MODELING UNKNOWN DYNAMICAL SYSTEMS WITH HIDDEN PARAMETERS

Xiaohan Fu,[1,*] WeiZe Mao,[2] Lo-Bin Chang,[1] & Dongbin Xiu[2]

[1]*Department of Statistics, The Ohio State University, Columbus, Ohio 43210, USA*

[2]*Department of Mathematics, The Ohio State University, Columbus, Ohio 43210, USA*

*Address all correspondence to: Xiaohan Fu, Department of Statistics, The Ohio State University, Columbus, OH 43210, USA, E-mail: fu.688@osu.edu

Original Manuscript Submitted: 4/25/2022; Final Draft Received: 9/11/2022

We present a data-driven numerical approach for modeling unknown dynamical systems with missing/hidden parameters. The method is based on training a deep neural network (DNN) model for the unknown system using its trajectory data. A key feature is that the unknown dynamical system contains system parameters that are completely hidden, in the sense that no information about the parameters is available through either the measurement trajectory data or our prior knowledge of the system. We demonstrate that by training a DNN using the trajectory data with sufficient time history, the resulting DNN model can accurately model the unknown dynamical system. For new initial conditions associated with new, and unknown, system parameters, the DNN model can produce accurate system predictions over longer time.

KEY WORDS: *deep neural network, hidden parameters, unknown dynamical systems*

1. INTRODUCTION

There has been a growing interest in learning unknown dynamical systems using observational data. A common approach is to construct a mapping from the state variables to their time derivatives. Various numerical approximation techniques can be used to construct such a mapping. These include sparse regression, polynomial approximations, model selection, and Gaussian process regression (Brunton et al., 2016; Mangan et al., 2017; Raissi et al., 2017a; Rudy et al., 2017; Schaeffer et al., 2018; Wu et al., 2019; Wu and Xiu, 2019), to name a few. More recently, deep neural networks (DNNs) have been adopted to construct the mapping. Studies have empirically demonstrated the ability of DNN to model ordinary differential equations (ODEs) (Qin et al., 2019; Raissi et al., 2018; Rudy et al., 2019) and partial differential equations (PDEs) (Long et al., 2018a,b; Raissi, 2018; Raissi et al., 2017b,c; Sun et al., 2019). A notable recent development is to model the mapping between two system states separated by a short time (Qin et al., 2019). This approach essentially models the underlying flow map of the unknown system, and is notably different from the earlier approach of modeling the map between the state variables and their time derivatives. The flow map based approach eliminates the need for temporal derivative data, which are often difficult to acquire in practice and subject to larger errors. Once an accurate DNN model for the flow map is constructed, it can be used as an evolution operator to conduct system predictions. In particular, the residual network (ResNet), developed in the image analysis community (He et al., 2016), was found to be suitable for recovering the flow map (Chen and Xiu, 2021; Qin et al., 2019). Since its introduction (Qin et al., 2019), the flow map based DNN modeling approach has been extended to modeling of non-autonomous dynamical systems (Qin et al., 2021a), parametric dynamical systems (Qin et al., 2021b), and partially observed dynamical systems (Fu et al., 2020), as well as PDEs (Wu and Xiu, 2020).

The focus of this paper is on a different type of data-driven modeling problem. We assume that the target unknown dynamical system is parameterized by a set of parameters that are completely hidden, in the sense that no prior knowledge about the form, or even the existence, of the parameters is available. The only available information of the dynamical system is in the form of the trajectory data of its state variables. The trajectory data are also parameterized, in an unknown manner, by the hidden parameters. Our goal is to construct a predictive model of the underlying dynamical system by using only the trajectory data. Once the predictive model is constructed, it shall be able to produce accurate predictions of the system states over time, for any given initial conditions that are parameterized by the hidden parameters in an unknown manner. The distinct feature of this work is that no knowledge of the system parameters is assumed to be available, not in the (unknown) governing equations or in the trajectory data (for training or prediction). This is often the case for many complex systems, whose dynamics are controlled by a large, and sometimes unknown, number of parameters that are not measurable.

The method proposed in this paper is motivated by the work of Fu et al. (2020), who studied the modeling of partially observed dynamical systems where trajectory data of only a subset of the state variables are available. While the celebrated Mori–Zwanzig (MZ) formulation (Mori, 1965; Zwanzig, 1973) defines a closed-form dynamical system for the observed state variables, the MZ system is intractable for practical computations as it involves a memory integral of an unknown kernel function. Upon assuming a finite effective memory length, a DNN structure with explicit incorporation of "past memory" was proposed in Fu et al. (2020) and shown to be highly effective for learning and modeling partially observed systems. Compared to other DNN strutures with memory gates, e.g., LSTM, the DNN structure from Fu et al. (2020) is notably simpler and serves as a direct approximation of the MZ formulation.

In this paper, we adopt the DNN structure developed in Fu et al. (2020) and demonstrate that it can be used to model unknown dynamical systems with hidden parameters. The theoretical motivation is that the hidden parameters can be viewed as a set of unobserved state variables with trivial dynamics. Consequently the DNN structure from Fu et al. (2020) becomes applicable. Moreover, for long-term prediction accuracy and stability, we introduce a recurrent structure during network training. Once the DNN model is constructed, it is able to produce accurate system predictions over longer time, for any given initial conditions containing unknown hidden parameters.

2. SETUP AND PRELIMINARIES

Let us consider a dynamical system,

$$\frac{d\widetilde{\mathbf{x}}}{dt}(t;\boldsymbol{\alpha}) = \mathbf{f}(\widetilde{\mathbf{x}}, \boldsymbol{\alpha}), \qquad \widetilde{\mathbf{x}}(0;\boldsymbol{\alpha}) = \widetilde{\mathbf{x}}_0, \tag{1}$$

where $\widetilde{\mathbf{x}} \in \mathbb{R}^n$ are state variables and $\boldsymbol{\alpha} \in \mathbb{R}^d$ are system parameters. We assume that the form of the governing equations, which manifests itself via $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$, is unknown. More importantly, we assume that the information about the system parameters $\boldsymbol{\alpha}$ is not available. In fact, even the dimensionality d of $\boldsymbol{\alpha}$ can be unknown.

2.1 Learning Objective

We assume trajectory data are available for the state variables $\widetilde{\mathbf{x}}$. Let N_T be the total number of observed trajectories. For each ith trajectory, we have

$$\mathbf{X}^{(i)} = \left\{\widetilde{\mathbf{x}}\left(t_k^{(i)}\right)\right\}, \qquad k = 1, \dots, K^{(i)}, \quad i = 1, \dots, N_T, \tag{2}$$

where $\left\{t_k^{(i)}\right\}$ are discrete time instances at which the data are available, and $K^{(i)}$ is the total number of data entries in the ith trajectory. Note that each ith trajectory is associated with an initial condition $\widetilde{\mathbf{x}}_0^{(i)}$ and system parameters $\boldsymbol{\alpha}^{(i)}$, both of which are unknown.

Our goal is to construct an accurate numerical model, $\mathcal{M}$ for the system (1), by using the data set (2). More specifically, let

$$0 = t_0 < \cdots < t_N = T,$$

be a sequence of time instances with a finite horizon $T > 0$. This will be our prediction time stencil. We seek a predictive model $\mathcal{M}$ such that, for any given initial condition $\mathbf{x}_0$, which is associated with an unknown system parameter $\boldsymbol{\alpha}$, the model prediction is an accurate approximation of the true system, in the sense that

$$\mathcal{M}(t_k; \mathbf{x}_0, \boldsymbol{\alpha}) \approx \widetilde{\mathbf{x}}(t_k; \mathbf{x}_0, \boldsymbol{\alpha}), \qquad k = 1, \dots, N, \tag{3}$$

with satisfactory accuracy.

2.2 Related Study

Our topic is related to, and extends to, two recent studies on modeling dynamical systems. The first related study is on recovering unknown deterministic dynamical systems. When data of the state variables $\mathbf{x}$ are available, it was shown in Qin et al. (2019) that the residual network (ResNet) can be used to construct a predictive model. In fact, for autonomous systems, the ResNet based DNN model is an exact integrator of the underlying system. It is a one-step predictive model and consequently requires only trajectory data of two consecutive data entries. For parameterized systems, when the parameters $\boldsymbol{\alpha}$ are known from the trajectory data, the ResNet model can be modified to incorporate more input neurons to represent the system parameters $\boldsymbol{\alpha}$. See Qin et al. (2021b) for details.

Another related study is on modeling unknown dynamical systems with partially observed state variables. Let $\mathbf{x}^\top = (\mathbf{z}^\top, \mathbf{w}^\top)$ be the full set of state variables, where $\mathbf{z} \in \mathbb{R}^n$ is the subset of the state variables with available data, and $\mathbf{w} \in \mathbb{R}^d$ is the subset of missing variables. Based on the celebrated MZ formulation (Mori, 1965; Zwanzig, 1973), the evolution of $\mathbf{z}$ follows a generalized Langevin equation,

$$\frac{d}{dt}\mathbf{z}(t) = \mathbf{R}\big(\mathbf{z}(t)\big) + \int_0^t \mathbf{K}\big(\mathbf{z}(t-s), s\big)ds + \mathbf{F}(t, \mathbf{x}_0), \tag{4}$$

which involves a Markovian term $\mathbf{R}$, a memory integral with kernel $\mathbf{K}$ and a random term $\mathbf{F}$ involving the unknown initial condition. Upon making an assumption on finite effective memory, a discrete approximate MZ equation was proposed in Fu et al. (2020),

$$\frac{d}{dt}\hat{\mathbf{z}}(t)\bigg|_{t=t_n} = \mathbf{R}\big(\hat{\mathbf{z}}(t)\big)\big|_{t=t_n} + \mathbf{M}(\hat{\mathbf{z}}_{n-n_M}, \dots, \hat{\mathbf{z}}_{n-1}, \hat{\mathbf{z}}_n), \tag{5}$$

where $\hat{\mathbf{z}}_n = \hat{\mathbf{z}}(t_n)$ is the solution at time $t_n = n\Delta$ over a constant time step Δ; n_M is the number of memory terms. A DNN structure to explicitly account for the memory terms was then proposed in Fu et al. (2020) and shown to be highly effective and accurate.

3. METHOD DESCRIPTION

In this section, we describe the details of our proposed deep learning approach for systems with hidden parameters. The distinct feature of our work is that not only are the system equations unknown, the associated system parameters remain completely unknown throughout the modeling and prediction process.

3.1 Motivation

For the unkonwn system with missing/hidden parameters (1), one can view it in an alternative form,

$$\begin{cases} \dfrac{d\widetilde{\mathbf{x}}}{dt} = \mathbf{f}(\widetilde{\mathbf{x}}, \boldsymbol{\alpha}), \quad \widetilde{\mathbf{x}}(0) = \widetilde{\mathbf{x}}_0, \\ \dfrac{d\boldsymbol{\alpha}}{dt} = 0, \quad \boldsymbol{\alpha}(0) = \boldsymbol{\alpha}. \end{cases} \tag{6}$$

If one treats $\boldsymbol{\alpha}$ also as state variables with trivial dynamics and views $\widetilde{\mathbf{X}} = (\widetilde{\mathbf{x}}^\top, \boldsymbol{\alpha}^\top)^\top$ as the complete set of state variables, the data set (2) on $\widetilde{\mathbf{x}}$ then represents the data of a subset of the full variable set $\widetilde{\mathbf{X}}$. From this perspective, the memory based DNN structure, designed in Fu et al. (2020) for partially observed systems, becomes applicable. Hereafter we will employ the DNN structure of Fu et al. (2020) and modify it to suit our modeling needs.

3.2 Network Structure

Our basic DNN structure consists of a forward block and a recurrent block. For notational convenience, hereafter we shall assume a constance time step,

$$\Delta \equiv t_{k+1}^{(i)} - t_k^{(i)}, \qquad \forall k = 1, \dots, K^{(i)} - 1, \quad i = 1, \dots, N_T, \tag{7}$$

for all the trajectory data, as well as for the prediction time stencil. [Variable time steps can be readily incorporated into the DNN model as an additional input. See Qin et al. (2021b) for details.]

3.2.1 Forward Block

The forward block of our DNN model is similar to the DNN with memory model developed in Fu et al. (2020). The structure of the forward block is illustrated in Fig. 1, where we use the following notation

$$\mathbf{X}_{i:j} = \left(\mathbf{x}_i^\top, \dots, \mathbf{x}_j^\top\right)^\top, \qquad j \geq i, \tag{8}$$

to denote the concatenated vector of the state variables of consecutive indices from $\mathbf{x}_i$ to $\mathbf{x}_j$. The input layer of the DNN, $\mathbf{X}_{n-n_M:n}$, incorporates $(n_M + 1)$ state vectors $\mathbf{x}$, each of which has

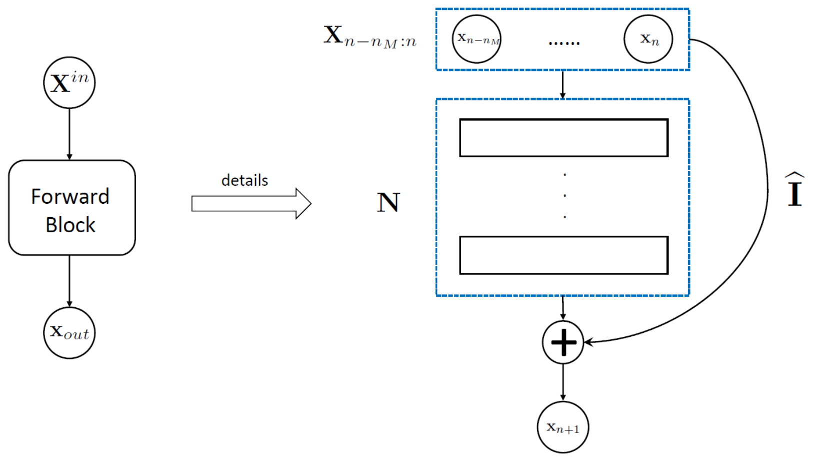


FIG. 1: Illustration of forward block

size n. The output layer incorporates a single state vector $\mathbf{x}$ of length n. A standard fully connected feedforward network (FFN) serves as the mapping from the input layer to the output layer. We use $\mathbf{N}$ to denote the mapping operator defined by the FFN. An operator $\widehat{\mathbf{I}}$ is introduced to the input layer and then applied to the output of the FFN. This is to reintroduce $\mathbf{x}_n$ before the DNN output layer to achieve the ResNet-like operation.

More specifically, the DNN input $\mathbf{X}^{\text{in}} = \mathbf{X}_{n-n_M:n} \in \mathbb{R}^D$, where the dimension D, i.e., the number of neurons in the input layer, is

$$D = n \times (n_M + 1). \tag{9}$$

The operator $\widehat{\mathbf{I}}$ is defined as a $(n \times D)$ matrix,

$$\widehat{\mathbf{I}} = [\mathbf{I}_n, 0, \dots, 0],$$

where the size $(n \times n)$ identity matrix $\mathbf{I}_n$ is concatenated by n_M zero matrices of size $(d \times d)$. The fully connected FFN connecting the input and output layers then defines a mapping operator

$$\mathbf{N}(\cdot; \Theta) : \mathbb{R}^D \to \mathbb{R}^n, \tag{10}$$

where Θ is the hyperparameter set associated with the FFN. Upon applying the operator $\widehat{\mathbf{I}}$ to the input and reintroducing $\mathbf{x}_n$ at the output of the FFN operation, our DNN model defines the following operation,

$$\mathbf{x}^{\text{out}} = \left[\widehat{\mathbf{I}} + \mathbf{N}\right](\mathbf{X}^{\text{in}}), \tag{11}$$

which in turn can be written as

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{N}(\mathbf{x}_n, \mathbf{x}_{n-1}, \dots, \mathbf{x}_{n-n_M}; \Theta), \qquad n \geq n_M. \tag{12}$$

We remark that $n_M \geq 0$ is the number of memory steps included in our DNN model. Let $T_M = n_M \times \Delta$. This shall be the length of the effective memory, a concept introduced in Fu et al. (2020). The choice of T_M is problem dependent and requires certain prior knowledge/experience about the underlying system. Sometimes trial and error is also necessary. Such practice is not uncommon in many aspects of numerical analysis, for example, choices of domain size and grid size. Note that $n_M = 0$ represents the memoryless case, which reduces the DNN back to the standard ResNet structure used for modeling the complete system Qin et al. (2019).

3.2.2 Recurrent Block

The forward DNN block discussed in the previous section is essentially the same DNN structure developed in Fu et al. (2020), for modeling systems with missing variables. In principle, it is also applicable for modeling systems with hidden parameters, as explained in Section 3.1. However, during our initial numerical experimentations, we have repeatedly discovered that it lacks sufficient long-term numerical stability. To mitigate the numerical instability, we thus introduce a recurrent structure, in conjunction with the forward block, in our final DNN model.

The structure of the recurrent block is illustrated in Fig. 2, where $n_R \geq 1$ is the number of recurrent steps. The trivial case of $n_R = 1$ reduces the DNN model to the forward block structure in the previous section. The recurrent blocks are to recursively apply the forward DNN block over n_R time steps and compute the loss function using the outputs of the n_R steps.

Using the notation (8), our final DNN model with n_R recurrent steps can then be defined as, for any time t_n with $n \geq n_M$,

$$\begin{cases} \mathbf{X}^{\text{in}} = \mathbf{X}_{n-n_M:n}, \\ \mathbf{x}_{k+1} = \left[\widehat{\mathbf{I}} + \mathbf{N}\right](\mathbf{X}_{k-n_M:k}), \qquad k = n, \dots, n+n_R, \\ \mathbf{X}^{\text{out}} = \mathbf{X}_{n+1:n+n_R}. \end{cases} \tag{13}$$

Note that the n_R forward blocks share the same parameter set Θ. In other words, it is the same forward block that is applied recurrently n_R times. The input of the entire DNN network is the same as that of the nonrecurrent forward block, $\mathbf{X}_{n-n_M:n} = \left(\mathbf{x}_{n-n_M}^\top, \dots, \mathbf{x}_n^\top\right)^\top$, $n_M + 1$ steps of solution vectors. The output of the DNN is a sequence of n_R steps of the outputs of the forward block.

We remark that, although the recurrent forward block is not a mathematical necessity, it is an important component from a practical point of view. Our extensive numerical experimentations, not only in this work but also in related work such as Fu et al. (2020), have indicated that

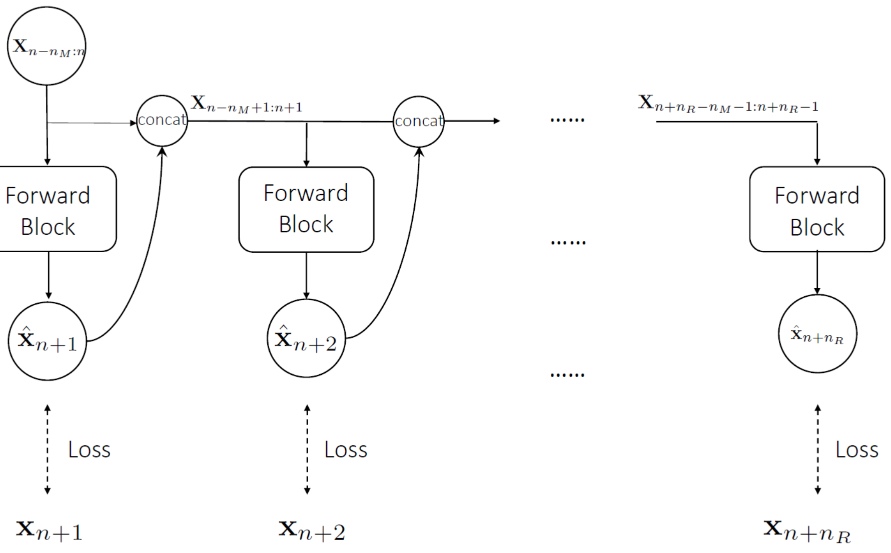


FIG. 2: Illustration of recurrent-forward-block structure with n_R recurrent steps

the use of recurrent forward block can significantly enhance numerical stability for long-term system predictions. Unfortunately, due to the lack of available mathematical analysis of the basic properties of DNN, we are unable to provide rigorous stability analysis of the proposed DNN model at this stage.

It is also worth noting that one may construct a many-to-many DNN mapping from the $n_M + 1$ memory terms in the "past" to the entire n_R terms in the "future." Such a construction does not perform as well as the proposed one-step recurrent forward structure. In addition, the many-to-many mapping is less flexible, as it does not readily provide a one-step forward prediction.

3.3 Network Training and Predictive Modeling

The DNN model (13) effectively defines a mapping

$$\mathbf{X}^{\text{out}} = \mathcal{N}(\mathbf{X}^{\text{in}}; \Theta), \tag{14}$$

where $\mathbf{X}^{\text{in}}$ consists of $n_M + 1$ steps of the state variables $\mathbf{x}$, and $\mathbf{X}^{\text{out}}$ consists of n_R steps of the state variables. Therefore, to train the DNN model, we require state variable trajectories of length at least $n_{tot} = n_M + n_R + 1$.

Let us assume that each ith trajectory in our data set (2) has its number of entries satisfying $K^{(i)} \geq n_{tot}$ entries. (In other words, the trajectories with a smaller number of entries are already eliminated from the data set.) We then randomly select a piece of n_{tot} number of consecutive entries from the trajectory and regroup them into two segments: the first $n_M + 1$ entries vs. the last n_R entries:

$$\left\{\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}\right\}, \tag{15}$$

where

$$\begin{aligned} \mathbf{X}^{(i)} &= \left[\mathbf{x}\left(t_k^{(i)}\right)^\top, \ldots, \mathbf{x}\left(t_{k+n_M}^{(i)}\right)^\top\right]^\top, \\ \mathbf{Y}^{(i)} &= \left[\mathbf{x}\left(t_{k+n_M+1}^{(i)}\right)^\top, \ldots, \mathbf{x}\left(t_{k+n_M+n_R}^{(i)}\right)^\top\right]^\top. \end{aligned} \tag{16}$$

This random selection procedure is repeated for all the N_T trajectories in the data set (2). Note that for each $i = 1, \ldots, N_T$ trajectory, it is possible to select more than one such grouping whenever $K^{(i)} > n_{tot}$. Upon conducting the random sequence selection for all the trajectories in (2), we obtain a collection of the grouping (15). After reordering all the selected groupings with a single index, we obtain the training data set for our DNN model,

$$\mathcal{X} = \{\mathbf{X}_j, \mathbf{Y}_j\}, \qquad j = 1, \ldots, J, \tag{17}$$

where J is the total number of data groupings. (Note that at this stage the information of the ith trajectory, from which the grouping $\{\mathbf{X}_j, \mathbf{Y}_j\}$ is originated, is not important.)

Our DNN model training is then conducted by minimizing the following mean squared loss:

$$\Theta^* = \underset{\Theta}{\operatorname{argmin}} \frac{1}{J} \sum_{j=1}^{J} \left\|\mathcal{N}(\mathbf{X}_j^{\text{in}}; \Theta) - \mathbf{Y}_j\right\|^2. \tag{18}$$

Upon finding the optimal network parameter Θ^*, we obtain our trained network model in the form of (13),

$$\mathbf{x}^{\text{out}} = \left[\widehat{\mathbf{I}} + \mathbf{N}(\cdot;\Theta^*)\right]\left(\mathbf{X}^{\text{in}}\right), \tag{19}$$

where the optimized parameter Θ^* will be omitted hereafter, unless confusion arises otherwise.

The trained DNN model defines a predictive model for the unknown dynamical system (1) with hidden parameters. It requires $n_M + 1$ initial conditions. Once given a sequence of $n_M + 1$ state variables $\mathbf{x}$, which are associated with unknown parameters $\boldsymbol{\alpha}$, the DNN model is able to conduct one-step prediction iteratively for the system state, corresponding to the same (and yet still unknown) parameters $\boldsymbol{\alpha}$. More specifically, the predictive scheme takes the following form: For any unknown hidden parameter $\boldsymbol{\alpha}$,

$$\begin{cases} \mathbf{x}_k = \mathbf{x}(t_k;\boldsymbol{\alpha}), \qquad k = 0,\ldots,n_M, \\ \mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{N}(\mathbf{x}_n, \mathbf{x}_{n-1}, \ldots, \mathbf{x}_{n-n_M};\Theta^*), \qquad n \geq n_M. \end{cases} \tag{20}$$

4. NUMERICAL EXAMPLES

In this section, we present four numerical examples to examine the performance of the proposed method. The examples include (1) a nonlinear pendulum system with two hidden parameters; (2) a larger linear system with 100 hidden parameters; (3) a nonlinear chemical reactor system with one hidden parameter that induces bifurcation in the system behavior; and (4) a nonlinear system for modeling cell signaling cascade with 12 hidden parameters. In all the examples, the underlying "true" models are known and used only to generate the training data sets. Note that in the training data sets, only the solution trajectories are recorded; the corresponding parameter values are not recorded. By doing so, the parameters in the true models remain completely hidden from the DNNs. To validate the trained DNN predictive models, we use the corresponding true models to generate a set of initial conditions that are not in the training data sets and with the associated parameter values hidden. The DNN predictive models are then used to produce system predictions over a longer time horizon and compared against the reference solutions generated by the true models.

In all the examples here, the time step is fixed at $\Delta = 0.02$. The number of memory steps n_M and recurrent steps n_R are problem dependent and determined numerically by gradually increasing the values till converged numerical results are obtained. Unless otherwise noted, the DNNs used in the examples consist of three hidden layers, with 30 neurons each, and have rectified linear unit (ReLU) activation function.

4.1 Example 1: Nonlinear Pendulum System

We first consider a small nonlinear system, the damped pendulum system,

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = -\alpha x_2 - \beta \sin x_1, \end{cases} \tag{21}$$

where the system parameters $\boldsymbol{\alpha} = (\alpha, \beta)^\top$ are treated as hidden and confined to a region $D_{\boldsymbol{\alpha}} = [-0.05, 0.15] \times [8, 10]$. The domain of interest for the state variables is set as $D_{\mathbf{x}} = [-0.5, 0.5] \times [-1.6, 1.6]$.

The memory step is tested for $n_M = 10$, 20, 40, 60, 80, 100, and 120, and the recurrent step is tested for $n_R = 1$, 10, 20, and 40. The model prediction errors at different memory steps and the recurrent steps are shown in Fig. 3. The prediction errors are computed using, ℓ_2-norm of the DNN model predictions against the reference solutions at time level $t = 100$, averaged over 100 simulations with random initial conditions and system parameters. We observe that the accuracy improvement over increasing n_M starts to saturate with $n_M \geq 100$. We also notice that a larger n_R produces better results consistently.

The DNN model predictive results with $n_M = 100$ and $n_R = 40$ are shown in Fig. 4, with two sets of arbitrarily chosen initial conditions and (hidden) system parameters. This corresponds to memory length $n_M \times \Delta = 0.4$, which is in fact rather short. We observe very good agreement between the DNN model predictions and the reference solutions for the long-term integrations up

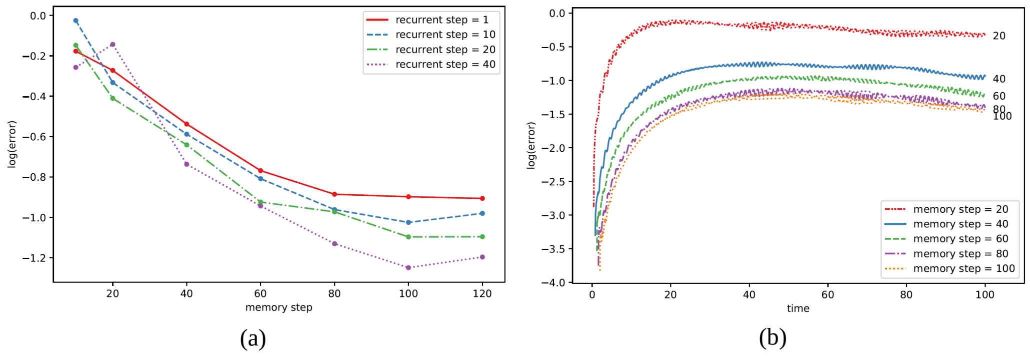


FIG. 3: Example 1. Model prediction errors at different memory steps and recurrent steps. (a) Errors vs. n_M, (b) errors over time for $n_R = 40$.

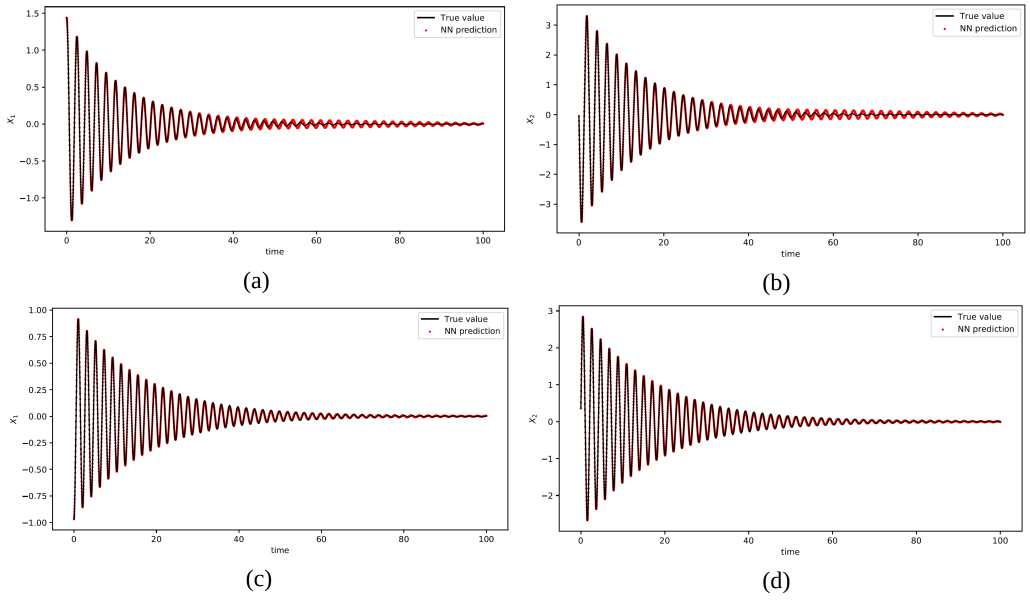


FIG. 4: Example 1. Model predictions up to $t = 100$ with $n_M = 100$ and $n_R = 40$ using two sets of arbitrary initial conditions and system parameters. (a), (c) x_1; (b), (d) x_2.

to $t = 100$. The corresponding numerical errors are plotted in Fig. 5, along with the comparison of the phase portraits.

4.2 Example 2: Larger Linear System

We now consider a larger linear system involving 20 state variables,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^{20}, \quad \mathbf{A} \in \mathbb{R}^{20\times 20},$$

where among the 400 entries of the coefficient matrix $\mathbf{A}$, we treat 100 of them as hidden parameters. More specifically, let us rewrite the system in terms of $\mathbf{x} = (\mathbf{p}; \mathbf{q})$, where $\mathbf{p} \in \mathbb{R}^{10}$ and $\mathbf{q} \in \mathbb{R}^{10}$ satisfy

$$\begin{cases} \dot{\mathbf{p}} = \Sigma_{11}\mathbf{p} + (\mathbf{I} + \Sigma_{12})\mathbf{q}, \\ \dot{\mathbf{q}} = -(\mathbf{I} + \Sigma_{21})\mathbf{p} - \Sigma_{22}\mathbf{q}. \end{cases} \tag{22}$$

Here, $\mathbf{I}$ is the identity matrix of size 10×10, and $\Sigma_{ij} \in \mathbb{R}^{10\times 10}$, $i = 1, 2$, $j = 1, 2$ are four coefficient matrices. We set three of the coefficient matrices to be known, with $\Sigma_{11} = \Sigma_{12} = 0$, and Σ_{22} with the entries listed in Appendix A. The 100 entries of the matrix Σ_{21} are treated as hidden parameters within the domain $[-0.05, 0.05]^{100}$. The domain of interest for the state variables is set as $[-2, 2]^{20}$. With a larger number of missing hidden parameters (compared to

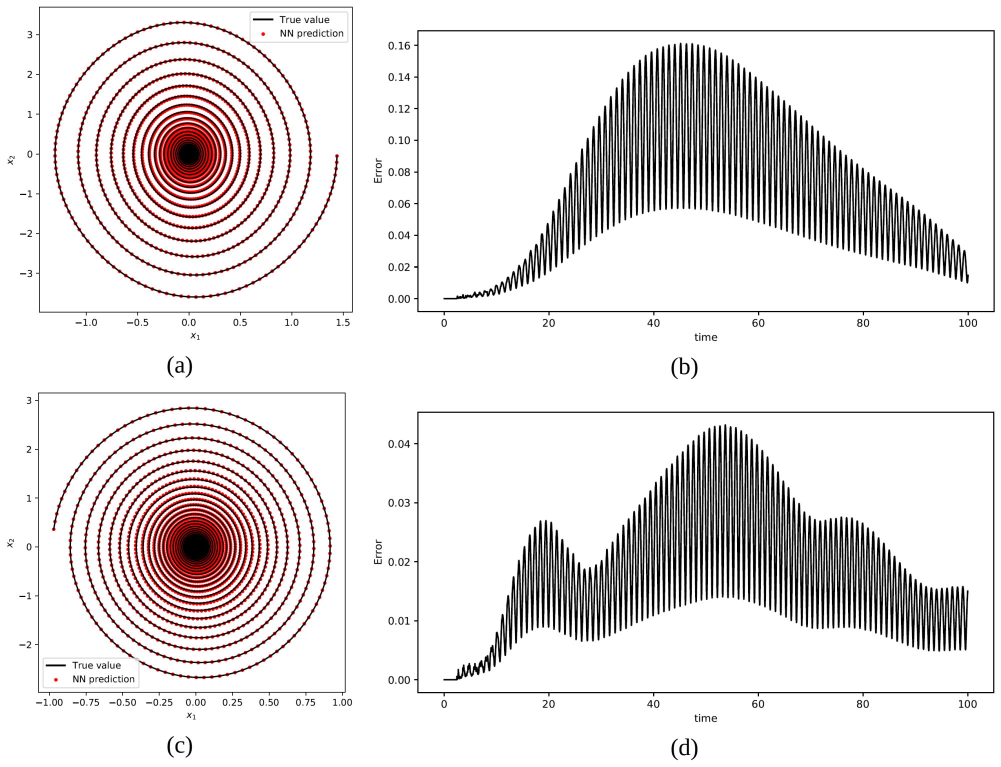


FIG. 5: Example 1. Model predictions and errors up to $t = 100$ with $n_M = 100$ and $n_R = 40$ using two sets of arbitrary initial conditions and parameters as in Fig. 4. (a), (c) phase plot; (b), (d) error.

Example 1), this problem requires longer memory length to construct an accurate DNN model. Memory steps of $n_M = 100$, 300, 500, 700, 900, 1100, 1300, and 1500 are tested. The results indicate the $n_M = 1300$ is sufficient to produce converged prediction results. The recurrent step is tested for $n_R = 1$ to $n_R = 5$. For this problem, the number of recurrent step does not induce a noticeable difference in the prediction. We therefore fix $n_R = 1$. The DNN model predictions for long-term integration up to $t = 100$ with $n_M = 1300$ and $n_R = 1$ are shown in Fig. 6 for the state variables $\mathbf{p}$ and in Fig. 7 for the state variables $\mathbf{q}$, using a set of arbitrarily chosen initial conditions and hidden parameter values. We observe very good agreement between the DNN model predictions and the corresponding reference solutions.

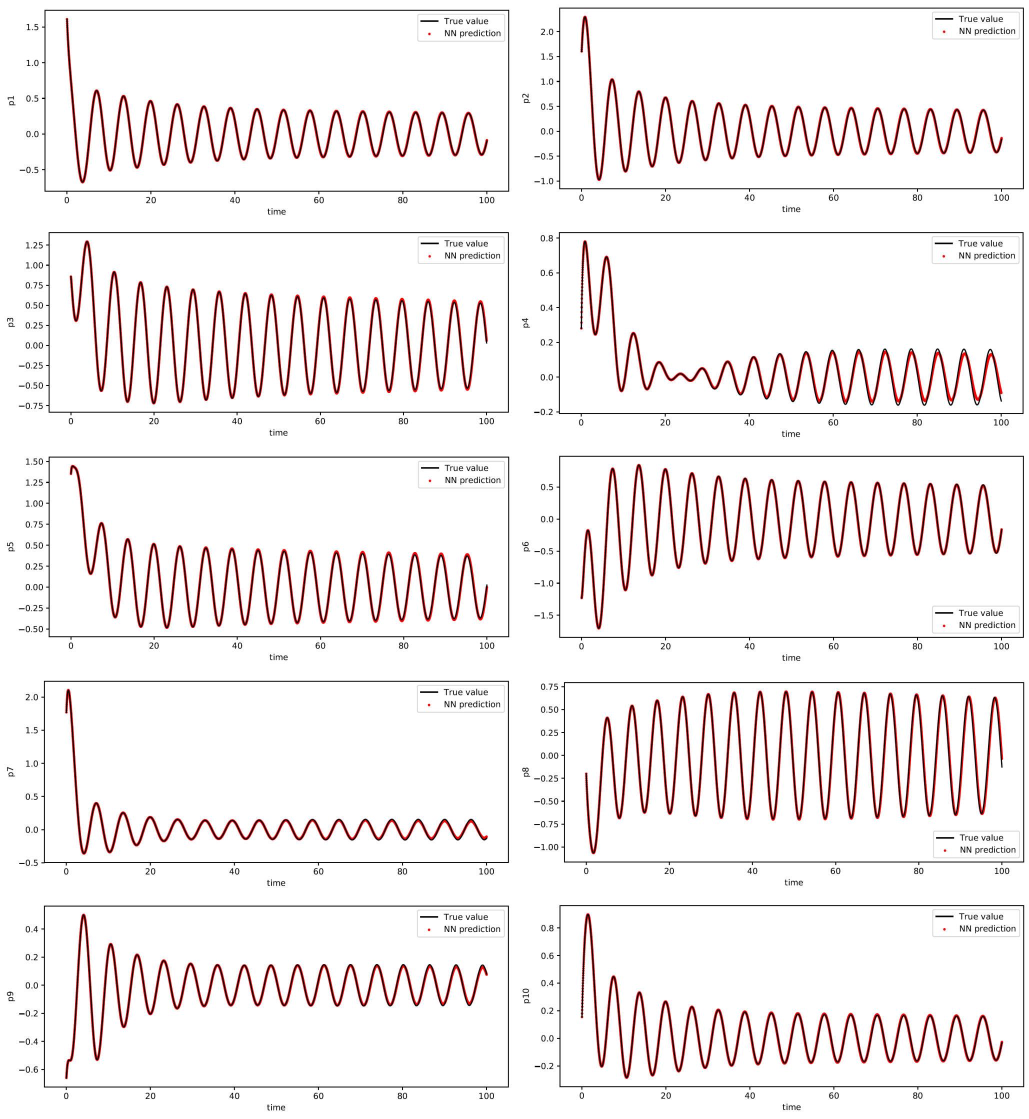


FIG. 6: Example 2. Model predictions of $\mathbf{p}$ up to $t = 100$ with $n_M = 1300$ and $n_R = 1$.

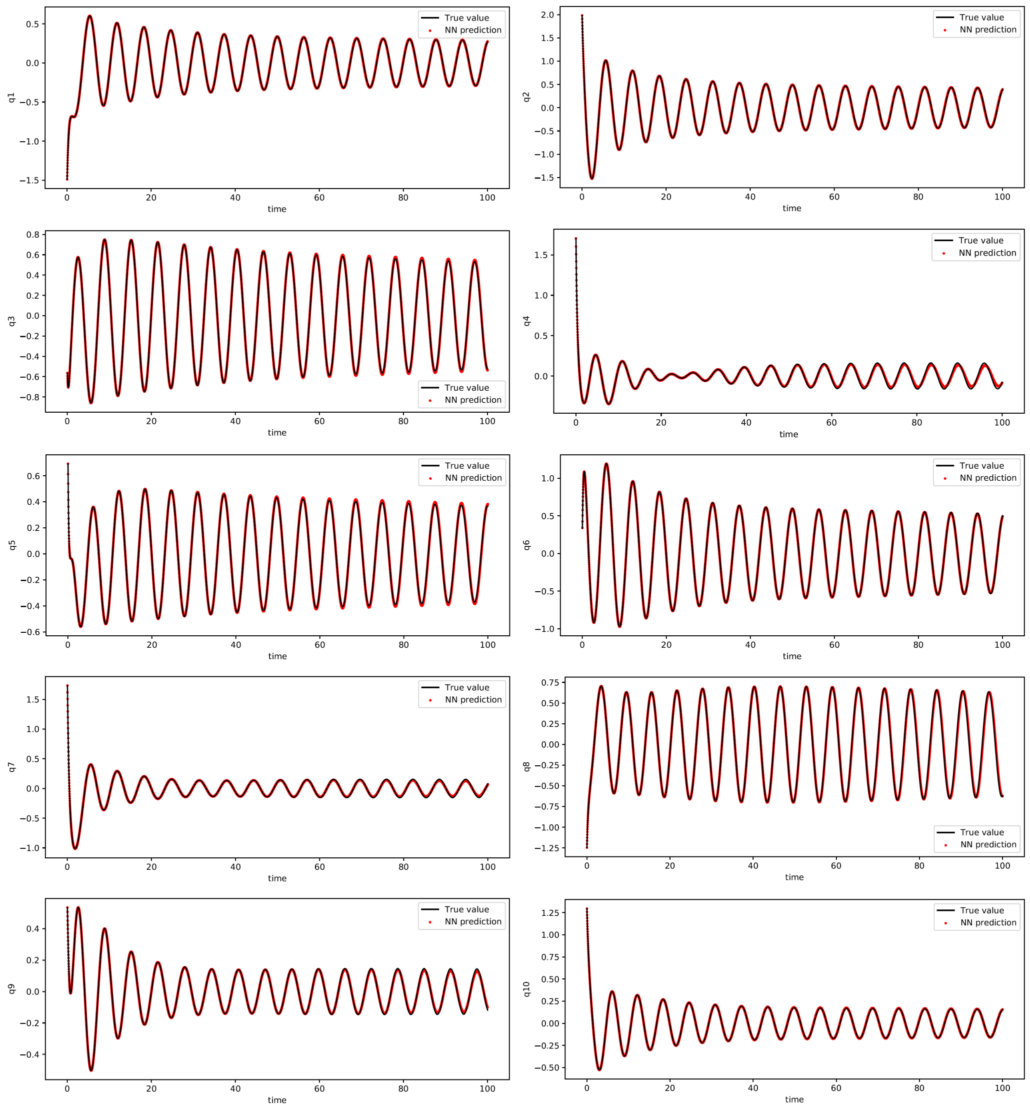


FIG. 7: Example 2. Model predictions of $\mathbf{q}$ up to $t = 100$ with $n_M = 1300$ and $n_R = 1$.

4.3 Example 3: CSTR

We now consider a smaller nonlinear system with bifurcation behavior controlled by the hidden parameter. It is a continuous stirred-tank chemical reactor (CSTR) model with a single and irreversible exothermic reaction. The (unknown) governing equations are

$$\begin{cases} \dot{x}_1 = -x_1 + \mathrm{Da}(1 - x_1)\exp\left(\dfrac{x_2}{1 + x_2/\gamma}\right), \\ \dot{x}_2 = -x_2 + B\,\mathrm{Da}(1 - x_1)\exp\left(\dfrac{x_2}{1 + x_2/\gamma}\right) - \beta(x_2 - x_{2c}), \end{cases} \tag{23}$$

where x_1 is the conversion and x_2 the temperature, Da the Damköehler number, B the heat of reaction, β the heat transfer coefficient, γ the activation energy, and x_{2c} the coolant temperature. The dimensionless Damköehler number Da plays an important role in determining the qualitative system behavior and will be assumed to be a hidden parameter. All other parameters are fixed: $B = 22.0$, $\beta = 3.0$, $\gamma = 12.0$, and $x_{2c} = 0.5$.

We restrict the range of the hidden Da number to be within $\pm$ 10% of the value 0.078. This is an intentional choice, as Da $= 0.078$ is the critical value at which the system exhibits bifurcation behavior: the system reaches steady state when Da < 0.078 and limit cycle state when Da $>$ 0.078.

To generate the training data set, we set the domain of interest for the state variables to be $(x_1, x_2) \in [0.1, 1.0] \times [0.5, 5.5]$. The time step is set as $\Delta t = 0.02$. Upon conducting numerical tests, we set the memory step to $n_M = 700$ and the recurrent step to $n_R = 1$.

We show the DNN trajectory predictions in Fig. 8, with two sets of arbitrarily chosen initial conditions and parameters where trajectories exhibit steady state and limit cycle, respectively.

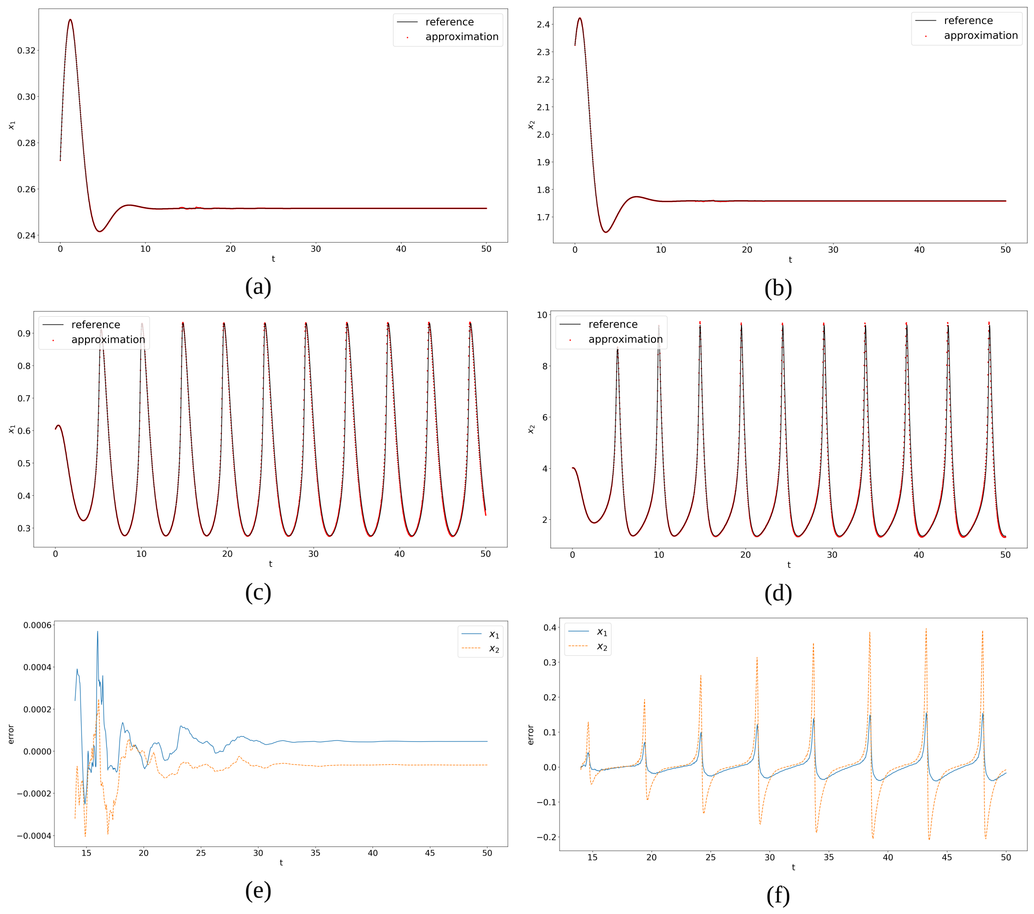


FIG. 8: Example 3. Model predictions up to $t = 50$ with $n_M = 700$ and $n_R = 1$ with two cases of arbitrarily chosen initial conditions and system parameters. (a) Case 1: x_1, (b) Case 1: x_2, (c) Case 2: x_1, (d) Case 2: x_2, (e) Case 1: relative error, (f) Case 2: relative error.

We observe that the predictions match the reference solutions very well in both cases. To determine the qualitative behavior of the solutions, we compute the amplitude of the solutions when they reach a stable state over a relatively longer time interval $t \in [50, 70]$. If the trajectory reaches a steady state, then the amplitude approaches 0; if the trajectory becomes periodic, then its amplitude approaches a constant value. Figure 9 shows the amplitudes of the predictions with respect to the value of Da, for both x_1 and x_2. We clearly observe the transition from steady state to periodic state when Da $\approx$ 0.078. The comparison between the DNN predictions and the reference true solutions again shows good agreement.

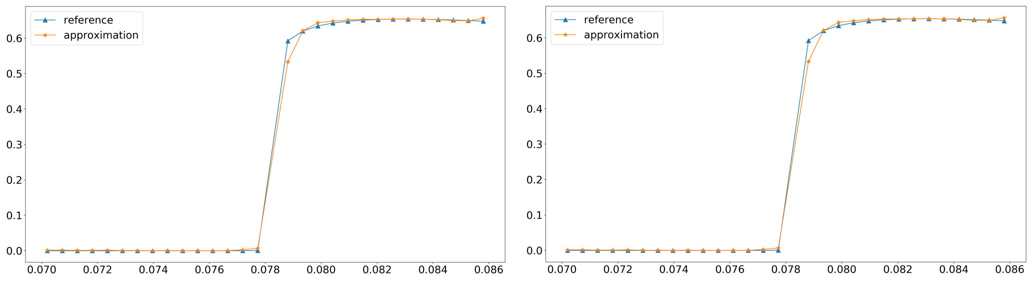
4.4 Example 4: Cell Signaling Cascade

We consider a dynamical system model for the autocrine cell-signaling loop. The three-dimensional state variable $[e_{1p}, e_{2p}, e_{3p}]^\top$ denotes the dimensionless concentrations of the active form of the enzymes. The true (and unknown) governing equations are

$$\begin{cases} \dfrac{de_{1p}}{dt} = \dfrac{I(t)}{1 + G_4 e_{3p}} \dfrac{V_{\max,1}(1 - e_{1p})}{K_{m,1} + (1 - e_{1p})} - \dfrac{V_{\max,2} e_{1p}}{K_{m,2} + e_{1p}}, \\ \dfrac{de_{2p}}{dt} = \dfrac{V_{\max,3} e_{1p}(1 - e_{2p})}{K_{m,3} + (1 - e_{2p})} - \dfrac{V_{\max,4} e_{2p}}{K_{m,4} + e_{2p}}, \\ \dfrac{de_{3p}}{dt} = \dfrac{V_{\max,5} e_{2p}(1 - e_{3p})}{K_{m,5} + (1 - e_{3p})} - \dfrac{V_{\max,6} e_{3p}}{K_{m,6} + e_{3p}}, \end{cases} \tag{24}$$

where $I = 1.0$, $G_4 = 0.2$ are fixed and the parameters $K_{m,i}$, $V_{\max,i}$, $i = 1, \ldots, 6$, are hidden parameters, for a total of 12 hidden parameters. For this study, we restrict the hidden parameters to within $\pm$ 10% of their nominal values. The nominal values for all $K_{m,i}$, $i = 1, \ldots, 6$, are fixed at 0.2, and for $V_{\max,1}$ is 0.5, for $V_{\max,2,3,4}$ are 0.15, for $V_{\max,5}$ is 0.25, and for $V_{\max,6}$ is 0.05. The domain of interest for the state variable is $[0, 1]^3$.

The training data are constructed by collecting two randomly selected sequences of consecutive data entries from 75,000 trajectories, generated by uniformly distributed random initial conditions over 300 steps with a time step $\Delta t = 0.1$. In our DNN model, the memory steps are set as $n_M = 50$ and the recurrent steps as $n_R = 12$. The trajectory predictions and the error plots are shown in Fig. 10, with a set of arbitrary initial conditions and system parameters. We observe that the DNN predictions match the reference solutions very well for up to $t = 20$.



(a) (b)

FIG. 9: Example 3. Solution amplitudes at limiting states with respect to Da number: (a) amplitude of x_1 vs. Da, (b) amplitude of x_2 vs. Da.

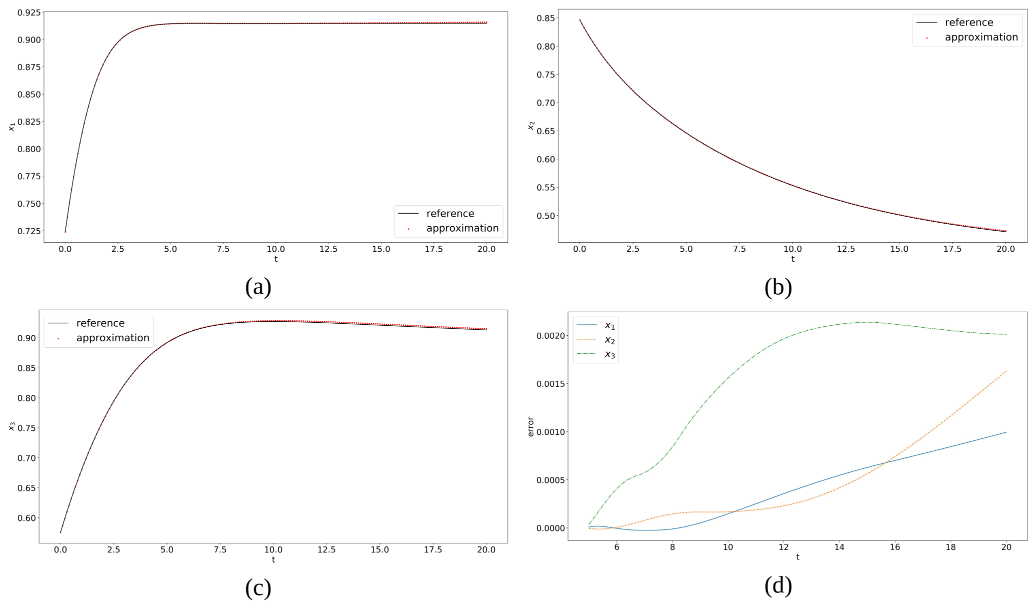


FIG. 10: Example 4. Model predictions and errors up to $t = 20$ with $n_M = 50$ and $n_R = 12$ using a set of arbitrary initial conditions and parameters. (a) x_1, (b) x_2, (c) x_3, (d) error.

5. CONCLUSION

We presented a deep learning strategy for modeling unknown dynamical systems with hidden parameters. By incorporating both memory terms in the network input layer and recurrent terms in the network loss function computation, the proposed DNN is able to learn the unknown flow map of the system, by only using trajectory data of the state variables. A distinct feature of the DNN structure is that it is able to model the system with completely hidden and unknown parameters. This can be useful for practical problems, where many system parameters cannot be measured. The proposed DNN method thus provides a highly flexible approach for learning unknown dynamical systems.

REFERENCES

Brunton, S.L., Proctor, J.L., and Kutz, J.N., Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems, *Proc. Natl. Acad. Sci., USA*, vol. **113**, no. 15, pp. 3932–3937, 2016.

Chen, Z. and Xiu, D., On Generalized Residual Network for Deep Learning of Unknown Dynamical Systems, *J. Comput. Phys.*, vol. **438**, p. 110362, 2021.

Fu, X., Chang, L.B., and Xiu, D., Learning Reduced Systems via Deep Neural Networks with Memory, *J. Mach. Learn. Model. Comput.*, vol. **1**, no. 2, pp. 97–118, 2020.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, pp. 770–778, 2016.

Long, Z., Lu, Y., and Dong, B., PDE-Net 2.0: Learning PDEs from Data with a Numeric-Symbolic Hybrid Deep Network, arXiv: 1812.04426, 2018a.

Long, Z., Lu, Y., Ma, X., and Dong, B., PDE-Net: Learning PDEs from Data, in *Proc. of the 35th Int. Conf. on Machine Learning*, Stockholmsmässan, Stockholm, Sweden, pp. 3208–3216, 2018b.

Mangan, N.M., Kutz, J.N., Brunton, S.L., and Proctor, J.L., Model Selection for Dynamical Systems via Sparse Regression and Information Criteria, *Proc. R. Soc. London, Ser. A: Math. Phys. Eng. Sci.*, vol. **473**, no. 2204, 2017.

Mori, H., Transport, Collective Motion, and Brownian Motion, *Prog. Theor. Phys.*, vol. **33**, no. 3, pp. 423–455, 1965.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Data-Driven Learning of Nonautonomous Systems, *SIAM J. Sci. Comput.*, vol. **43**, no. 3, pp. A1607–A1624, 2021a.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Deep Learning of Parameterized Equations with Applications to Uncertainty Quantification, *Int. J. Uncertainty Quant.*, vol. **11**, no. 2, pp. 63–82, 2021b.

Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, *J. Comput. Phys.*, vol. **395**, pp. 620–635, 2019.

Raissi, M., Deep Hidden Physics Models: Deep Leaming of Nonlinear Partial Differential Equations, *J. Mach. Learn. Res.*, vol. **19**, no. 25, pp. 1–24, 2018.

Raissi, M., Perdikaris, P., and Kamiadakis, G.E., Machine Leaming of Linear Differential Equations Using Gaussian Processes, *J. Comput. Phys.*, vol. **348**, pp. 683–693, 2017a.

Raissi, M., Perdikaris, P., and Kamiadakis, G.E., Physics Informed Deep Leaming (Part I): DataDriven Solutions of Nonlinear Partial Differential Equations, arXiv: 1711.10561, 2017b.

Raissi, M., Perdikaris, P., and Kamiadakis, G.E., Physics Informed Deep Leaming (Part II): DataDriven Discovery of Nonlinear Partial Differential Equations, arXiv: 1711.10566, 2017c.

Raissi, M., Perdikaris, P., and Kamiadakis, G.E., Multistep Neural Networks for Data-Driven Discovery of Nonlinear Dynamical Systems, arXiv: 1801.01236, 2018.

Rudy, S.H., Brunton, S.L., Proctor, J.L., and Kutz, J.N., Data-Driven Discovery of Partial Differential Equations, *Sci. Adv.*, vol. **3**, no. 4, p. e1602614, 2017.

Rudy, S.H., Kutz, J.N., and Brunton, S.L., Deep Learning of Dynamics and Signal-Noise Decomposition with Time-Stepping Constraints, *J. Comput. Phys.*, vol. **396**, pp. 483–506, 2019.

Schaeffer, H., Tran, G., and Ward, R., Extracting Sparse High-Dimensional Dynamics from Limited Data, *SIAM J. Appl. Math.*, vol. **78**, no. 6, pp. 3279–3295, 2018.

Sun, Y., Zhang, L., and Schaeffer, H., NEUPDE: Neural Network Based Ordinary and Partial Differential Equations for Modeling Time-Dependent Data, arXiv: 1908.03190, 2019.

Wu, K., Qin, T., and Xiu, D., Structure-Preserving Method for Reconstructing Unknown Hamiltonian Systems from Trajectory Data, arXiv: 1905.10396, 2019.

Wu, K. and Xiu, D., Numerical Aspects for Approximating Governing Equations Using Data, *J. Comput. Phys.*, vol. **384**, pp. 200–221, 2019.

Wu, K. and Xiu, D., Data-Driven Deep Leaming of Partial Differential Equations in Modal Space, *J. Comput. Phys.*, vol. **408**, p. 109307, 2020.

Zwanzig, R., Nonlinear Generalized Langevin Equations, *J. Stat. Phys.*, vol. **9**, no. 3, pp. 215–220, 1973.

APPENDIX A. DETAILS OF EXAMPLE 2

The detailed setting of Example 2 is $\mathbf{x} = (\mathbf{p}; \mathbf{q})$, where $\mathbf{p} \in \mathbb{R}^{10}$ and $\mathbf{q} \in \mathbb{R}^{10}$ satisfy

$$\begin{cases} \dot{\mathbf{p}} = \Sigma_{11}\mathbf{p} + (\mathbf{I} + \Sigma_{12})\mathbf{q}, \\ \dot{\mathbf{q}} = -(\mathbf{I} + \Sigma_{21})\mathbf{p} - \Sigma_{22}\mathbf{q}. \end{cases} \tag{A.1}$$

Here, $\mathbf{I}$ is the identity matrix of size 10×10, and $\Sigma_{ij} \in \mathbb{R}^{10\times 10}$, $i = 1, 2, j = 1, 2$ are four coefficient matrices. We set three of the coefficient matrices as fixed, with $\Sigma_{11} = \Sigma_{12} = 0$, and

$$
\begin{aligned}
&\Sigma_{22} \times 10^3 \\
&= \begin{pmatrix}
1500 & 124 & 814 & -104 & -179 & -223 & -731 & -189 & -400 & 242 \\
124 & 836 & 679 & 277 & 197 & -515 & -52.1 & -273 & 101 & 301 \\
814 & 679 & 1500 & 651 & 755 & -605 & -379 & -546 & -225 & 223 \\
-104 & 277 & 651 & 1960 & 720 & -782 & -299 & -775 & -180 & 506 \\
-179 & 197 & 755 & 720 & 2290 & -973 & 518 & -19.1 & -604 & -369 \\
-223 & -515 & -605 & -782 & -973 & 1290 & -400 & 412 & 314 & -420 \\
-731 & -52.1 & -379 & -299 & 518 & -400 & 1960 & 68.3 & 455 & -316 \\
-189 & -273 & -546 & -775 & -19.1 & 412 & 68.3 & 576 & -53.6 & -332 \\
-400 & 101 & -225 & -180 & -604 & 314 & 455 & -53.6 & 1030 & 265 \\
242 & 301 & 223 & 506 & -369 & -420 & -316 & -332 & 265 & 1090
\end{pmatrix}.
\end{aligned}
$$

The 100 entries of the matrix Σ_{21} are treated as hidden parameters.