A 2-stage Approach to Parameter Estimation of Differential Equations using Neural ODEs

Authors:

William Bradley, Fani Boukouvala

Date Submitted: 2021-11-07

Keywords: Neural ODEs, Neural-Networks, parameter estimation, Nonlinear programming

Abstract:

Modeling physio-chemical relationships using dynamic data is a common task in fields throughout science and engineering. A common step in developing generalizable, mechanistic models is to fit unmeasured parameters to measured data. However, fitting differential equation-based models can be computation intensive and uncertain due to the presence of nonlinearity, noise, and sparsity in the data, which in turn causes convergence to local minima and divergence issues. This work proposes a merger of Machine Learning (ML) and mechanistic approaches by employing ML models as a means to fit nonlinear mechanistic ODEs. Using a two-stage indirect approach, Neural ODEs are used to estimate state derivatives, which are then used to estimate the parameters of a more interpretable mechanistic ODE model. In addition to its computational efficiency, the proposed method demonstrates the ability of Neural ODEs to better estimate derivative information than interpolating methods based on algebraic data-driven models. Most notably, the proposed method is shown to yield accurate predictions even when little information is known about the parameters of the ODE equations. The proposed parameter estimation approach is believed to be most advantageous when the ODE to be fit is strongly nonlinear with respect its unknown parameters.

Record Type: Postprint

Submitted To: LAPSE (Living Archive for Process Systems Engineering)

Citation (overall record, always the latest version):	LAPSE:2021.0802
Citation (this specific file, latest version):	LAPSE:2021.0802-1
Citation (this specific file, this version):	LAPSE:2021.0802-1v1

A 2-stage Approach to Parameter Estimation of Differential Equations using Neural ODEs

William Bradley, Fani Boukouvala*

School of Chemical & Biomolecular Engineering, Georgia Institute of Technology

311 Ferst Dr., N.W. Atlanta, GA, 30332-0100 USA

Abstract

Modeling physio-chemical relationships using dynamic data is a common task in fields throughout science and engineering. A common step in developing generalizable, mechanistic models is to fit unmeasured parameters to measured data. However, fitting differential equation-based models can be computation intensive and uncertain due to the presence of nonlinearity, noise, and sparsity in the data, which in turn causes convergence to local minima and divergence issues. This work proposes a merger of Machine Learning (ML) and mechanistic approaches by employing ML models as a means to fit nonlinear mechanistic ODEs. Using a two-stage indirect approach, Neural ODEs are used to estimate state derivatives, which are then used to estimate the parameters of a more interpretable mechanistic ODE model. In addition to its computational efficiency, the proposed method demonstrates the ability of Neural ODEs to better estimate derivative information than interpolating methods based on algebraic data-driven models. Most notably, the proposed method is shown to yield accurate predictions even when little information is known about the parameters of the ODE equations. The proposed parameter estimation approach is believed to be most advantageous when the ODE to be fit is strongly nonlinear with respect its unknown parameters.

Keywords: Neural ODEs; Neural-Networks; parameter estimation; Nonlinear programming; Time series data

* Correspondence for this paper: fani.boukouvala@chbe.gatech.edu

1. Introduction

A truly optimal workflow for model-building is one that properly leverages available data resources, domain-knowledge resources, and computational resources. The first two of these can be maximized through mechanistic approaches, where hypotheses based on first principles knowledge are used to formulate mechanistic models, which can be fit and validated with fewer experiments than purely empirical approaches. However, sufficient first principles understanding is often lacking, and this hinders the formulation of accurate mechanistic models. Moreover, when they are available, accurate models tend to require excessive compute time for fitting of their parameters, simulation, and optimization. Data-driven models, on the other hand, tend to be computationally efficient, but require either too much data or have too little interpretability to solve many scientific problems.[1] Due to the contrasting yet complementary strengths of data-driven and mechanistic approaches to model-building, many authors have sought to combine these paradigms in ways that increase interpretability and lower data requirements.[2-4] Readers interested in a comparison of data-driven, mechanistic, and hybrid approaches to model-building are encouraged to consult recent surveys.[5-7]

Ultimately, mechanistic models offer the greatest interpretability and thus methods that efficiently regress parameters of mechanistic models, especially those formulated as differential algebraic equations, would facilitate vetting of model formulations when there is a high degree of uncertainty in the parameter values. Yet despite decades of increasing computational power, fitting and simulation of differential equation (DE) models remains computationally challenging for many systems of interest. The primary methods for fitting nonlinear ODE models include 'direct' approaches such as the nonlinear least squares (NLS),[8-11] principle differential analysis,[12-14] and direct Bayesian[15, 16] and Gaussian Process-based methods.[17-21] Following the nomenclature of [22] the direct NLS procedures can be further divided into sequential and simultaneous approaches.

Also known as the constrained or non-feasible path approach, the simultaneous approach avoids integrating the differential equations (DEs) repeatedly. For example, multiple shooting is a simultaneous approach which breaks up the state trajectory into linked stages or intervals, parameterized by polynomial basis functions.[23-26] Alternatively, using collocation methods the state profiles are approximated using polynomials connected on finite elements.[11, 27] In these algebraic nonlinear programs (NLPs), the parameters of the polynomial functions are solved simultaneously along with the parameters of the differential equations. Especially the latter approach is frequently used when solving a boundary value or optimal control problem as it offers a straightforward way to incorporate inequality or path constraints and can be solved even when initial parameter guesses cause the differential equations to diverge upon

integration. However, due its large formulation, it is less frequently used to solve initial value problems (IVPs).[22]

For the unconstrained, or sequential, NLS, the DEs are integrated repeatedly during training. The forward solution from integration is used to calculate the error between the DE model predictions and true data. The gradients of the computed error or loss function are calculated to give the optimizer the direction parameters should be updated at each iteration. Unconstrainted NLS is the version of the direct approach used in this work. A comparison of strategies for integration and parameter estimation using the direct approach can be found in [28].

However, the direct approach has several weaknesses, including poor rate of convergence for highly nonlinear systems and potential to converge to local minima.[29, 30] This can be ameliorated somewhat via multiple shooting methods, which may mitigate divergence when parameter values are far from their correct values. However, if a good initial guess of model parameters is unavailable, integration of the differential equations, especially for stiff systems, may still be infeasible. Although Bayesian approaches have the potential to overcome some of the local minima issues of direct NLS methods, the direct Bayesian methods are beholden to the same divergence issues as direct NLS methods since they involve integration of the original DEs.[31] In addition, as noted in [31] obtaining the posterior distribution for fully Bayesian and Gaussian Process-based methods often relies on sampling via Metropolis Hastings-type algorithms, which can be impractical for high dimensional problems. Finally, partial differential analysis (PDA) can be unattractive for similar reasons as the constrained NLS scheme proposed by [27] since both create a large optimization problem with a large number of unknowns, which may be challenging to solve.

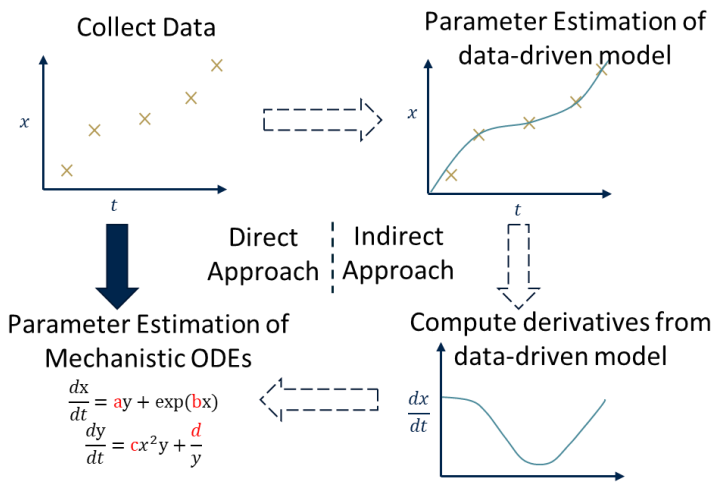

Figure 1. Depiction of the direct vs indirect (i.e. 2-stage) approach to parameter estimation.

Alternatively, a far less computationally costly method is the 2-stage, or indirect, approach to parameter estimation.[32-35] In the 2-stage approach, state measurements are interpolated (i.e., smoothed) via data-driven models. Next, the data-driven model is differentiated to estimate system derivative information at sampling times. Derivatives can also be inferred without interpolation, though with limited accuracy, from numerical approximations. Lastly, using the derivative and state estimates of the data-driven model, one can set up an algebraic nonlinear programming (NLP) problem to fit the parameters. We note here than the 2-stage indirect approach should not be confused with the indirect approach in control theory based on Pontryagin's Maximum Principle.[36, 37] In this work, the 2-stage indirect approach seeks to find the parameter values of a differential equation (DE) without integrating the original DE during training. By bypassing the integration of the original DEs, 2-stage methods tend to give significant compute advantages over direct approaches. Initially, the beta-splines were suggested as the data-driven interpolator for 2-stage methods for ODE parameter estimation.[33] Since then, authors have implemented the 2-step approach using other data-driven models, including support vector machines [38] and neural networks.[39] An illustration of the steps in the direct and indirect approaches is depicted in Figure 1.

Despite its compute advantages, traditional 2-stage approaches suffer from limited accuracy for real systems and at best are used to provide an initial guess for parameter values.[40] This is because, especially when data is noisy or contains outliers, data-driven models used to interpolate data tend to yield low quality derivative estimates, which reduces the quality of the parameter estimates obtained when solving the NLP.[41] Furthermore, it is often the desire to experimentally explore a system using multiple experimental runs with varying conditions, yet none of the derivative estimation techniques currently proposed have a straightforward way to account for multiple batches of data with a single data-driven

model. Each set of conditions would require estimation with a different data-driven model, increasing the data-burden and complexity further. Thus, to be useful for real systems, the data-driven model in the 2-stage approach needs to accurately capture derivatives for nonlinear dynamic systems, with minimal compute time, in the case of limited and/or noisy data, and possibly with data spread across system runs collected under different process conditions.

Among options for data-driven models, Neural Networks (NNs) are an attractive choice, as they have long been used to approximate nonlinear algebraic relationships due to their universal approximation potential.[42] Methods to model dynamic systems by applying NNs to approximate relationships within differential equations go as far back as the early 90's.[43-45] More recently, 'neural ordinary differential equations' (NODEs)[46] have been integrated with software with pervasive automatic differentiation to accelerate fitting to spatio-temporal data for a variety of systems.[47] By defining NNs to predict the system derivatives directly, the NODE captures both state and derivative information during NN training. This could potentially enable the NODE to better capture the curvature in the response of dynamically evolving data than algebraic data-driven models that don't consider derivative information during training. A conceptual depiction of this hypothesized advantage is illustrated in the abstract figure at the beginning of this article.

This work proposes a novel approach to address the shortcomings of a 2-stage approach through the application of Neural Ordinary Differential Equations (NODEs) as the data-driven component within an indirect approach framework. This work also proposes a novel integration scheme for fitting Neural ODEs. As the original ODE equations often have physically interpretable, albeit unknown, parameters values, they are herein referred to as the mechanistic ODE or simply the mechanistic model. This will aid in differentiating it from the more data-driven Neural ODE. In this work we set out to prove that 1) Neural ODEs generally outperform purely data-driven NN models at estimating 1st order state derivatives of dynamic data and 2) estimating mechanistic ODE parameters via a 2-stage approach abetted by Neural ODEs can be competitive computationally and more flexible than direct approaches to fitting DE models. To achieve this end, three cases studies are examined based on the Lotka-Volterra equations, the dehydrogenation of ethylbenzene, and penicillin production via cell culture fermentation. Different aspects of the method's flexibility are illustrated via each of these case studies.

The remainder of this paper is structured as follows. In Section 2, the two parameter fitting steps of the 2-stage approach are mathematically formulated and a general outline of the 2-stage approach is provided. The performance and flexibility of the approach is explored is the Results (Section 3) through the lens of

three case studies. A discussion of the results can be found in Section 4. Finally, Section 5 concludes and identifies opportunities for further investigation.

2. Methods

As illustrated previously in Figure 1, the 2-stage approach fits the parameters of the mechanistic model by solving 2 separate regression problems. In the first stage, the parameters of the data-driven model are fitted using the original measurement data. In the second stage, the parameters of the mechanistic ODE are found using the state and derivative estimates of the data-driven model. The novel implementation of the 2-stage approach proposed in this work (see Figure 2) fits a Neural ODE as the data-driven model. This is done by first solving the following regression problem:

$$\min \sum (x_{k,j,meas} - x_{k,j,pred})^2 + \lambda \sum w^2 \tag{1}$$

$$s.t.\ \frac{dx_k}{dt} = NN(x_k, w) \tag{2}$$

Here, K state variables x_k, where $k = 1, \dots, K$, are measured and predicted at time points j, where $j = 1, \dots, J$, by integrating a NODE with respect to independent variable t. Neural network parameters w are fitted to minimize an objective function equal to the sum of squared errors between the model prediction and measured state data and a regularization term. Due to the large number of parameters in the Neural Network, a regularized penalty term of the weights is added to the objective function multiplied by a hyperparameter λ. Once the NODE is trained, derivative estimates are obtained by integrating the trained NODE from time $t_0 = 0$ to a final time t_f of measured data using the same process conditions of the measured data. State predictions of the NODE are used to simulate derivatives at times where measured data is available. For the second stage of the 2-stage approach, a nonlinear program (NLP) is formulated as in Eq. 3 and 4 to find the parameters of the original mechanistic ODE.

$$\min \sum \left(\frac{dx_{j,k,NODE}}{dt} - \frac{dx_{j,k,MM}}{dt}\right)^2 \tag{3}$$

$$s.t.\ \frac{dx_{j,k,MM}}{dt} = f(x_{j,k,NODE}, p) \tag{4}$$

To solve this formulation, the parameters p of the mechanistic ODE model $f(x, p)$ are found by minimizing the sum of squared differences between the derivatives predicted by the NODE and the

derivatives predicted by the mechanistic model in the NLP. Alternatively, the objective to minimize could be the sum of squared errors of the states. Note that all equations in the NLP formulation are purely algebraic and no integration is involved. Further, it is worth emphasizing that the state and state derivative values ($x_{j,k,NODE}$ and $\frac{dx_{j,k,NODE}}{dt}$, respectively) used to solve the NLP are estimates from the fitted NODE, not the original measurement data. In order to test the limits of this approach, it is assumed minimal prior knowledge of the true parameter values was available. Thus, all parameters are initialized to the same value and given wide bounds when solving the NLP. Technically, since the Neural ODE can be simulated at any time t, additional points could be added to the NLP formulation. However, limiting the number of state/derivative values to the number of measured points was adequate for the purposes of this study. In addition, derivative estimates of the NODE at initial conditions $t = 0$ tended to be poor and were not used when formulating the NLP. For each of the two optimization routines in the 2-stage approach, an appropriate scaling method is used to account for states with differing orders of magnitude. Namely, all state variables were divided by the range of their respective state measurements.

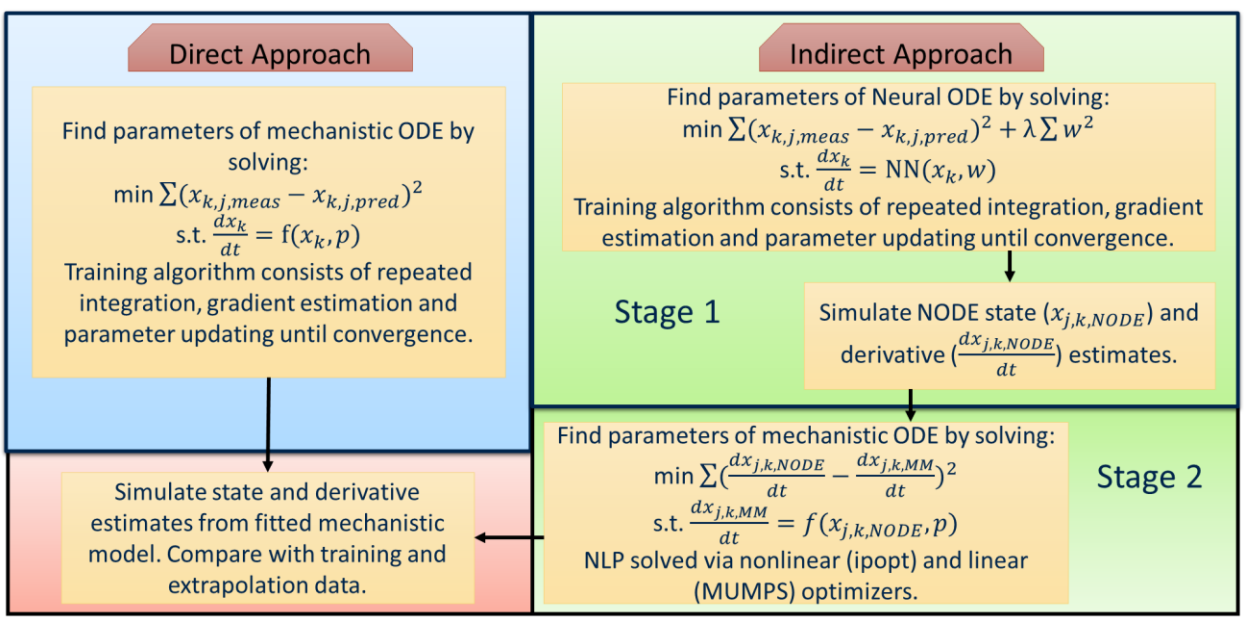

Figure 2. Depiction of steps and software used for training and testing DE models via the direct and NODE-based indirect approaches.

A comparison of steps for the direct and indirect approach is depicted in Figure 2. All Neural Networks were trained using PyTorch,[48] which uses automatic differentiation via the Autograd software package to accelerate gradient calculation and thus parameter estimation. Moreover, all numerical integration whether for the direct or indirect approach was conducted in PyTorch. The quasi-Newton method L-

BFGS was used to train all PyTorch models and all NLP formulations were solved with nonlinear solver IPOPT[49] using linear solver MUMPS,[50] in the Pyomo modeling environment.[51, 52] A neural network with a single hidden layer with a hyperbolic tangent activation function was found to give reasonable accuracy across all case studies. However, as this work also sought to analyze NODE performance across different noise levels, it was considered prudent to fit multiple NODEs for each level of noise, varying parameters of the NODE stage 1 fitting algorithm (also known as hyperparameters) to maximize the generalizability of the trained NODE. Specifically, the hyperparameter tuning was set to include 5, 7, or 10 hidden nodes and the weight of λ in the stage 1 objective function was set to 10E-4, 10E-5, or 10E-6. Using a grid search fitting of all combinations of these hyperparameters, the NODE whose hyperparameters led to the lowest mean squared error between model predictions and noisy training data was selected for the stage 2 regression problem.

A key technical challenge of this work was developing an integration training algorithm that consistently fit an interpolating model to continuous data of arbitrary nonlinearity, sparsity and quality. In addition to structural hyperparameters, some parameters of the optimization solver should be considered. Important hyperparameters were found to be the termination criteria of the Neural ODE training algorithm and the discretization method used. For all cases, the training algorithm was stopped when the objective function ceased to improve by a set tolerance (rtol = 10^{-6}) for more than 10 epochs. The forward Euler method was used to integrate the ODEs—a necessary step to obtain model gradients during training. However, additional modifications of the numerical integration algorithm were found necessary, which are best discussed in the results section and is shown through the Lotka-Volterra case study.

3. Results

Before presenting the results for each case study, a brief introduction and objective of each example is provided here. The Lotka-Volterra study will be used to illustrate key aspects of the NODE regression algorithm as well as differentiate between the behavior of NODEs and mechanistic ODEs. Next, the styrene reaction system will be used to contrast the performance of NODEs with algebraic data-driven models, specifically Algebraic Neural Networks, when estimating system derivatives. This system is also used to demonstrate the indirect approach's ability to estimate parameters for mechanistic ODEs with highly nonlinear terms. Finally, a fermenter system will investigate the performance of NODEs for noisy systems as well as possible adaptions of the NODE indirect approach when domain knowledge is available to inform the interpolating model (i.e., via hybrid modeling). All case studies use the same integration algorithm, but due to their unique features and for the sake of concision, we present different results and highlight different aspects of our approach through each case study.

3.1 Lotka-Volterra Equations

The Lotka-Volterra equations[53, 54] were chosen as a first demonstration of the versatility of the NODE-based 2-stage approach. Commonly known as the predator-prey model, these equations are frequently used to track the interactions between oscillatory populations for a wide variety of systems, including chemical reactions,[55] biological competition,[56] and ecological systems.[57] In addition, these equations are frequently used to test differential equation solution methods (for example, see [27, 28, 32, 33]) due their characteristic nonlinearity and simple formulation. To train the NODE, 20 'measurements' for populations of species x and y were collected within a period t = [0,5] by simulating the Lotka-Volterra ODE model, summarized in the Supporting Information. The task at hand is to fit all the parameters of the mechanistic ODE using the 2-stage approach.

Initially, the NODE was fitted by integrating over a single interval from time t_o = 0 to final time t_f = 5. However, this consistently resulted in the NODE training converging to a local minimum between the min and max values of the state profiles as shown in Figure 3. To overcome this undesired behavior, the training algorithm was modified to integrate the Neural ODE not from a single initial value, but from multiple initial values. Specifically, each timepoint *j* with measured data is used as an initial value (IV) in the integrator, which is integrated forward in time for an arbitrary number of data points n, from t_j to t_{j+n}. Clearly, a balance must be made between the time interval for the forward integration steps, the nonlinearly of the state space, and the quality (i.e., level of noise) of the data. It was decided to fix the total integration to a span of 5 measured points for each initial value, and the number of Euler time steps between measured data was set to 6. For the LV equations with 20 simulated points in the time interval t = [0,5], the smallest Euler step size was $\Delta t = 0.0417$. The improved convergence using the revised integration algorithm can be seen in Figure 3. Due to the improved convergence, this method integrating over overlapping intervals spanning 5 measured points was used to train all NODEs in this work.

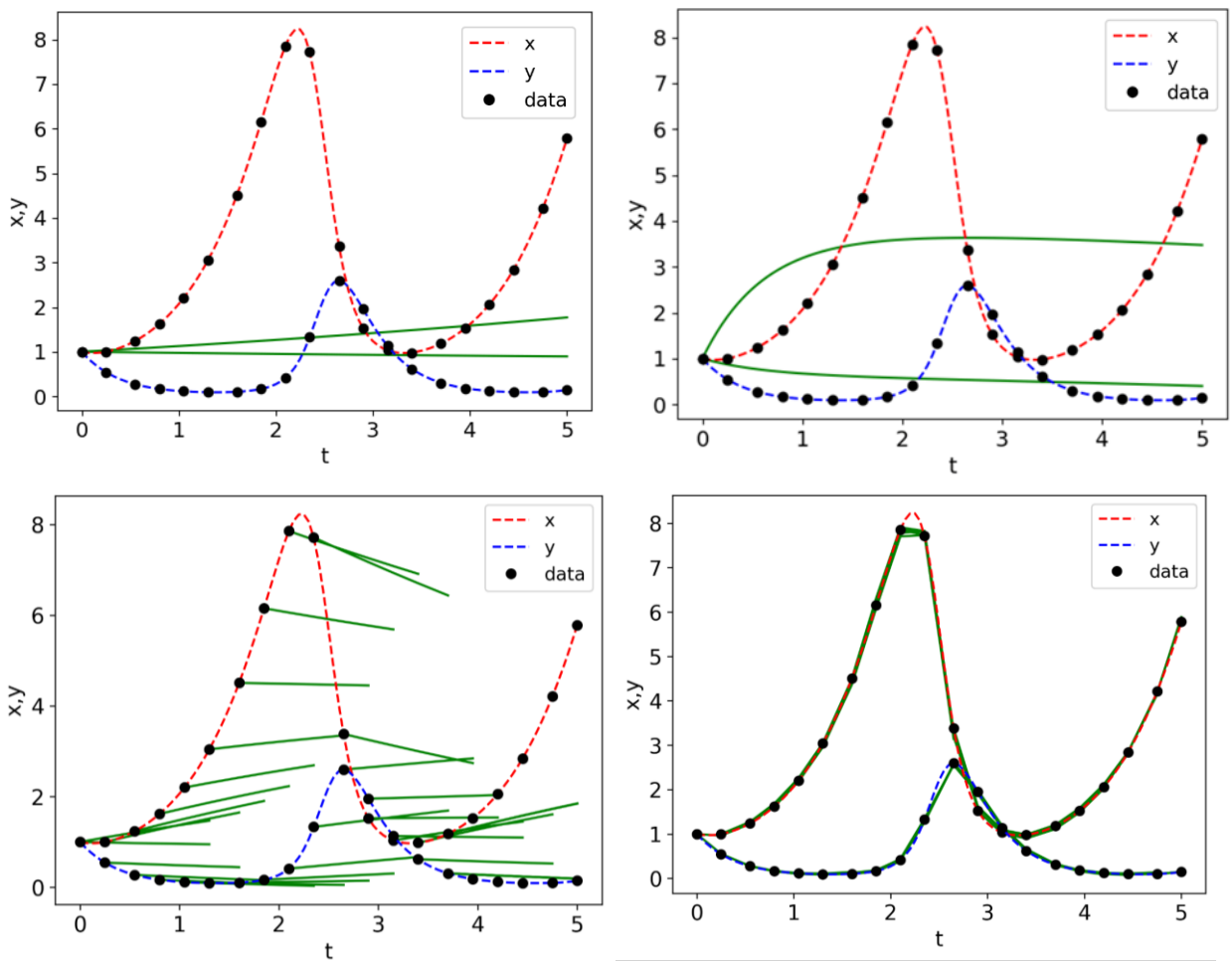

Figure 3. Progression of NODE predictions in green for the Lotka-Volterra system at the beginning (left) and end (right) of training when integrating from a single IV (top) and multiple IVs (bottom).

The use of integration from multiple initial values may appear similar to the multiple shooting approach. However, in general multiple shooting methods, the integration intervals do not overlap; rather, the boundary conditions are optimized with the other model parameters until the final values of one interval are equal to the initial value of the subsequent interval, creating a continuous dynamic solution. In contrast, the integration method applied herein integrates over multiple overlapping intervals, beginning from time points where measured data is available. Although the initial values could be included as trainable parameters, in this work the initial value of each integral is fixed at locations of measured data. The integration scheme also differs from multiple shooting in its fundamental purpose. Whereas the purpose of multiple shooting is to avoid divergence during integration, the motivation for our method is specifically to avoid convergence to local minima when training the NODE. To our best knowledge, this is the first work to propose integrating over overlapping intervals to enable interpolation of dynamic data of arbitrary nonlinearity.

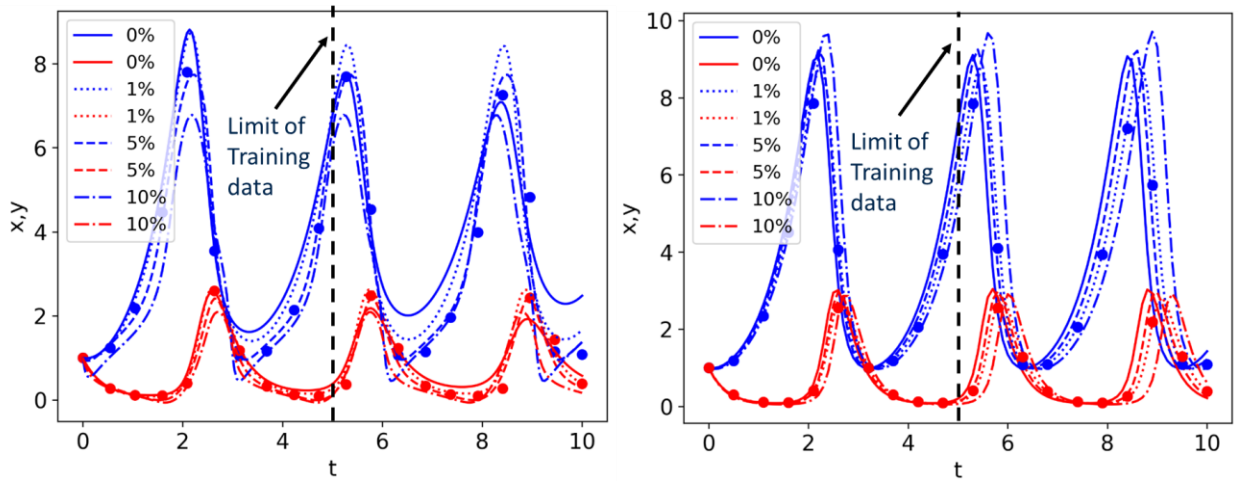

Figure 4. Simulation of fitted Neural ODE (left) and fitted Lotka-Volterra equations (right) when trained on data corrupted by Gaussian-distributed noise equivalent to 0, 1, 5, or 10% of the true data. True data represented by dots. Training data restricted to interval t = [0,5]. Same initial value as training data.

With a properly fitted NODE, the NODE can now be used to fit the mechanistic ODE (see again, stage 2 in Figure 2). Prior to this second fitting problem, the trained NODE is integrated from a single initial value across the entire time trajectory to obtain state and derivative estimates used in the NLP estimating mechanistic parameters. The NLP can then be solved without integrating the mechanistic ODE. It is worth clarifying that NODEs are not the end model in the 2-stage approach. More appropriately, the NODE can be viewed as a data-driven means to a mechanistic end. Due to their data-driven nature, NODEs cannot be expected to offer accurate predictions far beyond the range of training data, despite the fact that they are used to predict derivatives. Rather, the NODE is fitted to obtain system state and derivative estimates for regressing mechanistic differential equations. If properly formulated, the mechanistic model offers the system interpretability and extrapolation properties.

To illustrate this principle, Figure 4 demonstrates the effect of simulating a trained NODE beyond the limit of training data. For this illustration, the NODE was trained on 20 data samples in the interval t = [0,5] corrupted with Gaussian-distributed noise equal to 0, 1, 5 or 10% of the range of the state data, and it is then simulated for a period twice the time interval of the training data. In addition, a mechanistic model is fitted by solving an algebraic NLP using the NODE state and derivative estimates from the training interval t = [0,5], and is then simulated for double this interval. Several principles can be extracted from Figure 4, of which two are highlighted here. First, the Neural ODE predictions are not adversely affected by the addition of a small amount of noise, even improving when the noise added is

small, which may seem counterintuitive. However, this can be explained by the general overfitting properties of Neural Networks (NNs). Numerous previous studies have shown that in many cases NNs tend to generalize less well when data is 'perfect' (i.e., noiseless), suggesting that modelers add noise to the data to discourage overfitting.[58-61] NODEs are essentially Neural Networks that predict the instantaneous change in a system. Thus, they inherit similar overfitting properties of Neural Networks. However, as extrapolation is not required for estimation of the mechanistic ODE, the effects of overfitting on extrapolation is not of serious concern for the NODE indirect approach.

Second, it may appear from Figure 4, based on the case wherein the NODE is trained on 5% noise, that the overfitting issue has been overcome and the NODE can extrapolate competitively with the fitted mechanistic model. The ability of Neural ODEs to capture oscillatory dynamics is congruent with similar studies.[62] However, this behavior is better interpreted as sophisticated pattern-matching rather than rigorous extrapolation. To clarify this claim, we tested the fitted Neural ODE and mechanistic ODE on the case where the initial conditions of the predator-prey system change (see Figure 5). Without retraining the models, the NODE and mechanistic model are simulated assuming a higher initial amount of 'predator' in our system. This time the NODE clearly fails to capture the nuanced interactions between system variables, regardless of the quality of the training data—even predicting physically unrealistic negative values. As a juxtaposition, the correctly parameterized mechanistic model captures the variable interactions with far greater precision. It is the potential for increased interpretability and extrapolation that motivates the final model to be a mechanistic model in the 2-stage approach.

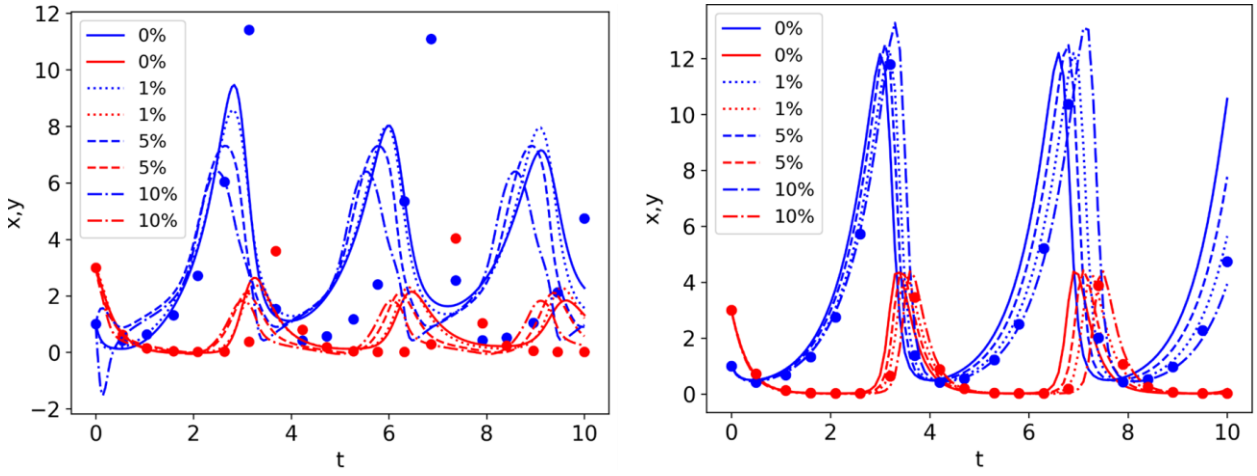

Figure 5. Simulation of fitted NODE (left) and fitted Lotka-Volterra equations (right) when trained on data corrupted by Gaussian-distributed noise equivalent to 0, 1, 5, or 10% of the true data. True data represented by dots. Different initial values from training data.

3.2 Styrene Example

Serving as a second demonstration, the dehydrogenation of the ethylbenzene (EB) to form styrene is modeled in a tubular reactor.[63, 64] The reactor is assumed to operate in plug flow and thus reactant concentrations change only in the axial direction. This system consists of a reversible reaction to the desired products styrene and hydrogen as well as two undesired, irreversible side reactions. Benzene and ethylene are produced in equimolar amounts and are thus assumed to have the same concentration. The same is true of toluene and methane. In total, the 7 chemical species involved in the reaction include ethylbenzene, styrene, hydrogen, benzene, ethylene, toluene, and methane. The stoichiometry of the reaction along with the mechanistic model used to simulate the styrene production process are found in the Supporting Information. To collect training data, the mechanistic model is simulated over a reactor length t = [0,12] meters with initial temperatures in the range of T = [850, 950] Kelvin and an initial ethylbenzene flow rate in the range F_{EB} = [3,5] mol/s, all other species concentrations starting at zero. Six system experiments are simulated with the above inlet conditions and 10 measurements of system states are sampled at equidistant points along the reactor for each experiment for a total of 60 timepoints of available training data.

To motivate the use of NODEs in the two-stage approach, we compared its ability to capture system derivatives with other data-driven models. For the EB system, the Neural Network representing the NODE receives K=6 inputs x_k corresponding to the flowrate of ethylbenzene, styrene, hydrogen, benzene/ethylene, and toluene/methane and temperature. The NODE has 6 outputs corresponding to the instantaneous derivatives of each of the system states. The states predicted by the NODE are obtained by numerically integrating the model with respect to reactor length t.

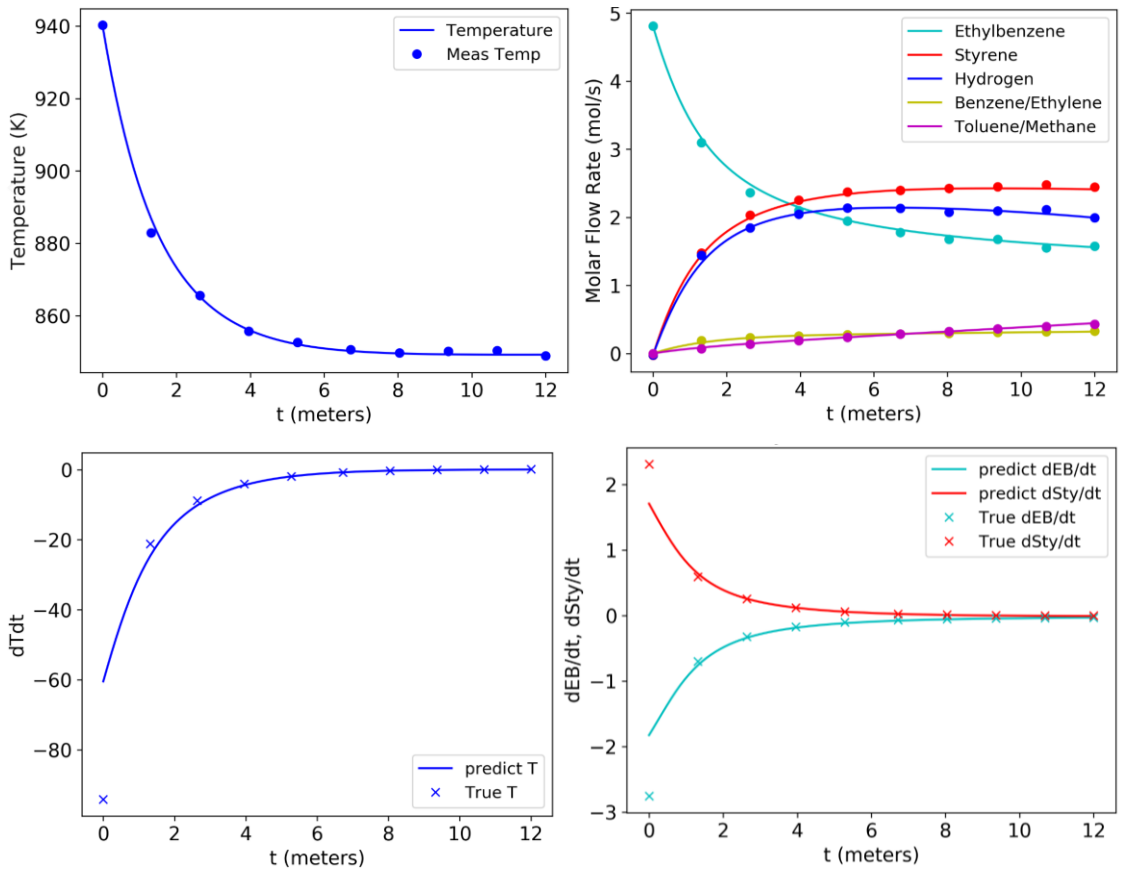

Figure 6. State and state derivative fits of the Neural ODE to styrene system data. Solid lines represent NODE predictions, solid points are training data and 'x' tick marks are the derivatives of the original, noiseless simulation.

As mentioned previously, a major shortcoming of the two-stage approaches found in literature so far, is poor estimation of system derivative information, which leads to poor estimation of mechanistic parameters. To demonstrate the superior performance of the NODE, an Algebraic (i.e., non-dynamic) Neural Network was also fitted, which receives length of reactor t as its only input and outputs the 6 state variables of the EB system (not derivatives). This Algebraic NN (a-NN) can predict state derivatives by computing the gradient of the NN outputs with respect to its input, reactor length. The state variables could also be used as inputs although these did not significantly enhance accuracy of the a-NN estimates. The mathematical equations for the Neural ODE and the a-NN are thus formalized in equation 5 and 6, respectively.

$$\frac{dx_k}{dt} = NN(x_k, w) \tag{5}$$

$$x_k = NN(t, w) \tag{6}$$

Both the Neural ODE and a-NN are trained on a single batch of reaction data (i.e., 10 points along the reactor) using the process conditions outlined in Experiment 1 in Table S.1 in the Supporting Information. A small amount of Gaussian-distributed noise equivalent to 1% of the range of each state variable is added. Depicted in Figure 6 and Figure 7 are the state and derivative estimates of the trained Neural ODE and the a-NN, respectively. Clearly, both data-driven models provide an adequate interpolation of the state data. Yet, when used to predict state derivatives, the Neural ODE estimates are far more reliable. The a-NN visibly fails to capture the derivative profiles despite the state data being corrupted with minimal error (i.e., 1% noise). The simple explanation for this lies in the fact that in the process of integrating the NODE to predict the states, the NODE must accurately predict the derivatives. In contrast, no state derivative information is involved in the training of the a-NN.

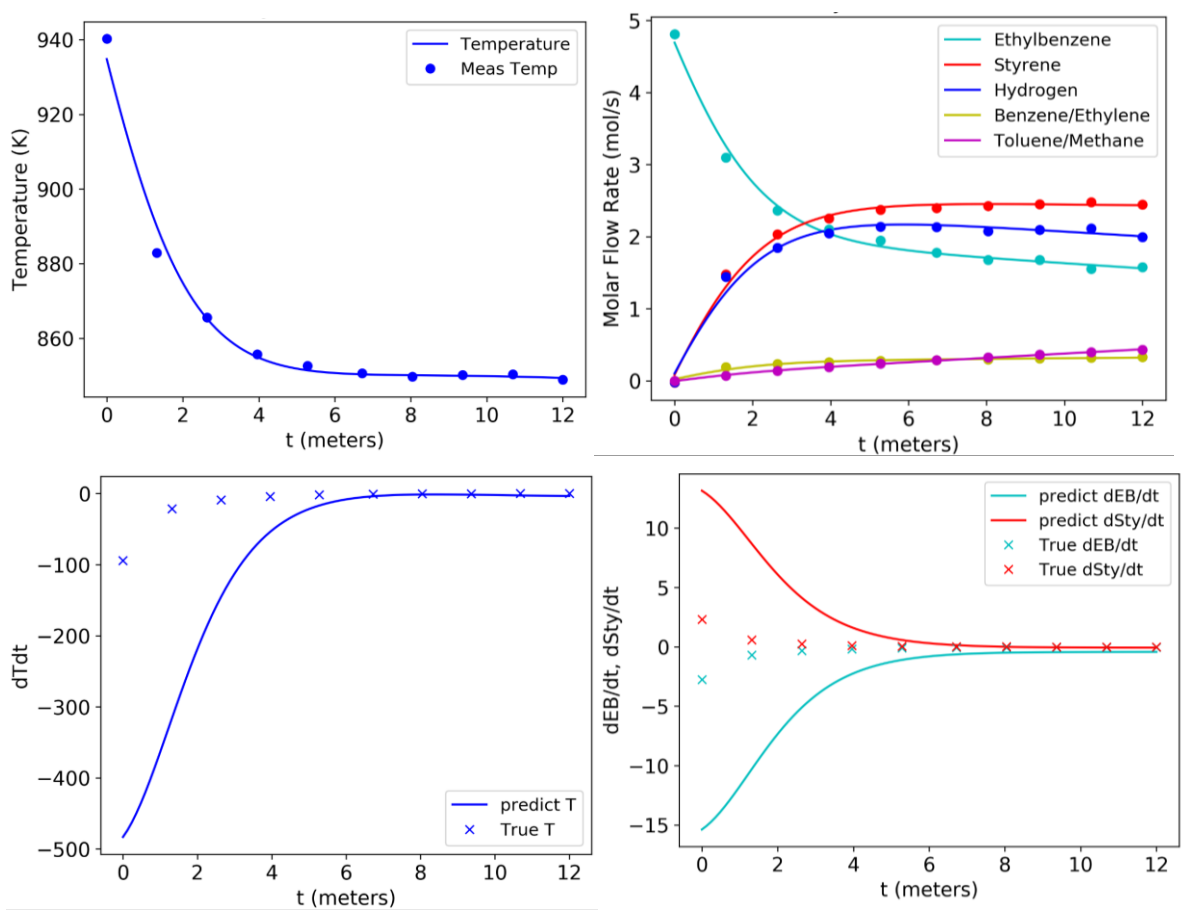

Figure 7. State and state derivative fits of the Algebraic NN to styrene system data. Solid lines represent NN predictions, solid points are training data and 'x' tick marks are the derivatives of the original, noiseless simulation.

The a-NN model was likewise fitted to the state data of the other case studies considered in this work and the predicted derivatives plotted against the true rates, with equally underwhelming results. For the sake of brevity, we surmise that for every system considered herein the NODE model gave more accurate estimates of the state derivatives than a standard a-NN. These results are not surprising in light of previous work, which has shown the importance of using dynamic data-driven models to interpolate dynamic data, rather than their algebraic equivalents.[45]

With the confidence in the NODE's ability to capture system derivative information, we now turn to NODE's ability to estimate the parameter values of the original mechanistic ODE. For this task, measurements from all six process conditions are used to fit the NODE and mechanistic ODE. It was assumed that 3 parameters of the EB model were unknown, namely the frequency factor (FF) of each reaction, all other parameters fixed at their true values. Unknown mechanistic parameters were initialized to a value of 2 prior to regression. To demonstrate the robustness of NODE models to low-quality training data, Gaussian noise was added to the measured data equal to 0, 5, and 10% of the range of the state data.

Table 1. Table of Frequency Factor (FF) Estimates via direct and NODE indirect approaches

	0% Noise	**5% Noise**	**10% Noise**
FF (direct)	[-0.1626, 2.0039, 0.2787]	[-0.1063, 2.0027, 0.3751]	[-0.1020, 2.0025, 0.4575]
FF (indirect)	[-0.1936, 13.0463, 0.16928]	[-0.2622, 12.8374, 0.2061]	[-0.2079, 12.8301, 0.3044]
True Frequency Factor Values: A_1, A_2, A_3 = [-0.08539, 13.2392, 0.2961]			
Initial FF Estimates (Pretraining): A_1, A_2, A_3 = [2.0, 2.0, 2.0]			

Table 1 shows the fits of the three frequency factors using the direct and indirect approaches. Recall that the direct approach requires the repeated integration of the mechanistic ODEs during parameter estimation whereas the indirect approach avoids integrating the mechanistic ODEs in favor of integrating NODEs via the NODE 2-stage approach. Both approaches perform well at estimating the frequency factors for reactions 1 & 3. However, the Neural ODE 2-stage approach consistently provides superior estimates for the frequency factor of reaction 2, even when training data is corrupted with a large amount of noise. The inability of the direct approach to estimate A_2 can be explained in part by the difference in magnitude of the model gradients calculated during training. The initial value of the second parameter is furthest from the true value, resulting in a gradient that is orders of magnitude different from the gradients

computed for the other parameters. This results in a poorly behaved parameter updating algorithm during training. To try to understand the success of the 2-stage approach in overcoming this parameter sensitivity issue, we also tried to solve the stage 2 formulation with L-BFGS rather than formulating it as an NLP and solving it with IPOPT. Briefly, the L-BFGS solver was unable to find the true frequency factor for reaction 2 even when 'perfect' state and derivative values were used in the stage 2 formulation. While the exact cause for the success of the IPOPT-solved NLP form remains under investigation, we hypothesize a possible reason for this behavior is the scaling performed internally by the IPOPT solver enables more accurate convergence. This issue could be resolved with *a priori* scaling or reformulation of the ODE model. However, without good foreknowledge of the true parameter values, such an *ad hoc* approach is not straightforward. In contrast, the NODE approach abetted by an advanced NLP solver offers good estimates of all system parameters without significant prior knowledge of the correct parameter values.

3.3 Penicillin Model

For the final case study used in this work to illustrate the versatility of NODEs, we chose to model the production of penicillin via yeast fermentation. The fermentation has several challenging elements unique to this system. First, modeling the reactor requires incorporating external forcing variables (also known as control or system operating variables), namely the flow rate and substrate concentration of the feed. Moreover, the level of nonlinearity in the system differs significantly between state variables. It is further assumed that none of the 11 parameter values of the original mechanistic ODE are known, posing a serious test to the proposed NODE algorithm. The system equations and process conditions can be found in the Supporting Information. Nine sets of process run conditions are used to generate training data. Assuming 10 data points can be collected from each run, 90 data points are available for training. A depiction of the continuous state profiles of the nine process runs are given for reference in Figure 8.

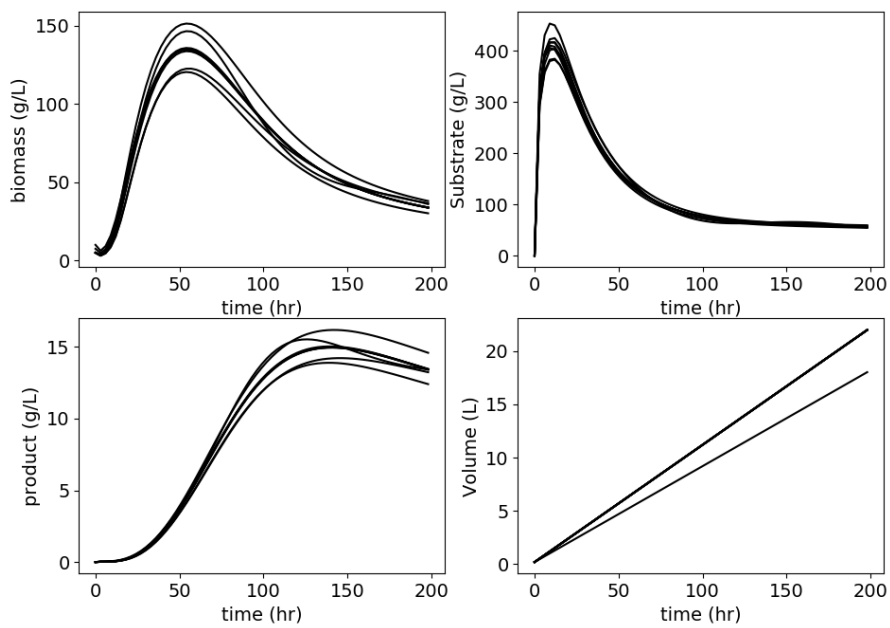

Figure 8. Simulated state profiles from penicillin fermentation ODE model with correct parameters for all 9 experimental batch conditions.

A few options exist for incorporating the forcing variables in the formulation of the Neural ODE. The simplest approach is to modify Equation 6 to include the forcing variables as inputs to the Neural Network.

$$\frac{dx_k}{dt} = NN(x_k, \mathrm{c}, w) \tag{7}$$

With all the state and forcing variables included, the NODE would have 6 inputs, including 2 forcing variables $c = [F, S_f]$ corresponding to the substrate concentration in the feed (S_f) and feed flow rate (F) and 4 state variables (x_k). With respect to outputs, the NODE would predict the derivatives of the 3 state variables biomass (B), substrate (S) and product (P) concentration. However, the addition of forcing variables requires the NODE to learn complex nonlinear relationships with little extra data information since the forcing variables are often constant throughout the process. Not surprisingly, training with all variables resulted in inconsistent and diverging training properties. Alternatively, the size of the Neural Network component of the NODE can be reduced by including mechanistic information in the neural differential equation. Generally speaking, engineering systems have some readily available mechanistic knowledge such as conservation balances that can be combined with data-driven models to create more interpretable models. This is akin to hybrid semi-parametric modeling introduced in the early 90's.[43, 44]

In the case of the fermenter example, the change in volume and the effect on concentration from the feed rate can easily be deduced from a mass balance. This 'hybrid' model is formulated below:

$$\frac{dB}{dt} = NN(x_k, w) - BD \tag{8}$$

$$\frac{dS}{dt} = NN(x_k, w) + (S_f - S)D \tag{9}$$

$$\frac{dP}{dt} = NN(x_k, w) - PD \tag{10}$$

$$\frac{dV}{dt} = F \tag{11}$$

$$D = \frac{F}{V} \tag{12}$$

With the mass balance properly specified, the number of NN inputs required to predict the remaining rate term is reduced from 6 to 3. To thoroughly characterize the potential of the hybrid NODE formulation in the context of the fermentation case study, the NODE is fit to data with varying levels of noise ranging from 0-10%. Shown in Figure 9 is the NODE estimation versus state data for a single batch experiment after training the NODE on all 9 sets of batch data with 5% added noise. Figure 9 also shows NODE estimates of the state derivatives, having removed the poor derivative predictions at time t=0. Save for the initial value, the NODE tends to give reasonable estimates of the state derivatives.

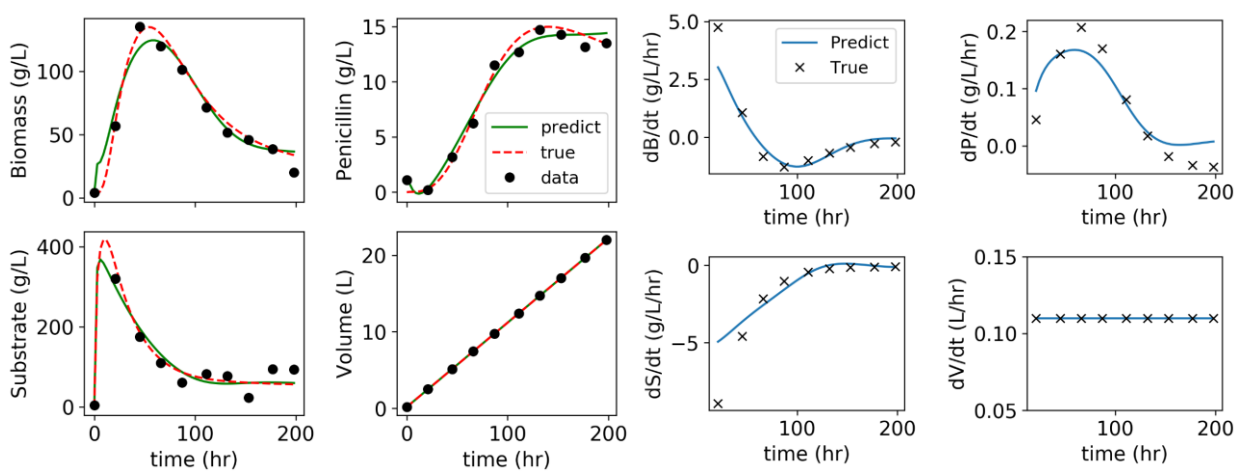

Figure 9. Fit of Neural ODE to penicillin state data (left) and estimate of the state derivatives (right) when data is corrupted with 5% Gaussian noise. Data and fit shown for batch case #1.

Similar to the previous examples, the derivative and state estimates from fitting the hybrid NODE are used to estimate parameters of the mechanistic ODE. Once again, little prior information is assumed about the values of the mechanistic parameters and thus all mechanistic parameters are initialized to equal 2 at the beginning of NLP optimization. The fitted mechanistic model using derivatives estimates from hybrid NODEs trained on different levels of noise are shown in Figure 10.

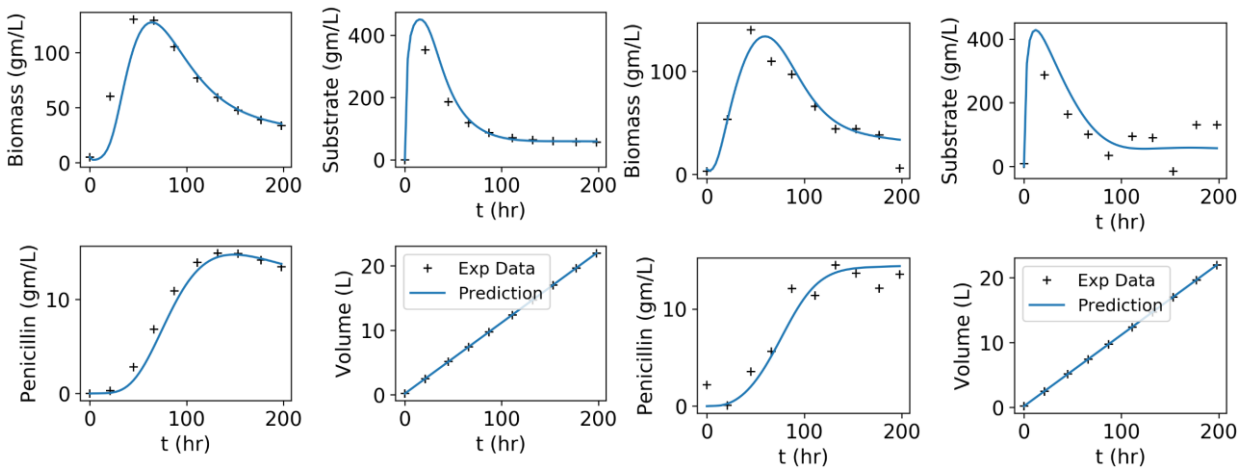

Figure 10. Simulation of the Penicillin ODE system after fitting with data corrupted with 0% (left) and 10% (right) noise. Data and fit shown for batch case #1.

Figure 11 shows calculated errors of the fitted Penicillin model and juxtaposes those errors with the errors from the mechanistic models of the previous case studies fitted via the 2-stage approach. Errors reported in Figure 11 are the mean absolute value error (MAE) between the state data predicted by the fitted mechanistic ODE and the original mechanistic ODE with true parameter values, averaged over N training data points (see equation 13), where i = 1, …, N.

$$\text{MAE} = \frac{\sum abs(x_{i,true} - x_{i,pred})}{N} \tag{13}$$

In order to visualize the errors on the same plot, the MAE of the styrene predictions are scaled by a factor of 10, all other errors left unscaled. The trends in accuracy tend to be consistent with what was observed earlier in the Lotka-Volterra study. In the presence of near perfect data with no noise, the fitted mechanistic model tends to show slightly inferior performance. This is believed to be caused by the NODE slightly overfitting the data, a problem less evident at small amounts (i.e., 1%) of noise. This is

interesting when considering the fact that the NODE is trained by using the measurement data as the fixed initial condition during integration, which becomes more erroneous as the level of noise increases. However, the NODE fit is by no means impervious to poor quality data, and this latter factor explains the increase in fitting error when training on data with greatest corruption (i.e., 10% noise). Nevertheless, the issues of overfitting and poor data quality notwithstanding, by using data from multiple experiments as well as the method of overlapping integration, the Neural ODE still offers a reasonable interpolation of the state data as depicted previously in Figure 9.

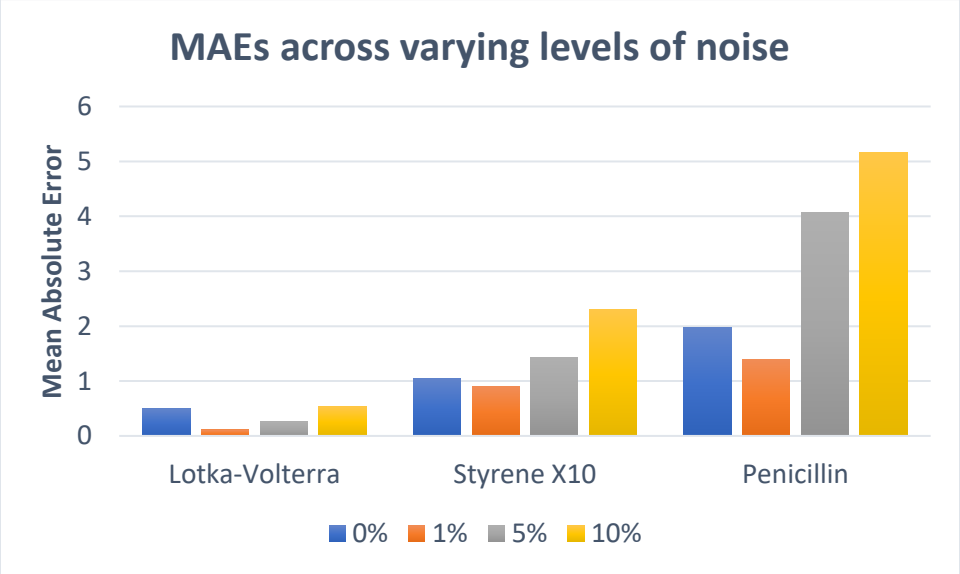

Figure 11. Mean Absolute Error for 3 case studies fitted to data with different levels of noise.

Table 2 shows parameter values of the fitted mechanistic model. Unlike the previous two case studies, the ODE parameters found via the 2-stage approach did not always approach values close to those in the original set of equations simulating the data. As a check that the NLP solution found is a global one, the parameters were also initialized to their true values and the NLP solved with the improved starting values. However, this consistently converged to same set of parameter values as the NLP with poor initial parameter values. This can be attributed to the variance in sensitivity of the parameters. In an actual modeling scenario, some parameters may be identified before model fitting using separate experiments or nominal literature values. Modelers may often choose to fix insensitive parameters to nominal values, thus decreasing the number of mechanistic parameters that require fitting. This would invariably increase the accuracy of the final parameter fit in our 2-stage approach.

Table 2. Actual and fitted parameter estimates for Penicillin case study

	cLmax	kL	ki	m_xm	k	kp	μ_m	kx	qpm	Yps	Yxs

True values	0.0519	0.05	1	0.01	0.0137	0.0001	0.0100	0.3	0.0837	1.2	0.47
0% Noise	0.0447	6.3077	11.0	0.0010	0.0084	9.9708	0.0049	0.1251	0.0083	0.2713	0.4076
1% Noise	0.0382	0.0104	19.99	0.0129	0.0077	9.9922	0.0069	0.2420	0.0056	0.4781	0.4273
5% Noise	0.0461	0.0108	19.99	0.0047	0.0054	9.9796	0.0138	0.535	0.0052	0.2780	0.4406
10% Noise	0.0401	0.0106	19.99	0.0420	0.0026	9.9919	0.0105	0.4693	0.0044	0.6278	0.6673

3.4 Compute Time—Direct and Indirect Approaches

As a final assessment of the relative merits of the NODE 2-stage approach, the computational requirements of the proposed approach and the traditional direct approach were tabulated for the case of noiseless training data. For this study, the NN in the NODE was fixed at 10 hidden nodes. All training studies were conducted on a laptop computer with an Intel Core i7-6700 CPU processor (3.4 GHz). To ensure a fair comparison, the algorithms and software used to find the parameters of mechanistic ODEs and Neural ODEs were kept nearly identical. Specifically, both ODE types are repeatedly integrated using the same numerical integrator (Euler's method), use the same method for gradient calculation (automatic differentiation), and use the same nonlinear optimizer (L-BFGS). Both minimize the same objective function (Eq. 1) except the direct approach does not include the regularization term penalizing large parameter values. For the direct approach, compute time was defined as the time required to train the mechanistic ODE parameters. Whereas the compute time for the indirect approach includes the time to fit the Neural ODE parameters and the time to solve the NLP for the mechanistic parameters—stage 1 and 2 of the indirect approach, respectively. Not included in the time comparison is the hyperparameter tuning. In other words, stage 1 includes only the time required to fit a single NODE. Although hyperparameter tuning invariably increases the compute cost of the NODE 2-stage approach, the cross-validation procedure using grid search can be parallelized to prevent such a procedure from substantially increasing compute times.

The results are present in Table 3. The compute times tend to be comparable despite the larger number of parameters in the Neural Network that must be fit. For example, in the case of the ethylbenzene system, which is made highly nonlinear by the presence of exponential functions, the direct approach is required to fit 3 mechanistic parameters vs. 136 parameters in the Neural ODE. In contrast, the Lotka-Volterra system, which is linear with respect to its 3 parameters, observes minimal compute gains by using the 2-stage approach. It is reasonable to conclude, therefore, that the influence of nonlinear operators on the sensitivity of the parameter gradients, rather than the number of parameters, plays a bigger role in

compute costs. The NLP when properly formulated requires little compute power in comparison to fitting the ODE models.

Table 3. Compute Times for Direct vs Indirect Approaches (dp=Number of data points used for training)

	Lotka Volterra (2 states, 3 params, 20 dp)	**Ethylbenzene (6 states, 3 params, 60 dp)**	**Penicillin (3 states, 11 params, 90 dp)**
Direct Approach (i.e. shooting)	Total: 76 s	Total: 352 s	Did not converge
Indirect NODE or Hybrid ODE Approach	Total: 62 s NODE: 62 s NLP: 0.009 s	Total: 116 s NODE: 110 s NLP: 6.76 s	Total: 183 s HODE: 181 s NLP: 1.932 s

A comparison with the direct approach for the penicillin model was not possible as the direct approach quickly diverged unless parameter estimates close to the true values are supplied as an initial guess. However, obtaining good parameter guesses is not always possible. Thus, more than faster compute times, it may be that the greatest advantage of the 2-stage approach is the ability to obtain reasonable model estimates when little is known about their parameter values. This obviates the need for *ad hoc* scaling and parameter bounding that would be required for direct approaches, which although harder to quantify, may represent a significant time savings of the 2-stage approach.

4. Discussion

There are several aspects and findings of this study that are worth further discussion. Firstly, although there is a trend toward deep machine learning architectures, a simple neural network with a single hidden layer was found to be sufficiently robust when used in the NODE to model the state and rate space of each case study. This is not to say that alternate NN architectures could not improve the approximation accuracy of Neural ODEs—a question that may hold interesting answers, especially for more complex systems or systems with more dimensions. It is also important to mention that this approach offers more than a simple data-driven correction to the mechanistic model. In this approach the NODE (or hybrid ODE) is not the final model, rather it is the tool that helps us arrive at a parametrized mechanistic model.

Hyperparameter tuning may help minimize overfitting of the NODE, generating a more optimal interpolation of the data. In this work, the NODE was selected after hyperparameters such as the number of hidden nodes and regularization weight were varied. Hyperparameters such as the discretization method and termination criteria for training were held constant across case studies or given random initial values (i.e., initialization of NN weights). Other possible hyperparameters that one may choose to consider include the NN structure (number of hidden layers, activation function, etc.) and features of the nonlinear optimizer (e.g. learning rate). We acknowledge, however, that these are all (hyper)parameters of the approach that may need to be optimized for other case studies using either an automated grid search or more advanced techniques. Parallel computing can be used to prevent such a grid search from exponentially increasing compute time. As software packages for implementing NODEs become more standardized, we anticipate selection of these hyperparameters to be increasingly streamlined, making hyperparameter tuning and cross-validation a fast and straightforward process.

Another interesting finding of this work is related to the use of integration when training Neural ODEs. Typical 2-stage approaches use algebraic data-driven models to estimate state derivatives to avoid the time-intensive integration of DEs required in the direct approach. In contrast, by applying NODEs for derivative estimation, the proposed method effectively reintroduces integration into the 2-stage approach, albeit only in the first stage. Algebraic data-driven models, such as the Algebraic Neural Network used in this work, can be trained in fractions of the time required to train Neural ODEs, yet their derivative estimates are not sufficiently accurate for solving the NLP in stage 2. Therefore, as argued in this work, the ability of Neural ODEs to accurately capture derivative information favors their use notwithstanding the added compute cost of integration/gradient calculation during NODE training.

Conversely, although Neural ODEs required more training time than an a-NN, their training time was competitive if not faster than the direct estimation of mechanistic parameters. Although the NODE's neural network architecture used in this work may be small when compared to many deep learning architectures, the NODE had many more parameters that required fitting than the mechanistic model. It may seem counterintuitive that a model with more parameters can be fit with competitive compute cost and more reliably than directly fitting a model with fewer parameters.

Two factors are believed to contribute to this observation. First, due to their large number of parameters, NN fits are often non-unique, enabling the model to interpolate the available data equally well with several combinations of parameter values. This feature is not an issue in terms of generalizability of the proposed approach as the NODE is not the final model and is not expected to extrapolate. Secondly,

gradients used to update parameters during NN training tend to be better behaved and fall within a tighter range than what can be expected of some mechanistic models. This is because the nonlinear operators in the mechanistic ODEs, especially exponential and logarithmic terms, tend to cause parameter values and gradients to range over larger orders of magnitude. For example, in the case of the styrene system, when the direct approach was used and the gradients of the unknown mechanistic parameters (initially assumed to equal 2) were calculated with respect to the loss function at the start of training, it was found that the difference between the true parameter values and parameter gradients varied over 8 orders in magnitude. In contrast, none of the initial gradients of the 136 NODE parameters at the start of training differed from the final parameter values by more than 3 orders in magnitude. A wide discrepancy between parameter gradients and true parameter values invariably leads to poor convergence. In summary, the NODE 2-stage approach can be faster than the direct estimation of mechanistic models by avoiding integration of the mechanistic model during estimation of its parameters, which may be less sensitive and more constrained than NODE parameters. Only in stage 2 of the 2-stage approach must the mechanistic constraints be considered, but since the NLP formulation is already algebraic, no numerical methods are involved at this stage. Although the problem of mechanistic parameter sensitivity could be addressed with model scaling and reformulation, when the mechanistic parameters are unknown, the proper scaling is not always obvious. A more thorough treatment of the Neural ODEs observed stable convergence is a potential topic for future work.

A common concern when training the weights of Neural Networks is convergence to local minima. In this work, designing the algorithm to integrate over overlapping intervals was found to overcome convergence to unacceptable local minima. Moreover, in this work, it was assumed that a sample for each state variable was available at each sampling moment and the initial values for integration were fixed at the measurement values. Although not emphasized in the results, the algorithm also proved to readily generalize to training on data sampled at irregular intervals. For example, the interval between sampled states of the fermenter varied slightly between 21 and 24 hrs, requiring the automated adjustment of the Euler step size during training. For cases where the data is even sparser, states are measured at uneven time intervals or some measurements are missing, algebraic data-driven models could be used to interpolate missing state values. However, it should be acknowledged that if data is too sparse such that it does not cover the curvature of state trajectory, NODE interpolations will not represent the true trajectory well and the 2-stage method is not expected to give satisfactory results. Global optimization methods could be added to the Neural ODE training stage to ensure global convergence; however, this would significantly increase computational cost. More importantly, as the NODE is not the end model and already offers the needed accuracy for derivative estimates, global optimization methods for NODE

training might not be necessary. Although not done here, the weights of the neural network could be pre-trained with the courser derivative estimates of algebraic data-driven models prior to training the Neural ODE, which could further accelerate NODE training.

A potential weakness of estimating system derivatives with data-driven models is the poor derivative estimates at initial conditions. This is unfortunate as the most interesting nonlinear behavior tends to occur at the initial stage of the process. This behavior has been observed repeatedly for splines[34, 41] but has not been commented on for NODEs. A hypothesis for this behavior is that the initial system derivatives present an extra degree of freedom not constrained by data as at intermediate time points. If the modeler knows the initial rates these could be enforced, eliminating the extra degree of freedom, though such information is generally not known. More realistically, since the NODE can be regressed on multiple batches of data, thoughtful design of experiments could mitigate the effect of poor initial estimates. Regardless, this work follows the heuristic of removing derivative estimates at initial conditions, and the two-stage approach still managed to yield reasonable fits to the state data.

It is well-known that a direct approach can have greater statistical accuracy than indirect approaches—this is because the additional step of fitting the data-driven model in the indirect approach may incur an 'information loss' that may bias the final mechanistic model fit.[65, 66] Even in this case, as has been demonstrated in previous studies,[30, 33, 66] the discovered parameters from the faster indirect approach can be used as initial guesses in the direct approach to alleviate some of the computational burden. This advantage would be even more pronounced in situations where the exact formula of the mechanistic model is uncertain. In this contribution, it was assumed that the available mechanistic model was the same as mechanistic ODEs that simulated the measurement data. However, when the true ODE model is unknown, the modeler may be required to select between multiple mechanistic models with different parameterizations. In this case, a single NODE can be regressed whose derivative estimates are used to fit all the proposed mechanistic models separately, a task that would be computationally more significant with a direct approach.

Finally, it should be stressed that our proposed approach is not exempt from issues of parameter identifiability and sensitivity. With all parameter estimation approaches, one should verify that data quantity and quality is adequate to obtain the needed level of parameter precision prior to parameter fitting (e.g. by conducting identifiability and sensitivity analysis). Moreover, other sources of domain knowledge that could be used to improve initial guesses and tighten the feasibility bounds for parameters should be incorporated when available. As with other indirect approaches, the NODE indirect approach

does not have a straightforward extension to systems with unmeasured (i.e. latent) states. If the mechanistic DEs are assumed a function of unmeasured states, methods typically associated with the direct approaches will be required to relate unmeasured states to measured variables during parameter estimation. A comparison of the proposed approach with direct integration methods was conducted to highlight the advantage of the proposed approach versus direct approaches. However, to fully classify the scenarios where indirect estimation via Neural ODEs would be superior, a more comprehensive comparison with other DE solution methods mentioned in the introduction should be conducted. Many of these direct approaches are under active development or being revisited and a full comparison is outside the scope of this study.

5. Conclusions

This work compared the ability of NODEs and Algebraic NNs to extract state derivative information from data. It further compared the indirect approach based on NODEs with a direct NLS approach for regressing ODE models. A clear increase in accuracy was shown when NODEs are the interpolating model. Other data-driven models could be used to estimate system derivatives and an exhaustive comparison with all methods was outside the scope of this work. However, we anticipate that NODEs will outperform all methods based on algebraic interpolating models (e.g. splines) as none of these methods consider state derivatives during model fitting. Moreover, a single algebraic data-driven model has no straightforward way to interpolate data from multiple batch runs based on different system forcing conditions. In contrast, the differential equation-based NODEs can easily incorporate data from different system conditions via user-specification of initial and boundary conditions and external forcing variables.

Although the Neural ODE-based approach showed computational gains over direct integration of the mechanistic ODEs, the most attractive advantage of this approach lies in its ability to find mechanistic parameters with minimal prior knowledge of their values and minimal parameter scaling. Improvement in parameter estimation is most notable for mechanistic DEs that require parameterization of highly nonlinear operators (e.g. logarithms and exponentials). Many interesting questions remain in regards to possible extensions of the method. Especially interesting would be analyses on the scalability of the NODE 2-stage method to more complex differential equation systems (e.g. PDEs) and higher order DE systems, for which previous work encourages promising results.[47, 67] It may also be worth exploring the effect of including the initial conditions as trainable parameters along with the NODE parameters if the initial conditions are uncertain. The potential to improve parameter estimates by solving the NLP with global optimization solvers is yet another interesting direction. These and other directions are a matter for future investigation.

Supporting Information

A complete mathematical formulation for each case study, its parameter values, and the system conditions used to simulate training data is summarized the Supporting Information file of this paper. This information is available free of charge via the Internet at http://pubs.acs.org/.

Author Information

Corresponding Author

Fani Boukouvala – Department of Chemical and Biomolecular Engineering, Georgia Institute of Technology, 311 Ferst Dr., N.W. Atlanta, Georgia, 30332-0100 USA

*Tel.: (404) 385-5371, Email: fani.boukouvala@chbe.gatech.edu, ORCID iD: 0000-0002-0584-1517

Authors

William Bradley – Department of Chemical and Biomolecular Engineering, Georgia Institute of Technology, 311 Ferst Dr., N.W. Atlanta, Georgia, 30332-0100 USA, ORCID iD: 0000-0003-2505-6898

Notes

The authors declare no competing financial interest.

Acknowledgements

WB and FB gratefully acknowledge funding received from RAPID/NNMI Grant #GR10002225, Georgia Tech start-up grant and NSF CBET grants (1336386 and **1944678**).

Abbreviations

ANN – Algebraic Neural Network

DE – Differential Equation

EB – Ethylbenzene

FF – Frequency Factor

IVP – Initial Value Problem

ML – Machine Learning

NLP – Nonlinear Program(ming)

NLS – Nonlinear Least Squares

NN – Neural Network

NODE – Neural Ordinary Differential Equation

ODE – Ordinary Differential Equation

PDE – Partial Differential Equation

References

1. Venkatasubramanian, V., The promise of artificial intelligence in chemical engineering: Is it here, finally? *Aiche J* **2019,** *65* (2), 466-478.
2. Lee, D.; Jayaraman, A.; Kwon, J. S., Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling. *PLOS Computational Biology* **2020,** *16* (12), e1008472.
3. Quaghebeur, W.; Nopens, I.; Baets, B. D., Incorporating Unmodeled Dynamics Into First-Principles Models Through Machine Learning. *IEEE Access* **2021,** *9*, 22014-22022.
4. von Stosch, M.; Oliveira, R.; Peres, J.; Feyo de Azevedo, S., Hybrid semi-parametric modeling in process systems engineering: Past, present and future. *Comput Chem Eng* **2014,** *60*, 86-101.
5. Willard, J.; Jia, X.; Xu, S.; Steinbach, M.; Kumar, V., Integrating Physics-Based Modeling with Machine Learning: A Survey. *ArXiv* **2020,** *abs/2003.04919*.
6. Ahmad, I.; Ayub, A.; Kano, M.; Cheema, I. I., Gray-box Soft Sensors in Process Industry: Current Practice, and Future Prospects in Era of Big Data. *Processes* **2020,** *8* (2), 243.
7. Rüden, L. v.; Mayer, S.; Beckh, K.; Georgiev, B.; Giesselbach, S.; Heese, R.; Kirsch, B.; Pfrommer, J.; Pick, A.; Ramamurthy, R.; Walczak, M.; Garcke, J.; Bauckhage, C.; Schücker, J., Informed Machine Learning -- A Taxonomy and Survey of Integrating Knowledge into Learning Systems. *arXiv: Machine Learning* **2019**.
8. Hemker, P., Numerical methods for differential equations in system simulation and in parameter estimation : (Analysis and simulation of biochemical systems; proc. of the 8th FEBS meeting, Amsterdam, 1972, p 59-80). Stichting Mathematisch Centrum: 1972.
9. Bard, Y., Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems. *SIAM Journal on Numerical Analysis* **1970,** *7* (1), 157-186.
10. Benson, M., Parameter fitting in dynamic models. *Ecological Modelling* **1979,** *6* (2), 97-115.
11. Li, Z.; Osborne, M. R.; Prvan, T., Parameter estimation of ordinary differential equations. *IMA Journal of Numerical Analysis* **2005,** *25* (2), 264-285.
12. Ramsay, J. O., Principal Differential Analysis: Data Reduction by Differential Operators. *Journal of the Royal Statistical Society: Series B (Methodological)* **1996,** *58* (3), 495-508.
13. Varziri, M. S.; Poyton, A. A.; McAuley, K. B.; McLellan, P. J.; Ramsay, J. O., Selecting optimal weighting factors in iPDA for parameter estimation in continuous-time dynamic models. *Comput Chem Eng* **2008,** *32* (12), 3011-3022.
14. Ramsay, J. O.; Hooker, G.; Campbell, D.; Cao, J., Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **2007,** *69* (5), 741-796.
15. Putter, H.; Heisterkamp, S. H.; Lange, J. M. A.; de Wolf, F., A Bayesian approach to parameter estimation in HIV dynamical models. *Statistics in Medicine* **2002,** *21* (15), 2199-2214.
16. Huang, Y.; Liu, D.; Wu, H., Hierarchical Bayesian Methods for Estimation of Parameters in a Longitudinal HIV Dynamic System. *Biometrics* **2006,** *62* (2), 413-423.
17. Calderhead, B.; Girolami, M.; Lawrence, N. D., Accelerating Bayesian inference over nonlinear differential equations with Gaussian processes. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, Curran Associates Inc.: Vancouver, British Columbia, Canada, 2008; pp 217–224.
18. Dondelinger, F.; Husmeier, D.; Rogers, S.; Filippone, M., ODE parameter inference using adaptive gradient matching with Gaussian processes. In *Proceedings of the Sixteenth International*

Conference on Artificial Intelligence and Statistics, Carlos, M. C.; Pradeep, R., Eds. PMLR: Proceedings of Machine Learning Research, 2013; Vol. 31, pp 216--228.

19. Chkrebtii, O. A.; Campbell, D. A.; Calderhead, B.; Girolami, M. A., Bayesian Solution Uncertainty Quantification for Differential Equations. *Bayesian Anal.* **2016,** *11* (4), 1239-1267.

20. Wang, Y.; Barber, D., Gaussian processes for Bayesian estimation in ordinary differential equations. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, JMLR.org: Beijing, China, 2014; pp II–1485–II–1493.

21. Schober, M.; Duvenaud, D.; Hennig, P., Probabilistic ODE solvers with Runge-Kutta means. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, MIT Press: Montreal, Canada, 2014; pp 739–747.

22. Biegler, L. T., An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification* **2007,** *46* (11), 1043-1053.

23. Bock, H. G.; Plitt, K. J., A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems*. *IFAC Proceedings Volumes* **1984,** *17* (2), 1603-1608.

24. Baake, E.; Baake, M.; Bock, H. G.; Briggs, K. M., Fitting ordinary differential equations to chaotic data. *Physical Review A* **1992,** *45* (8), 5524-5529.

25. van Domselaar, B.; Hemker, P., Nonlinear parameter estimation in initial value problems, Technical Report NW 18/75. Mathematical Centre Amsterdam: 1975.

26. Hamilton, F. In *Parameter Estimation in Differential Equations: A Numerical Study of Shooting Methods*, 2011.

27. Tjoa, I. B.; Biegler, L. T., Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research* **1991,** *30* (2), 376-385.

28. Rackauckas, C.; Ma, Y.; Dixit, V.; Guo, X.; Innes, M.; Revels, J.; Nyberg, J.; Ivaturi, V. D., A Comparison of Automatic Differentiation and Continuous Sensitivity Analysis for Derivatives of Differential Equation Solutions. *ArXiv* **2018,** *abs/1812.01892*.

29. Ding, A. A.; Wu, H., Estimation of Ordinary Differential Equation Parameters Using Constrained Local Polynomial Regression. *Stat Sin* **2014,** *24* (4), 1613-1631.

30. Chang, J.-S.; Li, C.-C.; Liu, W.-L.; Deng, J.-H., Two-stage parameter estimation applied to ordinary differential equation models. *Journal of the Taiwan Institute of Chemical Engineers* **2015,** *57*, 26-35.

31. Huang, H.; Handel, A.; Song, X., A Bayesian approach to estimate parameters of ordinary differential equation. *Computational Statistics* **2020,** *35* (3), 1481-1499.

32. Swartz, J.; Bremermann, H., Discussion of parameter estimation in biological modelling: Algorithms for estimation and evaluation of the estimates. *Journal of Mathematical Biology* **1975,** *1* (3), 241-257.

33. Varah, J. M., A Spline Least Squares Method for Numerical Parameter Estimation in Differential Equations. *SIAM Journal on Scientific and Statistical Computing* **1982,** *3* (1), 28-46.

34. Liang, H.; Wu, H., Parameter Estimation for Differential Equation Models Using a Framework of Measurement Error in Regression Models. *Journal of the American Statistical Association* **2008,** *103* (484), 1570-1583.

35. Brunel, N. J. B., Parameter estimation of ODE's via nonparametric estimators. *Electron. J. Statist.* **2008,** *2*, 1242-1267.

36. Boltyanskiy, V.; Gamkrelidze, R. V. y.; Pontryagin, L. S. *Theory of optimal processes*; JOINT PUBLICATIONS RESEARCH SERVICE ARLINGTON VA: 1961.

37. Bryson, A.; Ho, Y. C.; Siouris, G., Applied Optimal Control: Optimization, Estimation, and Control. *Systems, Man and Cybernetics, IEEE Transactions on* **1979,** *9*, 366-367.

38. Mehrkanoon, S.; Mehrkanoon, S.; Suykens, J. A. K., Parameter estimation of delay differential equations: An integration-free LS-SVM approach. *Communications in Nonlinear Science and Numerical Simulation* **2014,** *19* (4), 830-841.
39. Dua, V., An Artificial Neural Network approximation based decomposition approach for parameter estimation of system of ordinary differential equations. *Comput Chem Eng* **2011,** *35* (3), 545-553.
40. Dattner, I., Differential equations in data analysis. *WIREs Computational Statistics n/a* (n/a), e1534.
41. Yang, A.; Martin, E.; Morris, J., Identification of semi-parametric hybrid process models. *Comput Chem Eng* **2011,** *35* (1), 63-70.
42. Cybenko, G., Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **1989,** *2* (4), 303-314.
43. Psichogios, D. C.; Ungar, L. H., A hybrid neural network-first principles approach to process modeling. *Aiche J* **1992,** *38* (10), 1499-1511.
44. Thompson, M. L.; Kramer, M. A., Modeling chemical processes using prior knowledge and neural networks. *Aiche J* **1994,** *40* (8), 1328-1340.
45. Rico-Martínez, R.; Krischer, K.; Kevrekidis, I. G.; Kube, M. C.; Hudson, J. L., DISCRETE- vs. CONTINUOUS-TIME NONLINEAR SIGNAL PROCESSING OF Cu ELECTRODISSOLUTION DATA. *Chemical Engineering Communications* **1992,** *118* (1), 25-48.
46. Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D. Neural Ordinary Differential Equations *arXiv e-prints* [Online], 2018. https://ui.adsabs.harvard.edu/abs/2018arXiv180607366C (accessed June 01, 2018).
47. Rackauckas, C.; Ma, Y.; Martensen, J.; Warner, C.; Zubov, K.; Supekar, R.; Skinner, D.; Ramadhan, A. Universal Differential Equations for Scientific Machine Learning *arXiv e-prints* [Online], 2020, p. arXiv:2001.04385. https://ui.adsabs.harvard.edu/abs/2020arXiv200104385R (accessed January 01, 2020).
48. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S., PyTorch: An Imperative Style, High-Performance Deep Learning Library. *ArXiv* **2019,** *abs/1912.01703*.
49. Wächter, A.; Biegler, L. T., On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* **2006,** *106* (1), 25-57.
50. Amestoy, P.; Buttari, A.; Duff, I.; Guermouche, A.; L'Excellent, J.-Y.; Uçar, B., Mumps. In *Encyclopedia of Parallel Computing*, Padua, D., Ed. Springer US: Boston, MA, 2011; pp 1232-1238.
51. Hart, W. E.; Laird, C. D.; Watson, J.-P.; Woodruff, D. L.; Hackebeil, G. A.; Nicholson, B. L.; Siirola, J. D., *Pyomo--optimization modeling in python*. Springer International Publishing: 2017.
52. Hart, W. E.; Watson, J.-P.; Woodruff, D. L., Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation* **2011,** *3* (3), 219.
53. Lotka, A. J., Contribution to the Mathematical Theory of Capture. *Proceedings of the National Academy of Sciences* **1932,** *18* (2), 172.
54. Volterra, V., *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Società anonima tipografica "Leonardo da Vinci: Città di Castello, 1926.
55. Sánchez-Pérez, J. F.; Conesa, M.; Alhama, I.; Cánovas, M., Study of Lotka–Volterra Biological or Chemical Oscillator Problem Using the Normalization Technique: Prediction of Time and Concentrations. *Mathematics* **2020,** *8* (8), 1324.
56. Joseph, T. A.; Shenhav, L.; Xavier, J. B.; Halperin, E.; Pe'er, I., Compositional Lotka-Volterra describes microbial dynamics in the simplex. *PLOS Computational Biology* **2020,** *16* (5), e1007917.

57. Stenseth, N. C.; Falck, W.; Bjørnstad, O. N.; Krebs, C. J., Population regulation in snowshoe hare and Canadian lynx: Asymmetric food web configurations between hare and lynx. *Proceedings of the National Academy of Sciences* **1997,** *94* (10), 5147-5152.
58. Bishop, C. M., Training with noise is equivalent to Tikhonov regularization. *Neural Comput.* **1995,** *7* (1), 108–116.
59. Sietsma, J.; Dow, R. J. F., Creating artificial neural networks that generalize. *Neural Networks* **1991,** *4* (1), 67-79.
60. Holmstrom, L.; Koistinen, P., Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks* **1992,** *3* (1), 24-38.
61. Poole, B.; Sohl-Dickstein, J.; Ganguli, S., Analyzing noise in autoencoders and deep networks. *ArXiv* **2014,** *abs/1406.1831*.
62. Rubanova, Y.; Chen, R. T.; Duvenaud, D., Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907* **2019**.
63. Snyder, J. D.; Subramaniam, B., A novel reverse flow strategy for ethylbenzene dehydrogenation in a packed-bed reactor. *Chemical Engineering Science* **1994,** *49* (24, Part 2), 5585-5601.
64. Fogler, H. S., *Elements of chemical reaction engineering*. Third edition. Upper Saddle River, N.J. : Prentice Hall PTR, [1999] ©1999: 1999.
65. Xue, H.; Miao, H.; Wu, H., Sieve Estimation of Constant and Time-Varying Coefficients in Nonlinear Ordinary Differential Equation Models by Considering Both Numerical Error and Measurement Error. *Annals of statistics* **2010,** *38 4*, 2351-2387.
66. Michalik, C.; Chachuat, B.; Marquardt, W., Incremental Global Parameter Estimation in Dynamical Systems. *Industrial & Engineering Chemistry Research* **2009,** *48* (11), 5489-5497.
67. Norcliffe, A.; Bodnar, C.; Day, B.; Simidjievski, N.; Lió, P., On Second Order Behaviour in Augmented Neural ODEs. *ArXiv* **2020,** *abs/2006.07220*.

Abstract Graphic

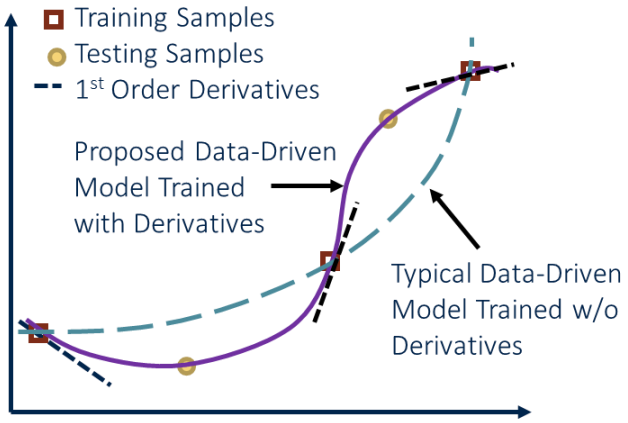

Table of Contents