MULTISTEP AND CONTINUOUS PHYSICS-INFORMED NEURAL NETWORK METHODS FOR LEARNING GOVERNING EQUATIONS AND CONSTITUTIVE RELATIONS

Ramakrishna Tipireddy,1,* Paris Perdikaris,2 Panos Stinis,1 & Alexandre Tartakovsky1,3

1Pacific Northwest National Laboratory, MSIN K7-90, Richland, Washington 99352, USA
2Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA
3Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois 61801, USA

*Address all correspondence to: Ramakrishna Tipireddy, Pacific Northwest National Laboratory, MSIN K7-90, Richland, Washington 99352, USA; Tel.: +1 509 375 6308, E-mail: ramakrishna.tipireddy@pnnl.gov

Original Manuscript Submitted: 11/12/2021; Final Draft Received: 1/24/2022

We investigate the applicability and relative merit of discrete and continuous versions of physics-informed neural network (PINN) methods for learning unknown governing equations or constitutive relations in a nonlinear dynamical system. In the case of unknown dynamics, entire right-hand-side (RHS) equations of the ordinary differential equations are unknown. In the case of unknown constitutive relations, however, the RHS equations are known up to the specification of constitutive relations (that may depend on the state of the system). We use a deep neural network to model unknown governing equations or constitutive relations. The discrete PINN approach combines classical multistep discretization methods for dynamical systems with neural-network-based machine learning methods. On the other hand, the continuous versions utilize deep neural networks to minimize the residual function for the continuous governing equations. We use the case of a fedbatch bioreactor system to study the effectiveness of these approaches and discuss conditions for their applicability. Our results indicate that the accuracy of the trained neural network models is much higher for the cases where we only have to learn a constitutive relation instead of all dynamics. This finding corroborates the well-known fact from scientific computing that building as much structural information as is available into an algorithm can enhance its efficiency and/or accuracy.

KEY WORDS: physics-informed neural networks, machine learning, deep learning, multistep methods
1. INTRODUCTION

Due to the abundance of data and computational resources (e.g., graphical processing units), machine learning methods and, in particular, deep learning approaches have gained prominence across various application fields, including the design of sustainable solutions for environmental problems (Kamali et al., 2021). The rising interest in deep learning methods is also due to the fact that the physics (and biochemistry) of many natural and engineered systems is not fully known (Psichogios and Ungar, 1992; Tartakovsky et al., 2020) and, therefore, the existing mathematical models of such systems are highly uncertain, making it challenging to design sustainable solutions for such systems.

In recent years, there has been significant interest in the scientific computing community to utilize deep learning methods to accelerate scientific discovery. Combining the partial or complete knowledge about the physics of a system with data through the use of deep learning has resulted in novel computational approaches (Berry et al., 2015; Chen et al., 2018; Felsberger and Koutsourelakis, 2019; Han et al., 2018; Ma et al., 2018; Raissi et al., 2019; Sirignano and Spiliopoulos, 2018; Wan et al., 2018). Dynamical systems play an important role in modeling and simulation of many real-world physical phenomena, such as structural dynamics, weather and climate systems, and biochemical systems, that are central to sustainability research. Often, mathematical models of the dynamical systems are derived from first principles; however, these models are only approximations of the real systems because of underlying assumptions. We often encounter dynamical systems whose mathematical models are completely known, partially known, or completely unknown.

In this work, we use the physics-informed neural networks (PINNs) method (Karniadakis et al., 2021) to study dynamical systems for which a mathematical model is either partially known or completely unknown. In the first case, we assume that the governing equations of the dynamical system are known but that the constitutive relationships of certain variables are unknown. In the second case, we assume that the equations governing the dynamical systems are completely unknown. We assume that the data corresponding to the state of the dynamical system are available through observations of an experiment or through numerical simulations.

There are “discrete” (e.g., Raissi et al., 2018) and “continuous” (e.g., He and Tartakovsky, 2021; Raissi et al., 2019) versions of the PINN method. In the discrete PINN method, the equations are numerically discretized in time and for each time step, and the states and/or unknown functions in the equation are approximated using deep neural networks (DNNs) with appropriate inputs that do not include time. In the continuous PINN method, the states are approximated with DNNs as functions of time (and other relevant variables), which eliminates the need for numerically discretizing the governing equation. PINN methods have been used to learn constant and space-dependent coefficients in PDE models (e.g., He et al., 2020; Raissi et al., 2019) and unknown physics in the form of constitutive relationships in steady-state partial-differential equation models (e.g., Reyes et al., 2021; Tartakovsky et al., 2020). In this work, we employ both the discrete and continuous PINN methods.

It is important to understand the applicability and the relative merits of these methods. In the current work, we aim to accomplish this task by testing these methods on the same physics model, namely, a fedbatch bioreactor model (FBR) (Psichogios and Ungar, 1992). Although the

---

†We note here that, strictly speaking, using the term PINN when the physics are completely unknown is a misnomer. However, with a slight abuse of terminology we will use PINN even for this case (please see Sections 2.2 and 2.4 for more details).
methods discussed here are very general and can be used for any dynamical system modeled with ordinary differential equations (ODEs), we choose the FBR model because it offers a dynamical system that is tractable but also very sensitive to initial conditions and variations of the problem parameters.

Our results indicate that the accuracy of the trained neural network models is much higher for the cases where we only have to learn a constitutive relation instead of all dynamics. This finding corroborates the well-known fact from scientific computing that building as much structural information as is available into an algorithm can enhance its efficiency and/or accuracy.

The structure of the paper is as follows: in Section 2, we discuss both discrete and continuous PINN methods for dynamical systems. We model an FBR as a nonlinear dynamical system to test the various PINN models in Section 3. In Section 4, we present numerical results for the various PINN models when applied to the FBR model. Section 5 contains conclusions and ideas for future work.

2. PINNS FOR DYNAMICAL SYSTEMS

We consider a dynamical system modeled as an ODE system as follows:

\[ \dot{y}(t) = f(y(t); \mu(y(t))), \]

where the state of the system at any time \( t \) is given by the vector \( y(t) \in \mathbb{R}^D \), \( f \) describes the dynamics of the system, and \( \mu(y) \) is a constitutive relation. We consider two cases: (1) where the governing dynamics \( f \) are completely unknown, and (2) where the form of \( f \) is known but the constitutive relation \( \mu(y) \) is unknown. The unknown governing equations could also depend on some external forcing, although we do not examine this case here. Given the measurements of \( y(t) \) at different time instants \( t_1, t_2, \ldots, t_n \), we want to learn all the unknown parts of the dynamics \( f \) using PINNs.

2.1 Neural Network Model

When the functional form of \( f(y(t); \mu) \) is completely unknown, we model it as a function of the states using a neural network [Fig. 1(a)] and train this neural network with time series data of the states. When \( f(y(t); \mu) \) is known but \( \mu(y(t)) \) is unknown, we represent \( \mu(y(t)) \) using a neural network [Fig. 1(b)].

2.2 Multistep Neural Network Model to Learn Unknown Governing Equations

In this section, we combine the multistep family of time-stepping methods (Iserles, 2009) for solving dynamical systems with PINNs (Raissi et al., 2018) to learn the unknown governing equations of a system from time series data of the state \( y_n \). We discretize (1) in time using a generalized multistep method with \( M \) steps as follows:

\[ \sum_{m=0}^{M} [\alpha_m y_{n-m} + \Delta t \beta_m f(y_{n-m})] = 0, \quad n = M, \ldots, N, \]

where \( y_{n-m} \) denotes the state of the system at time \( t_{n-m} \). Note that for a multistep method with \( M \) steps, we need to provide the first \( M \) steps for the initialization of the method, i.e.,
FIG. 1: Neural network model for (a) unknown governing equations and (b) unknown constitutive relations

the values $y_0, \ldots, y_{M-1}$. Different choices of $M$, $\alpha_m$, and $\beta_n$ lead to different schemes. For $M = 1$, $\alpha_0 = 1$, $\alpha_1 = -1$, and $\beta_0 = \beta_1 = 0.5$, we find the trapezoidal rule given by

$$y_n = y_{n-1} + \frac{1}{2} \Delta t [f(y_n) + f(y_{n-1})], \quad n = 1, \ldots, N. \quad (3)$$

If the dynamics of the system are completely unknown, then we approximate the function $f(y)$ by a feed-forward neural network and learn the parameters of the neural network from the time series data. We do so by minimizing the mean square error loss function given by

$$\text{MSE} = \frac{1}{N-M-1} \sum_{n=M}^{N} |l_n|^2, \quad (4)$$

where $l_n$ measures how well the neural network approximation $f^{\text{NN}}(y)$ of $f(y)$ reproduces the exact (but unknown) dynamics of the ODE at time $t_n$. The quantity $l_n$ is given by

$$l_n = \sum_{m=0}^{M} [\alpha_m y_{n-m} + \Delta t \beta_m f^{\text{NN}}(y_{n-m})], \quad n = M, \ldots, N. \quad (5)$$

Once $f^{\text{NN}}(y)$ is learned, the $y(t)$ can be found for any initial condition $y_0$ by numerically solving the equation

$$\dot{y}(t) = f^{\text{NN}}(y(t)), \quad y(0) = y_0. \quad (6)$$

2.3 Multistep Neural Network Model to Learn Constitutive Relations

In this section, we employ a multistep neural network method to learn only the constitutive relation process $\mu(y(t))$ of the dynamical system instead of all dynamics $f(y(t); \mu)$. This method is appropriate for situations where the governing equations of the dynamical system are known up to certain constitutive relations. In this case, we represent the unknown constitutive relation process $\mu(y(t))$ by a neural network and train as before to minimize the mean square error loss
MSE. In the multistep neural network approach for learning unknown governing equations and constitutive relations, we show different cases where multiple solution trajectories for \( (1) \) with different initial conditions and simulation times are used as the training data. After learning \( \mu_{NN}(y(t)) \), \( y(t) \) for any initial condition \( y_0 \) can be found by numerically solving the equation
\[
\dot{y}(t) = f(y(t), \mu_{NN}(y(t))), \quad y(0) = y_0.
\] (7)

### 2.4 Continuous Model to Learn Unknown Governing Equations

In the continuous PINN method, when the dynamics \( f(y(t)) \) are unknown, we model the state vector \( y(t) \) as a function of time and the RHS describing the dynamics \( f(y(t)) \) as a function of states \( y(t) \) using neural networks. We represent \( y \) and \( f \) as neural network models and train using time series data \( y^*(t_n) \) \((n = 1, \ldots, N_y)\) corresponding to the initial condition \( y(0) = y_0^* \) by minimizing the following loss function:
\[
\text{loss}(f_{NN}(y_{NN}(t))) = \frac{1}{N_y} \sum_{n=1}^{N_y} (y_{NN}(t_n) - y^*(t_n))^2 \nonumber \\
+ \frac{1}{N_f} \sum_{n=1}^{N_f} \left( \frac{dy_{NN}}{dt}(t_n) - f_{NN}(y_{NN}(t_n)) \right)^2.
\] (8)

One advantage of the continuous PINN method is that there is no need to discretize the dynamical system in time for learning \( f \) because the time derivative of \( y_{NN} \) in the loss function (8) can be computed using automatic differentiation. Another related advantage of the continuous PINN is that the \( y \) measurements should not be uniformly spaced as is required in the multistep PINN method. We note that the main objective of both PINN methods is to learn \( f(y) \) rather than \( y(t) \), noting that the latter depends on the initial condition. Similar to the multistep method \( y(t) \) can be found for any initial condition \( y_0 \) by numerically solving Eq. (6).

However, the continuous PINN method also allows for finding the state \( y \) corresponding to the initial condition \( y_0^* \) of the training set for any time within the time duration of the training data.

### 2.5 Continuous Neural Network Model to Learn Constitutive Relations

In this section, we employ the continuous PINN method for learning constitutive relations of the system. For the dynamical system described in Eq. (1), we represent the state of the system with the \( y_{NN}(t) \) neural network and constitutive relationship with the \( \mu_{NN}(y_{NN}(t)) \) neural network and train these neural networks by minimizing the following loss function:
\[
\text{loss}(\mu_{NN}(y_{NN}(t))) = \frac{1}{N_y} \sum_{n=1}^{N_y} (y_{NN}(t_n) - y^*(t_n))^2 \nonumber \\
+ \frac{1}{N_f} \sum_{n=1}^{N_f} \left( \frac{dy_{NN}}{dt}(t_n) - f(y_{NN}(t_n), \mu_{NN}(y_{NN}(t_n))) \right)^2.
\] (9)
where \( N_y \) is the number of measurements of the system state and \( N_f \) is the total number of predetermined collocation points. As in the above examples, the solution \( y(t) \) for any initial condition \( y_0 \) can be found by numerically solving Eq. (7).

3. FEDBATCH BIOREACTOR MODEL

Dynamical systems describing biological phenomena are very complex and can be of the form described in (1) where the state vector \( y \) depends on the constitutive relation \( \mu \), which in turn depends on the state as follows:

\[
\mu = g(y(t)),
\]

for some function \( g \). Such an interdependence is more often than not expressed through nonlinear terms, and the resulting system is often chaotic. The dependence of the constitutive relation \( \mu \) on the state vector is in general unknown and is difficult to derive from first principles due to complex biological reaction kinetics. Therefore, one has to estimate these constitutive relations by indirect means using experimental data. Inaccurate modeling of constitutive relations results in inaccurate predictions of the state vector.

Here we consider a bioreactor operating in fedbatch (nonstationary) conditions in which the microbial growth exhibits a wide range of dynamical behavior (Psichogios and Ungar, 1992) due to the continuous change of the growth rate in a complex manner. The FBR model is given by

\[
\begin{align*}
\frac{dX(t)}{dt} &= \mu(X(t), S(t), V(t))X(t) - \frac{F(t)X(t)}{V(t)}, \\
\frac{dS(t)}{dt} &= -k_1\mu(X(t), S(t), V(t))X(t) + \frac{F(t)(S_{in}(t) - S(t))}{V(t)}, \\
\frac{dV(t)}{dt} &= F(t),
\end{align*}
\]

subject to the initial conditions

\[
X(0) = X_0, \quad S(0) = S_0, \quad V(0) = V_0,
\]

where \( X(t) \) is the biomass concentration, \( S(t) \) is the substrate concentration, and \( V(t) \) is the volume of the bioreactor. The dynamics of the bioreactor are described by the mass balance between the reacting species and the kinetics of the specific growth rate \( \mu(X(t), S(t), V(t)) \), which accounts for the rate at which the substrate is converted to the biomass. Other parameters in the governing equations are the inlet substrate concentration \( S_{in}(t) \), the flow rate \( F(t) \), and the substrate-to-cell conversion coefficient \( k_1 \). As the ground truth for \( \mu \), we consider the Haldane model from Psichogios and Ungar (1992):

\[
\mu(S(t)) = \frac{\mu^* S(t)}{K_m + S(t) + \frac{(S(t))^2}{K_i}},
\]

where \( K_m \) and \( K_i \) are model constants. In combination with Eq. (11), this model is used to generate synthetic data \( (X(t_i), S(t_i), V(t_i), i = 1, \ldots, N_y) \).
3.1 FBR Model with Multistep and Continuous PINNs for Learning the Unknown Governing Equations

Let the state vector of the dynamical system be \( y(t) = [X(t), S(t), V(t)]^T \). The RHS of the dynamical system is described by the unknown function \( f(y(t)) = [f_1(y(t)), f_2(y(t)), f_3(y(t))]^T \) such that

\[
\begin{align*}
\frac{dX(t)}{dt} &= f_1[X(t), S(t), V(t)], \\
\frac{dS(t)}{dt} &= f_2[X(t), S(t), V(t)], \\
\frac{dV(t)}{dt} &= f_3[X(t), S(t), V(t)].
\end{align*}
\]

Using vector notation, we have

\[
\frac{dy}{dt} = f(y(t)).
\]

Then, the multistep and continuous PINN methods can be applied for learning \( f(y(t)) \) as described in Sections 2.2 and 2.4, respectively. We want to note that the equation for \( V(t) \) in the FBR model (11) depends only on the flow rate \( F(t) \), which is an explicit function of time \( t \). Yet, we assume a general formulation (14) that is an implicit function of \( t \) and allows for a dependence on \( X, S, \) and \( V \). Our results show that the PINN methods attempt to learn the correct form of \( f_3 \).

3.2 FBR Model with Multistep and Continuous PINNs for Learning the Constitutive Relation

Using the vector notation introduced in Section 3.1, the constitutive relation \( \mu \) can be found with the multistep and continuous PINN methods as described in Sections 2.3 and 2.5, respectively. Note that, in general, \( \mu \) can be a function of all components of the state vector \( y(t) = [X(t), S(t), V(t)]^T \) and, therefore, its neural network approximation \( \mu^\text{NN}(y(t)) \) can have all the components of \( y(t) \) as input. In this work, we assume that it is known that \( \mu \) depends only on the state \( S(t) \) as, e.g., in the Haldane model (13). This allows us to define the network \( \mu^\text{NN}(S(t)) \) with a one-dimensionless input space that requires a smaller amount of data to train without affecting the accuracy of the prediction.

4. NUMERICAL EXPERIMENTS

Here, we present numerical results for the FBR model. As the ground-truth, we assumed the constitutive relation (13) with the parameters \( k_1 = 1, K_M = 10, K_i = 0.1, \) and \( \mu^\ast = 5 \) (Dochain and Bastin, 1990). The synthetic datasets are generated as solutions of the governing equations (11) subject to the initial conditions of \( X_0 = 0.1, S_0 = 1, \) and \( V_0 = 10 \) (Dochain and Bastin, 1990). We consider two scenarios: (1) a simpler scenario with the constant flow rate \( F(t) = F_0 \) and input concentration \( S_{in}(t) = S_{in0} \), and (2) a more challenging scenario where \( F(t) = 0.25F_0 \sin(w_2t) + F_0 \) and \( S_{in}(t) = 0.5S_{in0} \sin(w_1t) + 1.5S_{in0} \) with \( F_0 = 0.1, S_{in0} = 3.5, w_1 = 1.0, \) and \( w_2 = 2.0 \).

In Table 1, we provide the summary of our numerical study, including the list of considered problems and the corresponding errors. The case-by-case study is given in Section 4.1 for time-independent \( F \) and \( S_{in} \) and Section 4.2 for time-dependent \( F \) and \( S_{in} \). We present this summary
TABLE 1: The relative error estimate of the states $X$, $S$, and $V$ with reference to the test data. The notation “disc” refers to using the discrete time multistep neural network method, while “cont” refers to the continuous time neural network. Here, “learn $f$” refers to the case when only the constitutive relation is unknown, and “learn $\mu$” refers to the case when only the constitutive relation is unknown. In Figs. 2, 3, 4, 5, 7, 10, and 11, $F_t = F_0 = 0.1, S_{in}(t) = S_{in0} = 3.5, F(t) = F_0 \sin (w_2 t) + 1.5F_0; S_{in}(t) = S_{in0} \sin (w_1 t) + 2.5S_{in0}$ for Fig. 12; and $F(t) = 0.25F_0 \sin (w_2 t) + F_0$ and $S_{in}(t) = 0.5S_{in0} \sin (w_1 t) + 1.5S_{in0}$ for Fig. 15. Training data are a single trajectory of 50 units of time for Fig. 2, two shorter trajectories of 25 units of time for Fig. 3, and one shorter trajectory of 25 units of time for Fig. 7. In Fig. 13, $F(t) = 0.25F_0 \sin (w_2 t) + F_0$ and $S_{in}(t) = 0.5S_{in0} \sin (w_1 t) + 1.5S_{in0}$, and the training data are a single trajectory of 50 units of time. Figure 14 has two training trajectories of 50 units of time each.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\varepsilon_X$</th>
<th>$\varepsilon_S$</th>
<th>$\varepsilon_V$</th>
<th>Training Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2: disc (learn $f$)</td>
<td>0.059</td>
<td>0.17</td>
<td>0.0012</td>
<td>One long trajectory</td>
</tr>
<tr>
<td>Fig. 4: disc (learn $\mu$)</td>
<td>0.0243</td>
<td>0.0929</td>
<td>1.21e–015</td>
<td>One long trajectory</td>
</tr>
<tr>
<td>Fig. 3: disc (learn $f$)</td>
<td>0.034</td>
<td>0.137</td>
<td>0.0037</td>
<td>Two long trajectories</td>
</tr>
<tr>
<td>Fig. 5: disc (learn $\mu$)</td>
<td>0.0052</td>
<td>0.02</td>
<td>1.02e–015</td>
<td>Two long trajectories</td>
</tr>
<tr>
<td>Fig. 13: disc (learn $f$)</td>
<td>0.131</td>
<td>0.697</td>
<td>0.0028</td>
<td>One long trajectory</td>
</tr>
<tr>
<td>Fig. 14: disc (learn $f$)</td>
<td>0.05</td>
<td>0.197</td>
<td>0.003</td>
<td>One long trajectory</td>
</tr>
<tr>
<td>Fig. 12: disc (learn $\mu$)</td>
<td>0.0047</td>
<td>0.022</td>
<td>4.86e–08</td>
<td>One long trajectory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Continuous PINN Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 7: cont (learn $f$)</td>
</tr>
<tr>
<td>Fig. 10: cont (learn $\mu$)</td>
</tr>
<tr>
<td>Fig. 11: cont (learn $\mu$)</td>
</tr>
<tr>
<td>Fig. 15: cont (learn $\mu$)</td>
</tr>
</tbody>
</table>

The relative errors are calculated as

$$\varepsilon_q = \frac{\|q^{ref} - q^{NN}\|}{\|q^{ref}\|}$$

where $q$ is the quantity of interest. By $\| \cdot \|$, we denote the $L_2$ norm over the entire trajectory. The reference solution [$q^{ref}$ in Eq. (16) and “Test” data in figures] and the training data (“Train” data in figures) are obtained by solving the system (11) using the $M = 1$ step Adams-Moulton time-stepping method. The chosen time step is 0.005, and we used the odeint package within scipy (the “LSODA” algorithm). In all PINN models, we use fully connected feed-forward neural networks with six hidden layers of size 256 that are trained with 25,000 epochs (iterations). The neural networks are trained using the adaptive moment estimation (ADAM) optimizer (Kingma and Ba, 2014).

Solutions obtained by discrete multistep neural network and continuous PINN are represented as “NN learn” in the figures and $q^{NN}$ in Eq. (16). In Table 1, we compare the errors corresponding to the results obtained using discrete multistep neural network and continuous
PINN approaches for cases when the governing equations $f$ are unknown and similarly for the cases when and the constitutive relation ($\mu$) is unknown. In summary, the table shows that incorporation of prior knowledge helps the accuracy of the trained PINN (we discuss this point in more detail after presenting the numerical results). The detailed description of the figures corresponding to the errors in Table 1 are provided in the following sections.

4.1 Time-Independent $F$ and $S_{1n}$

We begin the presentation of our results with the case of time-independent $F$ and $S_{1n}$. The results for the time-dependent case will be shown in Section 4.2.

4.1.1 Discrete Time—Unknown Governing Equations

Figures 2 and 3 show the numerical results obtained using the multistep neural network model when the dynamics $f$ are completely unknown. Figure 2(a) shows the training data (in red), test data (in dashed blue), and the prediction of the multistep neural network model with learned dynamics $f^{\text{NN}}$ (in dashed black) of the states $X(t)$, $S(t)$, and $V(t)$. Note that the initial conditions corresponding to the training data ($X_0 = 0.1$, $S_0 = 1$, and $V_0 = 10$) and the test data ($X_0 = 0.12$, $S_0 = 12$, and $V_0 = 10$) are different. The training data are equally spaced in a time interval of 50 units of time.

Although the test data (in blue) and the neural-network-based predictions (in black) match very well for the states $X(t)$ and $V(t)$, they deviate slightly for the state $S(t)$. This is because there is an order of difference in magnitude for $X(t)$ and $S(t)$ compared to that of $V(t)$. This difference in magnitude makes it harder to train a network that learns unknown governing equations. As we show in later plots (Fig. 3) this deviation can be remedied by using additional time series from different initial conditions (even if the trajectories are shorter).

![Fig. 2: Unknown governing equations—multistep neural network (one trajectory used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using multistep neural networks. (b) RHS of the ODE describing the evolution dynamics.](image-url)
FIG. 3: Unknown governing equations—multistep neural network (two shorter trajectories used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using multistep neural networks. (b) RHS of the ODE describing the evolution dynamics.

Figure 2(b) shows the training and learned and test evaluations of the right-hand-side function $f$ describing the dynamics. We observe that there is a better agreement between learned and test solutions than that between learned and test dynamics (RHS of the equations). This agreement is because the solution as a function of time is smoother than that of the dynamics $f$.

In Fig. 3, we use two sets of training data (red and magenta) obtained with different initial conditions ([0.1, 1.0, 10.0] and [0.2, 1.5, 15.0]) but for a shorter time duration (25 units of time) than that of the test data obtained with initial conditions ([0.15, 1.2, 12.0]). On the other hand, we again compute neural-network-based predictions for 50 units of time. We do this to highlight the neural network’s ability to learn the correct physics so that it can be used to simulate the dynamics beyond the time interval used for training. Furthermore, we see that the addition of a second dataset slightly improves the accuracy of the predicted state vector [Fig. 3(a)] as well as the accuracy of the predicted dynamics [Fig. 3(b)].

### 4.1.2 Discrete Time—Constitutive Relation

In Fig. 4, we show the numerical results obtained with the multistep neural network model when the equations governing the dynamics $f$ are known but the constitutive relation $\mu(y)$ is unknown. Here, we model $\mu(y)$ as a neural network that takes the state vector $y$ as input and outputs $\mu_{NN}(y)$.

Figure 4(a) shows the training data (in red), test data (in dashed blue), and predictions of the multistep neural-network model (in dashed black) of the states $X(t)$, $S(t)$, and $V(t)$. Note that the initial conditions of the test and neural-network-learned dynamics are different from those of the training data. Because we have more information about the system, namely the equations governing the dynamics, the multistep neural network model works very well and its predictions (in dashed black) match well with the test data (dashed blue). Figures 4(b) and 4(c) show the learned constitutive relation $\mu$ as a function of the state $S$ and time $t$, respectively. These are compared with the ground-truth constitutive relation values from the Haldane model. There is good agreement between the learned constitutive relation $\mu_{NN}(S)$ and the Haldane model (13).
FIG. 4: Unknown constitutive relation—multistep neural network (one trajectory used for training). (a) Solution of FBR model computed from exact dynamics and with learned dynamics using multistep neural networks. (b) Coefficient $\mu(t)$ vs. state $S(t)$. (c) Coefficient $\mu(t)$ vs. time $t$.

Figure 5 shows similar plots as those in Fig. 4 with two sets of training data (shown in red and magenta) that extend to shorter time (25 units of time) compared to the length of the test data and learned predictions (50 units of time). Once again, we can observe very good agreement between the neural-network-based predictions and the test data [Fig. 5(a)]. The same applies to the agreement between the learned constitutive relation and ground truth from the Haldane model [Figs. 5(b) and 5(c)]. In the multistep NN approach we show various cases where the training data are obtained as the solution of Eq. (1) computed with different initial conditions and for different time duration (single trajectory of longer span, two trajectories of shorter span, etc.). It can be seen that using more trajectories improves the accuracy of the solution (e.g., Fig. 2 vs. Fig. 3 or Fig. 4 vs. Fig. 5). These cases demonstrate the flexibility of the multistep neural network approach and that the neural network can be trained to be more accurate as we include more data.
4.1.3 Continuous Time—Unknown Governing Equations

Figure 6 shows the results of learning unknown governing equations in continuous time as described in Section 2.4. The system state training data are displayed in red, and the learned solution from $y^{\text{NN}}$ is denoted by dashed blue lines. Here, the initial conditions ([0.1, 1.0, 10.0]) for the system dynamics are the same for both training dynamics and neural network learned dynamics. However, the time duration of the training dynamics (25 units of time) is shorter than that of the test and neural network learned dynamics. The neural network $y^{\text{NN}}$ acts as an interpolation function—the neural network learned solution agrees very well with the test solution within the interpolation range (from 0 to 25 units of time) and deviates in the extrapolation range (from 25 to 50 units of time), whereas $f^{\text{NN}}$ deviates from the correct dynamics throughout the solution range.
This approach also does not work for test data with different initial conditions than those of the training data. To address this issue, we obtain \( \dot{y} \) by solving the ODE using the odeint package within scipy (“LSODA” algorithm) with RHS \( f_{NN}(y) \) learned from the training dataset. However, the neural network \( f_{NN}(y) \) obtained with this approach is not accurate and results in rapid error accumulation. Figure 7(a) shows in dashed black the solution obtained by solving

![Figure 6](image1.png)

**FIG. 6:** Unknown governing equations—continuous time (one shorter trajectory with the same initial condition is used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using continuous time neural networks. (b) RHS of the ODE describing the evolution dynamics.

![Figure 7](image2.png)

**FIG. 7:** Unknown governing equations—continuous time (one shorter trajectory with a different initial condition is used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using continuous time neural networks. (b) RHS of the ODE describing the evolution dynamics.
the governing equation using $f^{NN}(y)$ as the RHS function. Figure 7(b) shows $f^{NN}$ (in dashed black) as a function of time. Finally, Fig. 8 shows the RHS $f$ as a function of the system states. This plot also confirms that $f^{NN}$ (in dashed black) does not approximate the RHS function accurately.

### 4.1.4 Continuous Time—Constitutive Relation

Finally, Fig. 9 shows the results from the continuous PINN method described in Section 2.5 for learning the constitutive relation $\mu$. We can see that the neural network learned solution [Fig. 9(a)] and the learned constitutive relation [Figs. 9(b) and 9(c)] agree very well with those of the test dynamics and the true constitutive relation, respectively. Here, we use the neural network $y^{NN}$, which is good for interpolation to obtain the neural network learned solution. As we can see from Figs. 9(b) and 9(c), the learned constitutive relation process using the neural network model $\mu^{NN}$ matches very well with that of the true constitutive relation process $\mu$.

**FIG. 8:** Unknown governing equations—continuous time: RHS $f$ of the FBR model computed from exact dynamics and with learned dynamics as a function of the system state.

*Journal of Machine Learning for Modeling and Computing*
For different initial conditions and time duration beyond those of the training data, we can solve the governing equations (11) with $\mu^{NN}$ as the constitutive relation process. This idea is further illustrated in Fig. 10, in which the training solution data (shown in solid red) is obtained with initial conditions $[0.1, 1.0, 10.0]$ for a shorter time duration (25 units of time), whereas the test solution (shown in dashed blue) is obtained with a different set of initial conditions $[0.15, 1.2, 12.0]$ for a longer duration (50 units of time). Figure 10(a) shows the solution (in dashed black) obtained by solving governing equations (11) with $\mu^{NN}(y)$ as the constitutive relation process. We can see that this solution (in dashed black) agrees very well with the test solution (in dashed blue). Figure 11 shows similar results when the trajectory of the training data is of the same length (50 units of time) as that of the test data. We can observe that the accuracy of the solution increased due to additional training data. The errors corresponding to unknown governing equations and unknown constitutive relation are given in Tables 2 and 3, respectively.
FIG. 10: Unknown constitutive relation—continuous time (one shorter trajectory and different initial conditions used for training). Solution obtained by solving governing equations with $\mu^{\text{NN}}$ as the constitutive relation. (a) Solution of the FBR model computed from exact dynamics and with learned dynamics. (b) Coefficient $\mu(t)$ vs. state $S(t)$. (c) Coefficient $\mu(t)$ vs. time $t$.

4.2 Time-Dependent $F$ and $S_{\text{in}}$

In this section, we show the numerical results for a nonlinear time-inhomogeneous ODE model of the FBR, where the parameters $F(t)$ and $S_{\text{in}}(t)$ are time dependent. These ODEs have more complex dynamics than the time-homogeneous ODEs considered in the previous section, which could be harder to learn with the neural networks. Here, we show the numerical results for three cases, namely using the discrete multistep neural network method for learning (1) unknown $\mu(S)$ and (2) unknown $f$ and continuous PINNs for learning (3) unknown $\mu$. We demonstrated in the previous section that continuous PINNs do not work well for learning $f$ for constant $F$ and $S_{\text{in}}$, and we do not consider continuous PINNs for learning $f$ here.
A Comparative Study of Physics-Informed Neural Network Models

FIG. 11: Unknown constitutive relation—continuous time (one trajectory and different initial conditions used for training). Solution obtained by solving governing equations with $\mu^{NN}$ as a constitutive relation). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics. (b) Coefficient $\mu(t)$ vs. state $S(t)$. (c) Coefficient $\mu(t)$ vs. time $t$.

Figure 12 shows a multistep neural network estimate of the unknown constitutive relation $\mu(S)$, when $F(t) = F_0 \sin(w_2 t) + 1.5F_0$ and $S_{in}(t) = S_{in0} \sin(w_1 t) + 2.5S_{in0}$. Table 3 shows the relative error in the estimated $\mu$. Unlike the time-homogeneous ODE solution, which is very smooth, the solution of the time-inhomogeneous ODEs is very oscillatory. Figure 12 and Table 3 show that the multistep neural network method for learning an unknown constitutive relation works well even when the solution is very oscillatory.

Figures 13 and 14 show the multistep neural network estimates of the unknown governing equations for different choices of training data (one trajectory and two trajectories, respectively), when $F(t) = 0.25F_0 \sin(w_2 t) + F_0$ and $S_{in}(t) = 0.5S_{in0} \sin(w_1 t) + 1.5S_{in0}$. We can also see that the results for time-inhomogeneous ODEs with the time-dependent $F(t)$ and $S_{in}(t)$

Volume 3, Issue 2, 2022
TABLE 2: The relative error estimate of the states $X$, $S$, and $V$ and the corresponding RHS functions $f_1$, $f_2$, and $f_3$ with reference to the test data when the mathematical model of the dynamical system is completely unknown. The notation “disc” refers to using the discrete time multistep neural network method, while “cont” refers to the continuous time neural network. In Figs. 2, 3, and 7, $F_t = F_0 = 0.1$, $S_{in}(t) = S_{in0} = 3.5$. Training data is a single trajectory of 50 units of time for Fig. 2, two shorter trajectories of 25 units of time for Fig. 3, and one shorter trajectory of 25 units of time for Fig. 7. In Fig. 13, $F(t) = 0.25F_0\sin(w_2t) + F_0$ and $S_{in}(t) = 0.5S_{in0}\sin(w_1t) + 1.5S_{in0}$, and the training data is a single trajectory of 50 units of time. Figure 14 has two training trajectories of 50 units of time.

<table>
<thead>
<tr>
<th>Unknown Governing Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Fig. 2: disc</td>
</tr>
<tr>
<td>Fig. 3: disc</td>
</tr>
<tr>
<td>Fig. 7: cont</td>
</tr>
<tr>
<td>Fig. 13: disc</td>
</tr>
<tr>
<td>Fig. 14: disc</td>
</tr>
</tbody>
</table>

TABLE 3: The relative error estimate of the states $X$, $S$, and $V$ and the corresponding constitutive relation $\mu(X,S,V)$ obtained using the multistep neural network and continuous PINN methods with reference to the test data when the mathematical model of the dynamical system is known but the constitutive relation $\mu(X,S,V)$ is unknown. $F_t = F_0 = 0.1$, $S_{in}(t) = S_{in0} = 3.5$ for Figs. 4, 5, 10 and 11; $F(t) = F_0\sin(w_2t) + 1.5F_0$ and $S_{in}(t) = S_{in0}\sin(w_1t) + 2.5S_{in0}$ for Fig. 12 and $F(t) = 0.25F_0\sin(w_2t) + F_0$; and $S_{in}(t) = 0.5S_{in0}\sin(w_1t) + 1.5S_{in0}$ for Fig. 15.

<table>
<thead>
<tr>
<th>Unknown Constitutive Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>Fig. 4: disc</td>
</tr>
<tr>
<td>Fig. 5: disc</td>
</tr>
<tr>
<td>Fig. 10: cont</td>
</tr>
<tr>
<td>Fig. 11: cont</td>
</tr>
<tr>
<td>Fig. 12: disc</td>
</tr>
<tr>
<td>Fig. 15: cont</td>
</tr>
</tbody>
</table>

are less accurate for smaller training data (Fig. 13) and significantly improve with larger data (Fig. 14). Figure 15 shows the continuous PINN estimate of the unknown constitutive relation $\mu(S)$ of the time-inhomogeneous ODE with $F(t) = 0.25F_0\sin(w_2t) + F_0$ and $S_{in}(t) = 0.5 \times S_{in0}\sin(w_1t) + 1.5S_{in0}$. Figures 12 and 15 and the relative errors in Table 3 show that the discrete multistep neural network method works much better than the continuous PINN method. This observation is consistent with the results observed in the previous section for time-homogeneous ODEs.
FIG. 12: Unknown constitutive relation—multistep neural network with \( F(t) = F_0 \sin (w_2 t) + 1.5F_0 \) and \( S_{in}(t) = S_{in0} \sin (w_1 t) + 2.5S_{in0} \) (one trajectory and different initial conditions used for training). Solution obtained by solving governing equations with \( \mu_{NN} \) as the constitutive relation. (a) Solution of the FBR model computed from exact dynamics and with learned dynamics. (b) Coefficient \( \mu(t) \) vs. state \( S(t) \). (c) Coefficient \( \mu(t) \) vs. time \( t \).

Table 2 summarizes the relative error values of the multistep and continuous PINN estimates of \( X, S, V, f_1, f_2, \) and \( f_3 \) for a problem when \( f \) is unknown. Here, we show the relative errors for the discrete time multistep neural network method with \( F_i = F_0 = 0.1, S_{in}(t) = S_{in0} = 3.5, \) and training data of a single trajectory of 50 units of time (this case is described in Section 2.2, Fig. 2), and two shorter trajectories of 25 units of time (Fig. 3) and for the continuous time neural network with one shorter trajectory of 25 units of time (Section 2.5, Fig. 7). The table also shows the multistep neural network method with \( F(t) = 0.25F_0 \sin (w_2 t) + F_0 \) and \( S_{in}(t) = 0.5S_{in0} \sin (w_1 t) + 1.5S_{in0} \) and training data for a single trajectory of 50 units of time (results are shown in Fig. 13) and two training trajectories of 50 units of time (Fig. 14).
FIG. 13: Unknown governing equations—multistep neural network with $F(t) = 0.25 F_0 \sin (w_2 t) + F_0$ and $S_{in}(t) = 0.5 S_{in0} \sin (w_1 t) + 1.5 S_{in0}$ (one trajectory used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using multistep neural networks. (b) RHS of the ODE describing the evolution dynamics.

FIG. 14: Unknown governing equations—multistep neural network with $F(t) = 0.25 F_0 \sin (w_2 t) + F_0$ and $S_{in}(t) = 0.5 S_{in0} \sin (w_1 t) + 1.5 S_{in0}$ (two trajectories used for training). (a) Solution of the FBR model computed from exact dynamics and with learned dynamics using multistep neural networks. (b) RHS of the ODE describing the evolution dynamics.

Table 3 provides the relative errors of the PINN estimates of $\mu$, $X$, $S$, and $V$ for a problem with unknown $\mu$. Here, we show the relative errors for the discrete time multistep neural network and continuous PINN methods with $F_t = F_0 = 0.1$, $S_{in}(t) = S_{in0} = 3.5$, $F_t = F_0 = 0.1$, $S_{in}(t) = S_{in0} = 3.5$ for Figs. 4, 5, 10 and 11; $F(t) = F_0 \sin (w_2 t) + 1.5 F_0$
FIG. 15: Unknown constitutive relation—continuous time with \( F(t) = 0.25F_0 \sin (w_2 t) + F_0 \) and \( S_{in}(t) = 0.5S_{in0} \sin (w_1 t) + 1.5S_{in0} \) (one trajectory and different initial conditions used for training). Solution obtained by solving governing equations with \( \mu^{NN} \) as the constitutive relation. (a) Solution of the FBR model computed from exact dynamics and with learned dynamics. (b) Coefficient \( \mu(t) \) vs. state \( S(t) \). (c) Coefficient \( \mu(t) \) vs. time \( t \).

and \( S_{in}(t) = S_{in0} \sin (w_1 t) + 2.5S_{in0} \) for Fig. 12; and \( F(t) = 0.25F_0 \sin (w_2 t) + F_0 \) and \( S_{in}(t) = 0.5S_{in0} \sin (w_1 t) + 1.5S_{in0} \) for Fig. 15.

We can see from the relative errors in Tables 2 and 3 that the multistep neural network method for learning unknown constitutive relations (\( \mu \)) is more accurate than other cases considered.

Our numerical tests show that incomplete knowledge (unknown constitutive relation, but known governing equations) is better than ignorance (completely unknown governing equations). More specifically, we find that learning the dynamics with unknown constitutive relationship \( \mu \) and a known general form of \( f \) is more accurate than learning the dynamics when the form of \( f \) is unknown. We find that the discrete multistep neural network method works better
5. CONCLUSIONS

We have used both discrete and continuous time PINN methods for learning unknown governing equations or constitutive relations of a system of ODEs. We have applied this framework to the case of an FBR modeled with (1) time-homogeneous and (2) time-inhomogeneous nonlinear ODEs. The processes in bioreactors are very complex due to continuously changing biological and chemical reaction kinetics and the nonlinear dependence of the constitutive relation on the system states. Our numerical results suggest that the discrete time framework is effective in training accurate neural-network-based estimators for the unknown system dynamics or constitutive relations. The continuous time framework is effective only for training accurate neural-network-based estimators for unknown constitutive relations. Our results corroborate the well-known lesson from scientific computing that whatever information we may have about the dynamics (i.e., the structure of $f$) should be used. In particular, if we know the dynamics of a system up to a constitutive relation, we can obtain more accurate predictions if we train a neural network to represent only the constitutive relation instead of the full dynamics. The contributions of this work are (1) comparing the applicability and formulation of different physics-informed machine learning approaches for learning unknown governing equations and constitutive relations of the dynamical system; (2) formulating and training the neural network models so that they accurately simulate the dynamics with different initial conditions than those of the training data; (3) demonstrating numerical tests for a highly nonlinear dynamical system (fedbatch bioreactor model) in which the constitutive relation [$\mu(S(t))$] depends on the system state $S(t)$ in a nonlinear manner.

In the considered FBR model, $\mu$ is a function of one variable ($S$) while $f$ is a function of three variables ($X, S, V$). Therefore, it should be expected that in order to achieve the same accuracy, more data (in this case, trajectories corresponding to different initial conditions) are needed when $f$ is unknown than when $\mu$ is unknown. In the considered examples, we used several trajectories to train multistep PINN models. However, the continuous PINN can also be trained with several trajectories, which we have not considered in this paper. In such a case, multiple neural networks $\mu_{NN}(t)$ corresponding to each trajectory $y_i(t)$ must be defined and trained jointly with the $f_{NN}(y)$ or $\mu_{NN}$ neural network. The existing analysis of PINN for certain types of differential equations (Shin et al., 2020) suggests that the accuracy of the continuous PINN method should increase with the increasing neural network sizes and the number of collocation points $N_f$. The above points call for more analysis of the PINN methods for learning equations and constitutive relationships. Of particular interest are systems with larger state spaces and driven by stochastic forces. However, the results in this work clearly show that given the same amount of data, a given PINN method can learn more accurately the constitutive relationship $\mu$ than the dynamics $f$ when $f$ is known up to the constitutive relationship. The current comparison study provides guidance for choosing the appropriate PINN approach based on the type of problem and availability of the data. Our results show that the discrete multistep approach is more effective for most cases. However, this approach requires availability of data at uniform time intervals. Also, the accuracy of the discrete multistep approach may decrease if the time between measurements is larger than the time step requirements in the discretization scheme. Continuous PINN models are appropriate when the data are obtained from experiments at nonuniform intervals. One of our conclusions, i.e., that knowing more about the model helps, is lost a bit in several approaches.
that have appeared in the literature where unknown dynamics are treated as black-box: see, e.g., neural ODEs. So, in our opinion, this manuscript offers a service to the community by showing that even for a system of small dimensionality, the inclusion of any partial knowledge is helpful.

ACKNOWLEDGMENTS

The material presented here is based upon work supported by Pacific Northwest National Laboratory (PNNL), “Deep Learning for Scientific Discovery Investment,” and the Department of Energy Advanced Scientific Computing Research program as part of the “Uncertainty Quantification For Complex Systems Described by Stochastic Partial Differential Equations” project. PNNL is operated by Battelle for the Department of Energy under Contract DE-AC05-76RL01830. P.P. acknowledges support from the Department of Energy under the Advanced Scientific Computing Research program (Grant No. DE-SC0019116).

REFERENCES


