis, C., 2013. Design of dynamic experiments: a data-driven methodology for the optimization of time-varying processes. Ind. Eng. Chem. Res. 52 (35), 12369–12382.
Härdle, W.K., et al., 1997. A review of nonparametric time series analysis. Int. Stat. Rev. 65, 49–72.
Harrell, F., 2015. Regression Modeling Strategies: with Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis.
Hemker, P., 1972. Numerical methods for differential equations in system simulation and in parameter estimation : (Analysis and simulation of biochemical systems. In: Proc. Of the 8th FEBS Meeting, Amsterdam, 1972. Stichting Mathematisch Centrum, pp. 59–80.
Huang, Y., Liu, D., Wu, H., 2006. Hierarchical bayesian methods for estimation of parameters in a longitudinal HIV dynamic system. Biometrics 62 (2), 413–423.
Hyndman, R., 2010. Moving Averages, pp. 866–869.
Kaheman, K., Kutz, J.N., Brunton, S.L., 2020. SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics. Proc. R. Soc. A 476 (2242), 20200279.
Kaheman, K., Brunton, S.L., Kutz, J.N., 2022. Automatic differentiation to simultaneously identify nonlinear dynamics and extract noise probability distributions from data. Mach. Learn.: Sci. Technol. 3 (1), 015031.
Kaiser, E., Kutz, J.N., Brunton, S.L., 2018. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. Proc. R. Soc. A 474 (2219), 20180335.
Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82 (1), 35–45.
Kaptanoglu, A., et al., 2022. PySINDy: a comprehensive Python package for robust sparse system identification. J. Open Source Softw. 7, 3994.
Kidger, P., et al., 2020. Neural Controlled Differential Equations for Irregular Time Series. ArXiv, 08926. abs/2005.
Lepot, M., Aubin, J.-B., Clemens, F.H.L.R., 2017. Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment. Water 9 (10), 796.
Li, Z., Osborne, M.R., Prvan, T., 2005. Parameter estimation of ordinary differential equations. IMA J. Numer. Anal. 25 (2), 264–285.
Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. Math. Program. 45 (1), 503–528.
Liu, B., Müller, H.-G., 2009. Estimating derivatives for samples of sparsely observed functions, with application to online auction dynamics. J. Am. Stat. Assoc. 104 (486), 704–717.
Lorenzi, M., Filippone, M., 2018. Constraining the dynamics of deep probabilistic models. In: International Conference on Machine Learning. PMLR.
Ma, J., et al., 2019. Model electrical activity of neuron under electric field. Nonlinear Dynam. 95 (2), 1585–1598.
Mangan, N.M., et al., 2019. Model selection for hybrid dynamical systems via sparse regression. Proceedings. Mathematical, Physical, and Engineering Sciences 475.
Mehrkanoon, S., Mehrkanoon, S., Suykens, J.A.K., 2014. Parameter estimation of delay differential equations: an integration-free LS-SVM approach. Commun. Nonlinear Sci. Numer. Simulat. 19 (4), 830–841.
Merkelbach, K., et al., 2022. HybridML: open source platform for hybrid modeling. Comput. Chem. Eng. 160, 107736.
Peng, J., et al., 2022. Low-rank and sparse representation for hyperspectral image processing: a review. IEEE Geoscience and Remote Sensing Magazine 10 (1), 10–43.
Perperoglou, A., et al., 2019. A review of spline function procedures in R. BMC Med. Res. Methodol. 19 (1), 46.
Psichogios, D.C., Ungar, L.H., 1992. A hybrid neural network-first principles approach to process modeling. AIChE J. 38 (10), 1499–1511.
Rackauckas, C., et al., 2020. Universal Differential Equations for Scientific Machine Learning. arXiv e-prints, 04385. arXiv:2001.
Rahardja, D., 2020. Statistical methodological review for time-series data. J. Stat. Manag. Syst. 23 (8), 1445–1461.
Raissi, M., Perdikaris, P., Karniadakis, G.E., 2017. Physics Informed Deep Learning (Part II): Data-Driven Discovery of Nonlinear Partial Differential Equations. ArXiv, p. 10566 abs/1711.
Ramsay, J.O., et al., 2007. Parameter estimation for differential equations: a generalized smoothing approach. J. Roy. Stat. Soc. B 69 (5), 741–796.
Rico-Martínez, R., et al., 1992. DISCRETE- vs. CONTINUOUS-TIME nonlinear signal processing of Cu electrodissolution data. Chem. Eng. Commun. 118 (1), 25–48.
Rubanova, Y., Chen, R.T., Duvenaud, D., 2019. Latent Odes for Irregularly-Sampled Time Series arXiv preprint arXiv:1907.03907.
Rudin, L.I., Osher, S., Fatemi, E., 1992. Nonlinear total variation based noise removal algorithms. Phys. Nonlinear Phenom. 60 (1), 259–268.
Savitzky, A., Golay, M.J.E., 1964. Smoothing and differentiation of data by simplified least squares procedures. Anal. Chem. 36 (8), 1627–1639.
Schafer, R.W., 2011. What is a savitzky-golay filter? [Lecture notes]. IEEE Signal Process. Mag. 28 (4), 111–117.
Sorourifar, F., et al., 2023. Physics-enhanced neural ordinary differential equations: application to industrial chemical reaction systems. Ind. Eng. Chem. Res. 62 (38), 15563–15577.
Sun, L., et al., 2017. Improving flexibility of multivariate spline model structures for aerodynamic modeling. J. Aero. Eng. 30 (5), 04017044.

Sun, F., Liu, Y., Sun, H., 2021. Physics-informed Spline Learning for Nonlinear Dynamics Discovery. IJCAI.
Swain, P.S., et al., 2016. Inferring time derivatives including cell growth rates using Gaussian processes. Nat. Commun. 7 (1), 13766.
Thebelt, A., et al., 2022. Maximizing information from chemical engineering data sets: applications to machine learning. Chem. Eng. Sci. 252, 117469.
van Breugel, F., Kutz, J.N., Brunton, B.W., 2020. Numerical differentiation of noisy data: a unifying multi-objective optimization framework. IEEE Access 8, 196865–196877.
Varah, J.M., 1982. A spline least squares method for numerical parameter estimation in differential equations. SIAM J. Sci. Stat. Comput. 3 (1), 28–46.
Virtanen, P., et al., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17 (3), 261–272.
Wang, Y., Barber, D., 2014. Gaussian processes for Bayesian estimation in ordinary differential equations. In: Proceedings of the 31st International Conference on International Conference on Machine Learning, vol. 32. JMLR.org, Beijing, China. II–1485–II–1493.
Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. Journal of Big Data 3 (1), 9.
Wenk, P., Abbati, G., Osborne, M.A., Schölkopf, B., Krause, A., Bauer, S., 2020, April. Odin: Ode-informed regression for parameter and state inference in time-continuous dynamical systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, pp. 6364–6371. No. 04.
Wilson, Z.T., Sahinidis, N.V., 2017. The ALAMO approach to machine learning. Comput. Chem. Eng. 106, 785–795.
Word, D.P., et al., 2012. A nonlinear programming approach for estimation of transmission parameters in childhood infectious disease using a continuous time model. J. R. Soc. Interface 9 (73), 1983–1997.
Xie, W., et al., 2019. An investigation of the nonlinear dynamic response of a flexible pipe undergoing vortex-induced vibrations and conveying internal fluid with variable-density. Ocean Eng. 183, 453–468.
Zhuang, F., et al., 2021. A comprehensive survey on transfer learning. Proc. IEEE 109, 43–76.