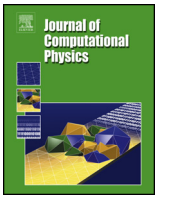
Journal of Computational Physics 493 (2023) 112502

Contents lists available at ScienceDirect

Journal of Computational Physics

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



DNN modeling of partial differential equations with incomplete data ☆

Victor Churchill [a], Yuan Chen [b], Zhongshu Xu [b], Dongbin Xiu [b,*]

[a] Department of Mathematics, Trinity College, Hartford, CT 06106, USA
[b] Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA

ARTICLE INFO

Article history:
Received 17 February 2023
Received in revised form 31 July 2023
Accepted 10 September 2023
Available online 17 September 2023

Keywords:
Data driven modeling
Deep neural networks
Reduced PDE systems
Incomplete data

ABSTRACT

We present a computational technique for modeling the evolution of partial differential equations (PDEs) with incomplete data. It is a significant extension of the recent work of data driven learning of PDEs, in the sense that we consider two forms of partial data: data are observed only on a subset of the domain, and data are observed only on a subset of the state variables. Both cases resemble more realistic data collection scenarios in real-world applications. Leveraging the recent work on modeling partially-observed dynamical systems, we present a deep neural network (DNN) structure that is suitable for PDE modeling with such kinds of incomplete data. In addition to the mathematical motivation for the DNN structure, we present an extensive set of numerical examples in both one- and two-dimensions to demonstrate the effectiveness of the proposed DNN modeling. In one example, the method can accurately predict the solution when data are only available in less than half (40%) of the domain.

1. Introduction

Data-driven learning of unknown partial differential equations (PDEs) has been a prominent area of research in recent years. One way to approach this problem is to construct a mapping from the state variables and their spatial derivatives to their time derivatives, which can explicitly reconstruct the governing equations via sparse approximation from a large dictionary set. In certain circumstances, exact equation recovery is possible. See, for example, [1] and its many extensions in recovering both ODEs ([1,9,28,29,31]) and PDEs ([25,27]). Deep neural networks (DNNs) have also been used to construct this mapping. See, for example, ODE modeling in [16,20,24,26], and PDE modeling in [13–15,22,23,21,30].

This paper focuses on another approach for learning unknown PDEs, which is to construct a mapping between two discretized system states separated by a short time. Unlike the governing equation discovery approach, this evolution discovery approach constructs an approximate flow map between system states and does not directly formulate the underlying equations. Instead, if an accurate approximation for the true flow map is constructed, it allows for the definition of an accurate predictive model for the unknown system that marches new initial conditions through time. This approach is particularly effective when DNNs, more specifically, residual networks (ResNet [7]) are used to approximate the flow map, and it does not require construction of an appropriate dictionary as in governing equation discovery. Introduced to model autonomous

☆ Funding: This work was partially supported by AFOSR FA9550-22-1-0011.
* Corresponding author.
E-mail addresses: victor.churchill@trincoll.edu (V. Churchill), chen.11050@osu.edu (Y. Chen), xu.4202@osu.edu (Z. Xu), xiu.16@osu.edu (D. Xiu).

https://doi.org/10.1016/j.jcp.2023.112502
0021-9991/

ODE systems in [20], the flow map learning (FML) method has been extended to modeling of non-autonomous ODE systems [19], parametric dynamical systems [18], partially-observed dynamical systems [6], chaotic systems [4], and systems with coarsely observed temporal data [5]. Modeling PDEs represents a significant deviation from modeling ODEs. To this end, several methods have been explored, cf., [10–12,15]. The FML approach was also extended and adapted to PDE modeling, with observation data in modal space [33], as well as in nodal space [2].

In this paper, we consider data-driven modeling of PDEs with incomplete data, which is a case rarely considered in the previous related literature. It is a step closer to real-world problems, where it is often difficult, if not impossible, to collect data of all the state variables in all areas of the domain for an unknown system. In fact, sometimes it may be difficult even to *identify* all state variables in an unknown system. In other words, we consider the case of modeling an unknown PDE system using measurable data that represent partial information of the system. More specifically, we consider two cases: (1) when only a subset of the state variables are observed; and (2) when data of the state variables are available only in a sub-domain of the entire domain. Most, if not all, of the existing methods are not applicable for these cases. The method presented in this paper is based on generalization of two recent works: modeling of partially-observed dynamical systems [6] and DNN modeling of PDEs in nodal space [2]. Motivated by the celebrated Mori-Zwanzig formulation, the work of [6] proposed the use of DNNs with memory terms to model dynamical systems when only a subset of the state variables are observed. (It should be noted that the same modeling principle was first proposed in [32], in the context of closure for reduced order modeling and by using LSTM structure.) In this paper, we adopt the simple DNN structure from [6], which explicitly incorporates memory terms from the measurement data as network inputs, in order to cope with the situation of incomplete data. To effectively model PDE systems, we employ the modeling principle from [2], where a specific DNN structure with disassembly layers and an assembly layer was developed to model the unknown spatial operators in the unknown PDEs. As a result, the proposed new DNN structure is capable of modeling unknown PDE systems using incomplete data. Through an extensive set of numerical examples, we demonstrate that our method is indeed highly effective and accurate.

The proposed method is particular useful for modeling time dependent PDEs system, whose the governing PDE equations are not known. When one has snapshot data of the solution variables, the method can be readily applied to create an accurate predictive model of the system, without seeking to identify the PDEs. More notably, our method can be used when the solution data are available in only part of the domain, or when only a subset of the solution variables can be observed.

2. Preliminaries

In this section, we briefly review the existing DNN-based framework for learning the evolution of unknown ODEs, partially-observed ODEs, and PDEs. Throughout this paper our discussion will be over discrete time instances with a constant time step Δt,

$$t_0 < t_1 < \cdots, \qquad t_{n+1} - t_n = \Delta t, \quad \forall n. \tag{2.1}$$

We will also use subscript to denote the time variable of a function, e.g., $\mathbf{x}_n = \mathbf{x}(t_n)$.

2.1. ResNet modeling of ODEs

For an unknown autonomous system,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d, \tag{2.2}$$

where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is unknown. The flow map of the system is a function that describes the evolution of the solution. The flow map of (2.2) depends only on the time difference but not the actual time, i.e., $\mathbf{x}_n = \mathbf{\Phi}_{t_n - t_s}(\mathbf{x}_s)$. Thus, the solution over one time step satisfies

$$\mathbf{x}_{n+1} = \mathbf{\Phi}_{\Delta t}(\mathbf{x}_n) = \mathbf{x}_n + \mathbf{\Psi}_{\Delta t}(\mathbf{x}_n), \tag{2.3}$$

where $\mathbf{\Psi}_{\Delta t} = \mathbf{\Phi}_{\Delta t} - \mathbf{I}$, with $\mathbf{I}$ as the identity operator.

When data for the state variables $\mathbf{x}$ over the time stencil (2.1) are available, they can be grouped into sequences separated by one time step

$$\{\mathbf{x}^{(l)}(0), \ \mathbf{x}^{(l)}(\Delta t), \ldots, \mathbf{x}^{(l)}(K\Delta t)\}, \quad l = 1, \ldots, L, \tag{2.4}$$

where L is the total number of such data sequences and $K+1$ is the length of each sequence.[1] These data sequences form the training data set. Inspired by basic one-step numerical schemes for solving ODEs, one can then model the unknown evolution operator using a residual network (ResNet [7]) in the form of

[1] The data sequence length K is assumed to be constant for notational convenience. If in fact data are of different lengths, chunks of the minimum sequence length can be taken from longer trajectories.

$$\mathbf{y}^{out} = [\mathbf{I} + \mathbf{N}]\,(\mathbf{y}^{in}), \tag{2.5}$$

where $\mathbf{N} : \mathbb{R}^d \to \mathbb{R}^d$ stands for the mapping operator of a standard feedforward fully connected neural network. The network is then trained by using the training data set and minimizing the recurrent mean squared loss function

$$\frac{1}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\left\|\mathbf{x}^{(l)}(k\Delta t) - [\mathbf{I}+\mathbf{N}]^k(\mathbf{x}^{(l)}(0))\right\|^2, \tag{2.6}$$

where $[\mathbf{I}+\mathbf{N}]^k$ indicates composition of the network function k times. This recurrent loss permits the incorporation of longer data lengths and is used to increase the stability of the network approximation over long term prediction. Once the network is trained to satisfactory accuracy, the trained network thus accomplishes

$$\mathbf{x}^{(l)}(k\Delta t) \approx [\mathbf{I}+\mathbf{N}]^k(\mathbf{x}^{(l)}(0)), \qquad \forall l = 1,\dots,L, \quad k = 1,\dots,K,$$

and it can be used as a predictive model

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{N}(\mathbf{x}_n), \qquad n = 0,1,2,\dots, \tag{2.7}$$

for any initial condition $\mathbf{x}(t_0)$. This framework was proposed in [20], with extensions to parametric systems and time-dependent (non-autonomous) systems ([18,19]).

2.2. Memory-based modeling of ODEs

A particularly relevant extension of the flow map modeling in [20] to systems with missing variables was presented in [6]. Let $\mathbf{x} = (\mathbf{z}; \mathbf{w})$, where $\mathbf{z} \in \mathbb{R}^p$ and $\mathbf{w} \in \mathbb{R}^{d-p}$. Let $\mathbf{z}$ be the observables and $\mathbf{w}$ be the missing variables. That is, no information about $\mathbf{w}$ is available. When data are available for $\mathbf{z}$, it is possible to derive a system of equations for $\mathbf{z}$ only. The celebrated Mori-Zwanzig formulation ([17,34]) asserts that the system for $\mathbf{z}$ requires a memory integral. By making a mild assumption that the memory is of finite length (which is problem dependent), memory-based DNN structures were investigated [32,6]. While [32] utilized LSTM (long short-term memory) networks, [6] proposed a fairly simple DNN structure, in direct correspondence to the Mori-Zwanzig formulation, that takes the following mathematical form,

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{N}(\mathbf{z}_n, \mathbf{z}_{n-1}, \dots, \mathbf{z}_{n-n_M}), \qquad n \geq n_M, \tag{2.8}$$

where $n_M \geq 0$ is the number of memory terms in the model. In this case, the DNN operator is $\mathbf{N} : \mathbb{R}^{p\times(n_M+1)} \to \mathbb{R}^p$. Note that in the case where memory is required, the dataset (2.4) is amended to

$$\{\mathbf{z}^{(l)}_{n-n_M}, \dots, \mathbf{z}^{(l)}_{n-1}, \mathbf{z}^{(l)}_{n}, \mathbf{z}^{(l)}_{n+1}, \dots, \mathbf{z}^{(l)}_{n+K}\}, \quad l = 1,\dots,L, \tag{2.9}$$

such that the data sequences are now length $n_M + K + 1$ so that the appropriate memory length and recurrent loss can be incorporated. The special case of $n_M = 0$ corresponds to the standard ResNet model (2.7), which is for modeling systems without missing variables (thus no need for memory).

2.3. Modeling of PDE

The flow map learning approach has also been applied in modeling PDEs. When the solutions of the PDE (i.e. the training data) can be expressed using a fixed basis, the learning can be conducted in modal space. In this case, the ResNet approach can be adopted. See [33] for details. When data of the PDE solutions are available as nodal values over a set of grid points in physical space, its DNN learning is more involved. In this case, a DNN structure was developed in [2], which can accommodate the situation when the data are on unstructured grids as well as PDE systems. It is based on a simple numerical scheme for solving the PDE, and consists of a set of specialized layers including disassembly layers and an assembly layer, which resemble the differential operators involved in the unknown PDE. The proposed DNN model defines the following mapping,

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mathbf{A}(\mathbf{F}_1(\mathbf{w}_n), \dots, \mathbf{F}_J(\mathbf{w}_n)), \tag{2.10}$$

where $\mathbf{F}_1, \dots, \mathbf{F}_J$ are the operators for the disassembly channels and $\mathbf{A}$ the operator for the assembly layer. The DNN modeling approach was shown to be highly flexible and accurate to learn a variety of PDEs [2]. See [2] for more details on this structure and its mathematical properties. Since this configuration can also be viewed as a residual network application, i.e. $\mathbf{w}_{n+1} = [\mathbf{I}+\mathbf{N}](\mathbf{w}_n)$ where $\mathbf{N} = \mathbf{A} \circ (\mathbf{F}_1, \dots, \mathbf{F}_J)$, we can directly apply the recurrent loss scheme described above. Since this scheme will be expanded in the next section, we leave additional details until then.

3. Learning partially-observed PDE systems

In the main task of this paper, we describe a new method for approximating the evolution operator of a partially-observed system of PDEs using DNNs. We first set up the problem being considered with appropriate notation, then move on to motivate and discuss the new DNN structure.

3.1. Problem setup and data

Consider an autonomous time-dependent system of PDEs,

$$\begin{cases} \mathbf{u}_t = \mathcal{L}(\mathbf{u}), & (x,t) \in \Omega \times \mathbb{R}^+, \\ \mathcal{B}(\mathbf{u}) = 0, & (x,t) \in \partial\Omega \times \mathbb{R}^+, \\ \mathbf{u}(x,0) = \mathbf{u}_0(x), & x \in \bar{\Omega}, \end{cases} \tag{3.1}$$

where $\mathbf{u}$, the vector of state variables, has n elements, $\Omega \subset \mathbb{R}^d$, $d = 1, 2, 3$, is the physical domain, and $\mathcal{L}$ and $\mathcal{B}$ stand for the PDE operator and boundary condition operator, respectively, of the entire system. We assume that the PDE is unknown.

Let $\mathbf{u} = (\mathbf{v}; \mathbf{w})$, where $\mathbf{v}$ contains the m state variables with available data and $\mathbf{w}$ contains the $n - m$ unobserved subset of state variables. Our goal is to model the effective governing equations for the reduced system of observed variables $\mathbf{v}$.

We assume that state variable solution data for $\mathbf{v}$ are available over a set of nodal points, or grids,

$$X_N = \{x_1, \dots, x_N\} \subset \Omega. \tag{3.2}$$

We note that we treat X_N as quite a general grid that may in fact be a discretization of some proper subset of Ω such that $\text{Conv}(X_N) \subsetneq \Omega$ where $\text{Conv}(X_N)$ is the convex hull of X_N. We emphasize that the difference can be nontrivial, i.e. $|\Omega \setminus \text{Conv}(X_N)| = \mathcal{O}(1)$ where $|\cdot|$ represents the Lebesgue measure.

Using vector notation, we concatenate

$$\mathbf{V}(t) = (V_1(t), \dots, V_m(t))^T \in \mathbb{R}^{m \cdot N}, \tag{3.3}$$

where each component of the subset of state variables is given by

$$V_j(t) = (v_j(x_1, t), \dots, v_j(x_N, t)) \in \mathbb{R}^N, \quad j = 1, \dots, m. \tag{3.4}$$

As in [2], we note that the DNN structure that follows will allow us to simply concatenate the state variable solutions on grids (or arbitrarily permute these grids if data is collected in a different manner). These state variables are observed at discrete time instances (2.1) such that the set

$$\{\mathbf{V}^{(l)}_{n-n_M}, \dots, \mathbf{V}^{(l)}_n, \mathbf{V}^{(l)}_{n+1}, \dots, \mathbf{V}^{(l)}_{n+K}\}, \quad l = 1, \dots, L, \tag{3.5}$$

represents all available data similar to (2.9) where $n_M \geq 0$ is the memory length, K is the recurrent loss parameter, and L is the total number of trajectories available. Note that we continue to use the compact notation, $\mathbf{V}_n = \mathbf{V}(t_n)$. Accounting for each of the L training samples with length of each vector $m \cdot N$ observed over the memory and recurrent length $n_M + K + 1$, the dataset is of size $m \cdot N \times (n_M + K + 1) \times L$.

3.2. Mathematical motivation

In [2], a simple one-step forward Euler method for solving PDEs motivated a DNN structure that nonlinearly combines several NN channel outputs to resemble the right-hand-side PDE operator. We now consider partially-observed PDEs.

Without loss of generality, to motivate the problem we consider a p-th order autonomous PDE in the following general form of

$$\mathbf{u}_t = \mathcal{L}(\mathbf{u}, \partial^{(1)}\mathbf{u}, \dots, \partial^{(p)}\mathbf{u}), \quad p \geq 1, \tag{3.6}$$

where $\mathbf{u} = (\mathbf{v}; \mathbf{w})^T$, and for $1 \leq s \leq p$,

$$\partial^{(s)} = \{\partial^{\alpha_1}_{x_1} \cdots \partial^{\alpha_d}_{x_d} : |\alpha| = s\}, \tag{3.7}$$

with $|\alpha| = \alpha_1 + \cdots + \alpha_d$. Note that for $d > 1$, each $\partial^{(|\alpha|)}$ represents multiple partial derivative operators. Without considering the boundary conditions, we assume that the unknown PDE (3.1) is well-posed and has a finite number $N_D \geq p$ of partial derivative terms.

In [2], the explicit construction of the following numerical scheme that relies upon approximating partial derivatives with differentiation matrices was used to inspire a DNN structure for learning the evolution of fully-observed PDEs and PDE

systems. For sufficiently smooth functions $f(x)$, for any α with $1 \le |\alpha| \le p$, there exists a differentiation matrix $\mathbf{D}^\alpha : \mathbb{R}^N \to \mathbb{R}^N$ such that

$$\partial_{x_1}^{\alpha_1} \cdots \partial_{x_d}^{\alpha_d} f(x)|_{X_N} \approx \mathbf{D}^\alpha \mathbf{f}, \tag{3.8}$$

where $\mathbf{f} = (f(x_1), \ldots, f(x_N))^T$. Then a numerical approximation scheme modeled after (3.6) can be constructed of the form

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \cdot \mathcal{L}\left(\mathbf{x}_n, \mathbf{D}^{(1)}\mathbf{x}_n, \ldots, \mathbf{D}^{(p)}\mathbf{x}_n\right), \tag{3.9}$$

where $\mathbf{D}^{(s)} = \{\mathbf{D}^\alpha : |\alpha| = s\}$, $1 \le s \le p$. Mild assumptions can be made on the PDE to show that this scheme can return low global truncation error (see [2] for details). This is a special case of the more general scheme

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \cdot \mathcal{M}\left[F_1(\mathbf{x}_n), \ldots, F_J(\mathbf{x}_n)\right], \tag{3.10}$$

where $\{F_j : \mathbb{R}^N \to \mathbb{R}^N, j = 1, \ldots, J\}$, $J \ge 1$, is a set of nonlinear functions, and $\mathcal{M}$ is a nonlinear function that operates component-wise (i.e. independently on each grid element). Specifically, $\mathcal{L}$ is $\mathcal{M}$, F_1 is the identity and $F_2, \ldots, F_J$ are the differentiation matrices, with $J = N_D + 1$. In [2], (3.10) served as a motivation for designing a residual network structure with feedforward neural network channels (resembling $F_1, \ldots, F_J$) whose outputs are nonlinearly combined component-wise (resembling $\mathcal{M}$).

Here, we must consider the case of approximating the evolution of the PDE for only $\mathbf{v}$, the observed variable of (3.6). The following design of a numerical scheme similar to (3.10) motivates the new network structure. In this case, a numerical scheme for only the observed variables would still require differentiation matrices not only for $\mathbf{v}$ but also differentiation matrices operating on the unobserved $\mathbf{w}$. For example, applying the scheme (3.10) to $\mathbf{u}$ (split into $\mathbf{v}$ and $\mathbf{w}$) yields

$$\begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix}_{n+1} = \begin{bmatrix}\mathbf{v}\\ \mathbf{w}\end{bmatrix}_n + \Delta t \cdot \begin{bmatrix}\mathcal{L}|_{\mathbf{v}}(\mathbf{v}_n, \mathbf{D}_{\mathbf{v}}^{(1)}\mathbf{v}_n, \ldots, \mathbf{D}_{\mathbf{v}}^{(p)}\mathbf{v}_n, \mathbf{w}_n, \mathbf{D}_{\mathbf{w}}^{(1)}\mathbf{w}_n, \ldots, \mathbf{D}_{\mathbf{w}}^{(p)}\mathbf{w}_n)\\ \mathcal{L}|_{\mathbf{w}}(\mathbf{v}_n, \mathbf{D}_{\mathbf{v}}^{(1)}\mathbf{v}_n, \ldots, \mathbf{D}_{\mathbf{v}}^{(p)}\mathbf{v}_n, \mathbf{w}_n, \mathbf{D}_{\mathbf{w}}^{(1)}\mathbf{w}_n, \ldots, \mathbf{D}_{\mathbf{w}}^{(p)}\mathbf{w}_n)\end{bmatrix}, \tag{3.11}$$

where $\mathbf{D}_{\mathbf{v}}$ and $\mathbf{D}_{\mathbf{w}}$ are the parts of the differentiation matrix $\mathbf{D}$ acting on $\mathbf{v}$ and $\mathbf{w}$, respectively. Similarly, the PDE operator $\mathcal{L}$ can be restricted to the equations for $\mathbf{v}$ or $\mathbf{w}$, denoted $\mathcal{L}|_{\mathbf{v}}$ and $\mathcal{L}|_{\mathbf{w}}$. Furthermore, we are only interested in a numerical scheme for $\mathbf{v}$, and we find that

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \cdot \mathcal{L}|_{\mathbf{v}}(\mathbf{v}_n, \mathbf{D}_{\mathbf{v}}^{(1)}\mathbf{v}_n, \ldots, \mathbf{D}_{\mathbf{v}}^{(p)}\mathbf{v}_n, \mathbf{w}_n, \mathbf{D}_{\mathbf{w}}^{(1)}\mathbf{w}_n, \ldots, \mathbf{D}_{\mathbf{w}}^{(p)}\mathbf{w}_n), \tag{3.12}$$

may be an appropriate scheme. However, a difficulty arises because the right hand side still has dependence on $\mathbf{w}$ and its derivatives, yet no data for $\mathbf{w}$ is available.

Relying on inspiration from the Mori-Zwanzig formalism for systems of ordinary differential equations [17,34] and previous work in [6], we posit that the reduced system (for $\mathbf{v}$ only) relies not only on the current time $\mathbf{v}_n$ but also on a finite time history. Hence, we propose to model the flow map via the scheme

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \cdot \mathcal{L}|_{\mathbf{v}}\left(\mathbf{F}^{(1)}(\mathbf{v}_n, \ldots, \mathbf{v}_{n-n_M}), \ldots, \mathbf{F}^{(q)}(\mathbf{v}_n, \ldots, \mathbf{v}_{n-n_M})\right), \tag{3.13}$$

where the functions $\mathbf{F}^{(j)}$, $j = 1, \ldots, q$, now input a time history of $\mathbf{v}$. Differing from the scheme above for fully-observed PDEs, we now have a nonlinear combination (via $\mathcal{L}|_{\mathbf{v}}$) of functions of a finite time history of $\mathbf{v}$.

The above scheme is a specific example of the more general nonlinear scheme

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathcal{M}\left[\mathcal{N}_1(\mathbf{v}_n, \ldots, \mathbf{v}_{n-n_M}), \ldots, \mathcal{N}_J(\mathbf{v}_n, \ldots, \mathbf{v}_{n-n_M})\right], \tag{3.14}$$

where $\mathcal{N}_1, \ldots, \mathcal{N}_J$ are nonlinear functions, and $\mathcal{M}$ is another nonlinear function which combines their outputs component-wise, which will serve as the main motivation for the DNN structure that follows.

3.3. DNN structure

Given the data (3.5), we present our DNN modeling of partially-observed PDE systems. We now discuss the structure of the proposed network. The basic structure is illustrated in Fig. 3.1, and consists of the following components:

- Input layer: This layer has dimensionality $m \cdot N \cdot (n_M + 1)$, accounted for by the number of components of $\mathbf{v}$ discretized over X_N and its $n_M + 1$ time history.
- Disassembly block: This block has $J \ge 1$ fully-connected feedforward neural networks operating "in parallel", where each NN channel has n_d hidden layers each with n_c neurons. It receives the $m \cdot N \cdot (n_M + 1)$ input and has output layers that create a matrix structure of dimension $m \cdot N \times J$.

- Assembly layer: This is a fully-connected feedforward neural network with n_a hidden layers each with J neurons. It operates componentwise (thickness-wise) on the outputs of the disassembly block, taking in a J-dimensional input and outputting a scalar. All $m \cdot N$ components of the disassembly output are fed through this same assembly structure, resulting in a final output size of $m \cdot N$.[2]
- Output layer: This layer has dimensionality $m \cdot N$, the dimensionality of $\mathbf{v}$ discretized over X_N, and incorporates the re-introduction of the input layer before the final output in the manner of a residual network (ResNet).

Therefore, per the motivating scheme (3.14) and the above specification of network components, we propose and define the entire network function

$$\mathcal{N}(\cdot;\Theta) : \mathbb{R}^{m\cdot N\cdot(n_M+1)} \to \mathbb{R}^{m\cdot N}, \tag{3.15}$$

consisting of hidden layers with weights and biases as well as nonlinear activation (collected in the network parameters Θ), as

$$\mathcal{N} = \left[\mathbf{I}_N + \mathcal{A} \circ (\mathcal{F}_1, \ldots, \mathcal{F}_J)\right], \tag{3.16}$$

where

$$\mathbf{I}_N : \mathbb{R}^{m\cdot N\cdot(n_M+1)} \to \mathbb{R}^{m\cdot N}, \tag{3.17}$$

is the appropriate identity operator returning only the most recent time history,

$$\mathcal{A} : \mathbb{R}^J \to \mathbb{R}, \tag{3.18}$$

is the assembly layer, a feedforward neural network function operating component-wise to nonlinearly combine the outputs of

$$\mathcal{F}_1, \ldots, \mathcal{F}_J : \mathbb{R}^{m\cdot N\cdot(n_M+1)} (\to \mathbb{R}^{n_c}) \to \mathbb{R}^{m\cdot N}, \quad J \geq 1 \tag{3.19}$$

which are the disassembly block channels, feedforward neural network functions operating independently that each take the input time history and map to an $m \cdot N$-dimensional output. The parentheses in (3.19) above serve as a reminder that the $m \cdot N \cdot (n_M + 1)$-dimensional input is first mapped to an n_c-dimensional vector and n_c is not necessarily equal to $m \cdot N$. It is important to note that the layers within the disassembly block channels $\mathcal{N}_i$ can accommodate a different number of neurons. This is codified in Fig. 3.1 as n_c. The case of choosing $n_c < m \cdot N$ is important, as (among other things) it can keep network size down as increasing memory length causes the input size to become very long in two-dimensional problems.

All together, the network operation on inputs from the dataset (3.5) looks like

$$\mathbf{V}_{n+1} = \mathbf{V}_n + \mathcal{A}\left[\mathcal{F}_1(\mathbf{V}_n, \ldots, \mathbf{V}_{n-n_M}), \ldots, \mathcal{F}_J(\mathbf{V}_n, \ldots, \mathbf{V}_{n-n_M})\right], \tag{3.20}$$

where we note that this is a generalization of the form of (3.14).

3.4. Model training and prediction

Our partially-observed PDE system learning is constructed by training the DNN model (3.20) with the dataset (3.5). We minimize the recurrent mean squared loss

$$\min_{\Theta} \frac{1}{L}\frac{1}{K}\sum_{l=1}^{L}\sum_{k=1}^{K} \|\mathcal{N}^k(\mathbf{V}_n^{(l)}, \ldots, \mathbf{V}_{n-n_M}^{(l)};\Theta) - \mathbf{V}_{n+k}^{(l)}\|_2^2, \tag{3.21}$$

where $\mathcal{N}^k$ denotes function composition k times.

Once trained, we obtain a predictive model over the grid set X_N for the underlying unknown PDE, such that given a new initial condition $(\mathbf{X}_{n_M}, \ldots, \mathbf{X}_0)^T$ over the grid X_N, we have

$$\begin{cases} (\mathbf{V}_{n_M}, \ldots, \mathbf{V}_0) = (\mathbf{X}_{n_M}, \ldots, \mathbf{X}_0), \\ \mathbf{V}_{k+n_M+1} = \mathcal{N}(\mathbf{V}_{k+n_M}, \ldots, \mathbf{V}_k), \quad k = 0, 1, 2, \ldots. \end{cases} \tag{3.22}$$

It should be noted that although the trained DNN model resembles a forward Euler numerical scheme, it is not restricted to the accuracy of a forward Euler scheme. Instead, it is a fully nonlinear model that does not generate traditional temporal error as a function of the time step. See [20] for details.

[2] We note that while it may seem logical to include m assembly layers each corresponding to $\mathcal{L}|_{v_j}$ for $j = 1, \ldots, m$, we stress that the disassembly outputs, while modeled to resemble the partial derivatives in the unknown equation, do not actually approximate these derivatives. We note also that in [2], a single assembly layer sufficed to account for PDE systems with multiple variables.

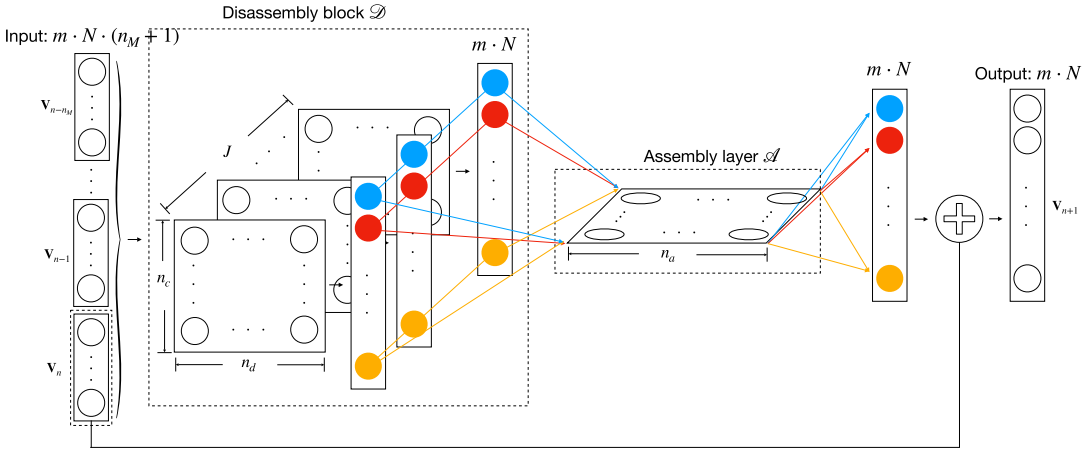


Fig. 3.1. The proposed DNN structure.

4. Computational studies

In this section, we present several numerical examples to demonstrate the proposed approach for learning partially-observed PDEs.

In each of the following numerical examples, the true governing PDEs are in fact known. However, this is only used to generate training data either by solving exactly or with a high-fidelity numerical solver, and to generate reference data for validation of the trained models. Spectral collocation method is used when solving the true governing PDEs.) We note again that the knowledge of the true PDEs does not facilitate the DNN model construction. Full specification of the data generation as well as network structure parameters are provided in each example.

We first present the case of missing variables, that is, when data are available only for a subset of the state variables. The examples include a system of linear wave equations, as well as FitzHugh-Nagumo system with diffusion in both one and two spatial dimensions. In both cases only one state variable is observed, and the other one remains unknown.

We then present the missing domain case, that is, when state variable data are available only in a sub-domain. Our examples include a fourth-order PDE, advection-diffusion equation, and viscous and inviscid Burger's equation. The parts of the domain where data are missing range from 10% to as large as 60%.

In each example, we have chosen a time step Δt such that it is reasonably large and yet delivers satisfactory temporal resolution. The numerical results exhibit strong numerical stability. Unfortunately, stability analysis remains an open issue, largely due to the lack of mathematical analysis of DNN properties. Also, how to adapt to real-world problems when the time step is subject to experimental restrictions is another an open issue.

4.1. Missing variable: wave equations

We first consider the following wave system

$$\begin{bmatrix} v \\ w \end{bmatrix}_t = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}_x, \quad (x,t) \in [0, 2\pi] \times \mathbb{R}^+, \tag{4.1}$$

with 2π-periodic boundary conditions. We consider the case when only data on the solution component v are available and aim at constructing an accurate DNN predictive model for the evolution of v.

Data generation is based on solving the full system analytically from $10,000$ randomized initial conditions given by

$$\begin{bmatrix} v \\ w \end{bmatrix}(x,0) = \begin{bmatrix} a_0 + \sum_{n=1}^{N_c} (a_n \cos(nx) + b_n \sin(nx)) \\ c_0 + \sum_{n=1}^{N_c} (c_n \cos(nx) + d_n \sin(nx)) \end{bmatrix}, \tag{4.2}$$

with $a_0, c_0 \sim U[-\frac{1}{2}, \frac{1}{2}]$ and $a_n, b_n, c_n, d_n \sim U[-\frac{1}{n}, \frac{1}{n}]$, $N_c = 7$. Then, *only the solutions for v are recorded into our training data*, which are observed on a uniform grid with $N = 50$ points at a time step $\Delta t = 0.01$ from $t = 0$ to $t = 0.3$ (30 time steps). This length is chosen as the memory length n_M plus the recurrent loss length ($K = 10$ here).

We compare memory lengths of $n_M = 0, 10, 20$, resulting in DNN structures with inputs of length $m \cdot N \cdot (n_M + 1) = 50, 550, 1050$. The $J = 5$ disassembly block channels each have 1 hidden layer with $n_c = 1,000$ neurons for all n_M. The assembly layer has $J = 5$ neurons. Hyperbolic tangent is used as the activation function for each layer. 10 DNN models are trained for $10,000$ epochs each with a constant learning rate of 10^{-3}. Testing data is created in the same way as training data, except now we use the new initial condition

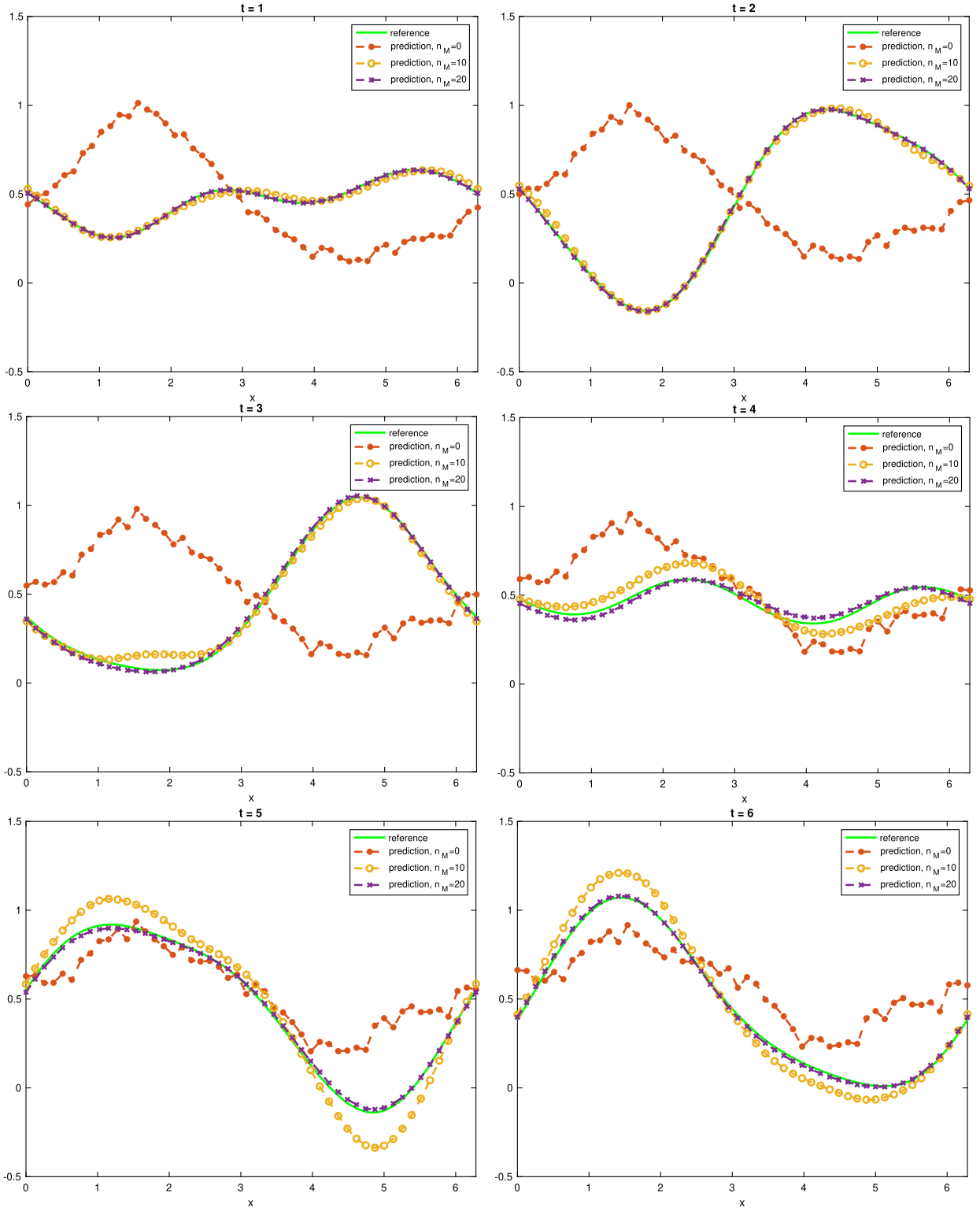


Fig. 4.1. Wave System – example test trajectory for v.

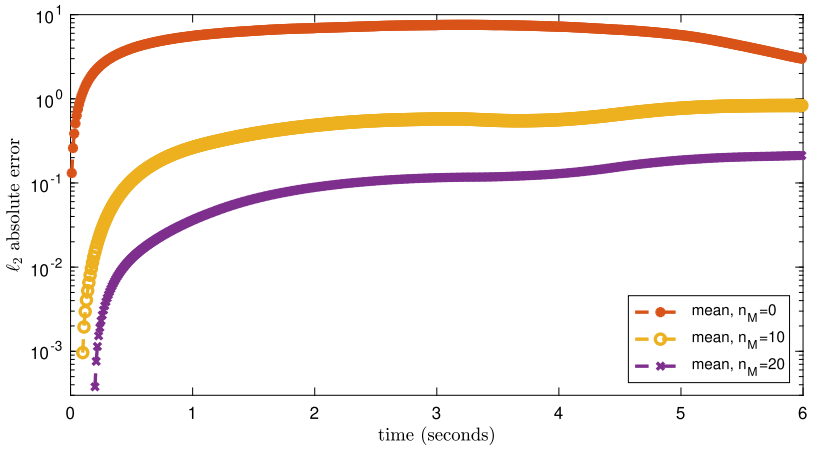


Fig. 4.2. Wave System – mean ℓ_2 absolute error of 100 test trajectories over time.

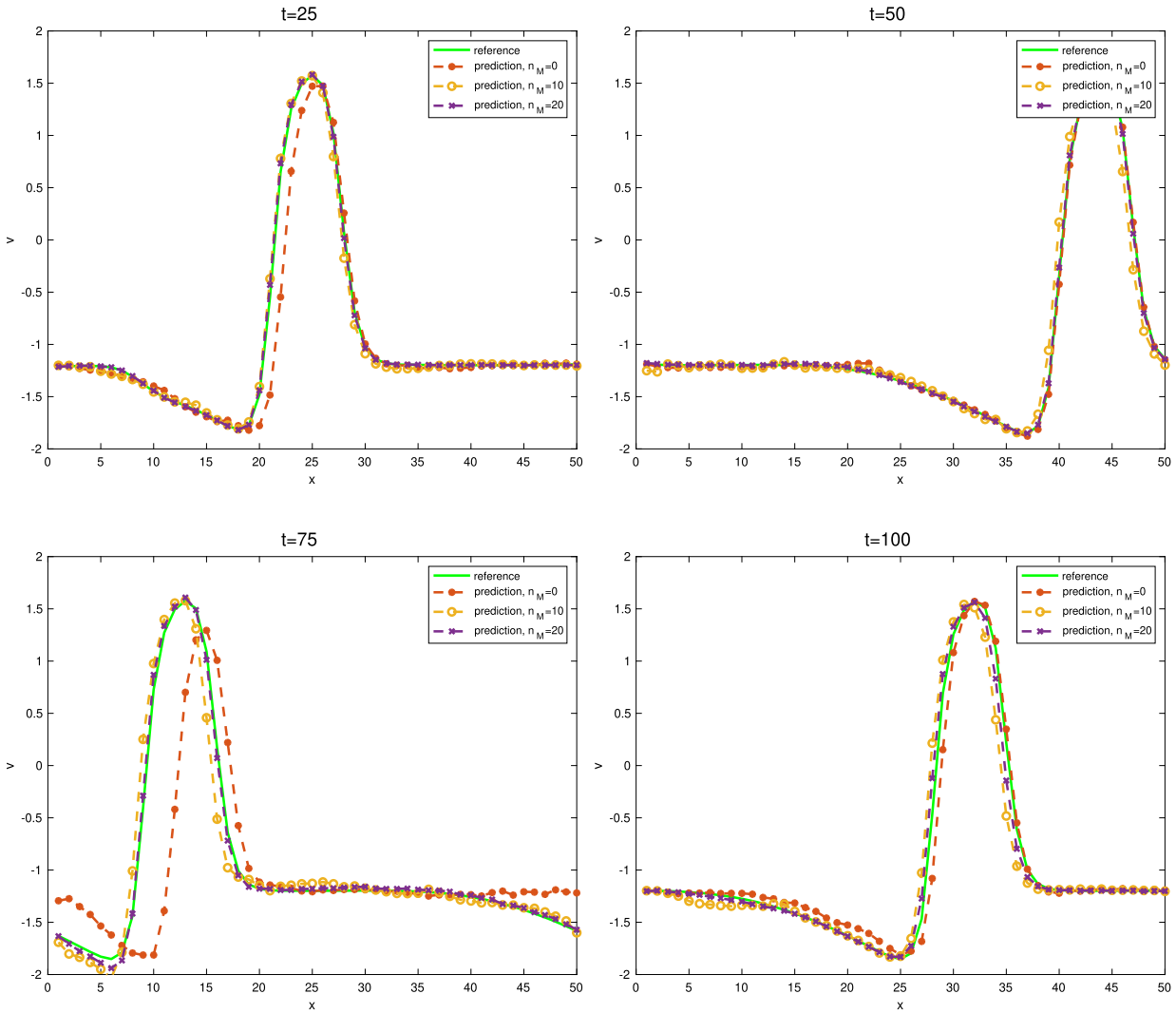


Fig. 4.3. 1D FitzHugh-Nagumo System – example test trajectory.

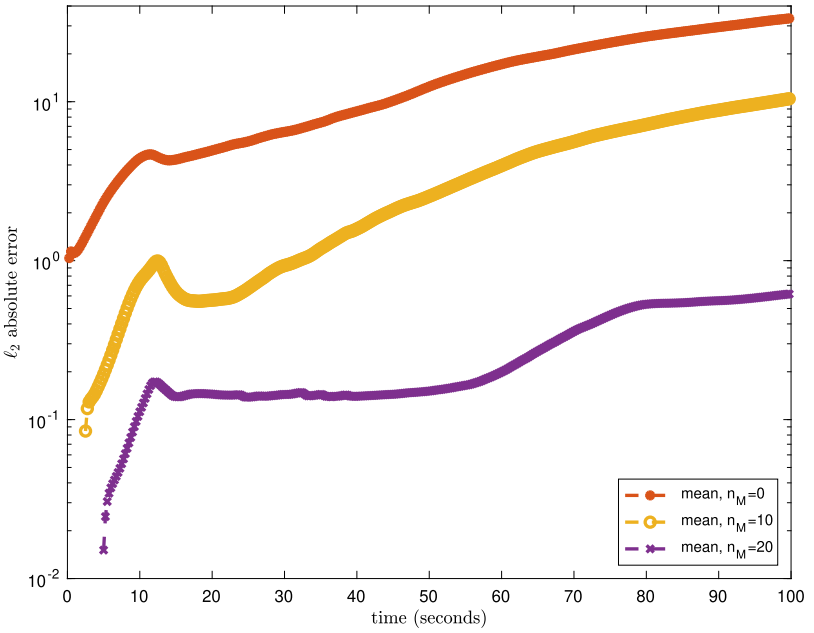


Fig. 4.4. 1D FitzHugh-Nagumo System – ℓ_2 absolute error over time.

$$\begin{bmatrix} v(x,0) \\ w(x,0) \end{bmatrix} = \begin{bmatrix} \exp(\sin(x)) \\ \exp(\cos(x)) \end{bmatrix}. \tag{4.3}$$

Ensemble prediction [3] using the 10 trained DNN models is carried out for 600 steps (total time $T = 6$ s) from only the v portion of the new initial condition (and the required memory). We see the results for an example test trajectory in Figs. 4.1. We see that the no-memory model is unable to learn the appropriate evolution, while $n_M = 10$ starts to get the correct shape. The $n_M = 20$ model is consistently the most accurate, as evidenced by the mean absolute ℓ_2 error over 100 test trajectories (created using (4.2)) displayed in Fig. 4.2. This verifies our mathematical motivation, which states memory terms are critical in modeling partially observed systems. Despite prediction time length far exceeding the training data sequence length, we still see accurate predictions in terms of ℓ_2 error.

4.2. Missing variable: FitzHugh-Nagumo system

We now consider the FitzHugh-Nagumo system with diffusion [8],

$$\begin{aligned} v_t &= v - \frac{1}{3}v^3 - w + I + D_v \Delta v, \\ w_t &= \epsilon(v + b - cw) + D_w \Delta w, \end{aligned} \tag{4.4}$$

a reaction-diffusion type system. We consider the case when only the solution component v is observed and seek to create an accurate DNN model for the evolution of v only. We present numerical studies in one dimension and two dimensions.

4.2.1. One dimension

In the following one-dimensional example, we set the parameters as $D_v = 0.01$, $D_w = 0$, $\epsilon = 0.08$, $b = 0.7$, $c = 0.8$, $I = 0$. Equation (4.4) is solved using a high-order numerical solver in the domain $[0, 5]$ on a uniform grid of $N = 50$ points and observed up to time $T = 7.5$ s with time step $\Delta t = 0.25$ (30 time steps), using $10,000$ randomized initial conditions in the form of the step functions

$$v(x,0) = \begin{cases} a_0, & m_1 \le x \le m_2 \\ a_1, & m_1 \ge x \ge m_2 \end{cases}, \tag{4.5}$$

$$w(x,0) = \begin{cases} b_0, & x \le m_1 \\ b_1, & x > m_1 \end{cases}, \tag{4.6}$$

where $a_0 \sim U[0.9, 1.1]$, $a_1 \sim U[-1.2, -1]$, $b_0 \sim U[-0.1, 0.1]$, $b_1 \sim U[-0.6, -0.4]$, $m_1 \sim U[0, 1.25]$, and $m_2 = m_1 + U[0.25, 1.5]$. These initial conditions along with periodic boundary conditions generate a traveling wave phenomenon. Upon solving the system, only the solution data of v are recorded into our training data set.

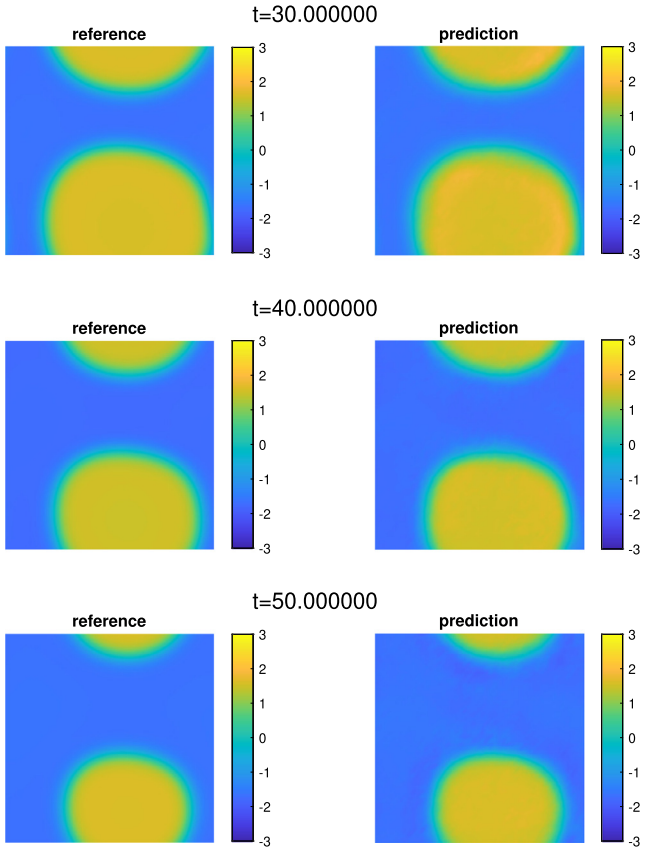


Fig. 4.5. 2D FitzHugh-Nagumo System: Comparison of DNN predictions of v with the reference solutions. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

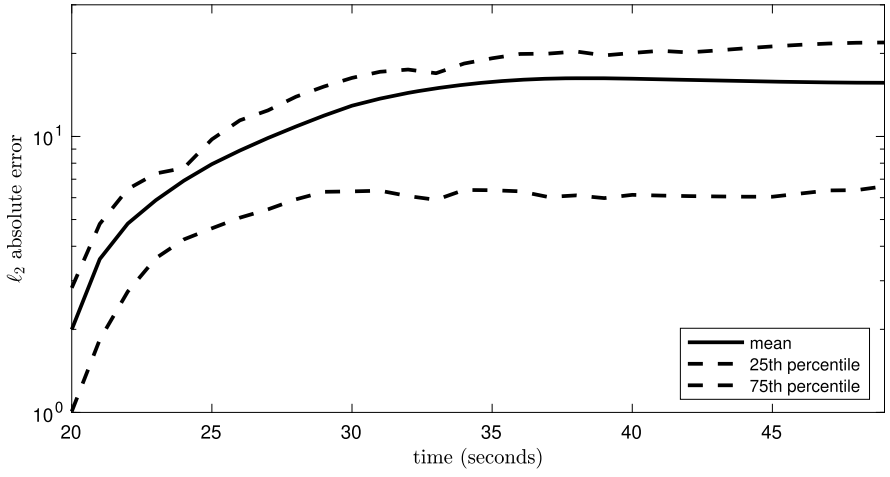


Fig. 4.6. 2D FitzHugh-Nagumo System – ℓ_2 absolute error over time.

Memory lengths of $n_M = 0, 10, 20$ are compared using a recurrent loss length of $K = 10$, resulting in a DNN structure with input lengths of $m \cdot N \cdot (n_M + 1) = 50, 550, 1050$. The $J = 5$ disassembly block channels each have 1 hidden layer with $n_c = 1000$ neurons. The assembly layer has $J = 5$ neurons. Hyperbolic tangent is used as the activation function for each layer. 10 DNN models are trained for 10,000 epochs each with a constant learning rate of 10^{-3}.

Testing data is created in the same way as training data (albeit with new random numbers). Ensemble prediction [3] using the 10 trained DNN models is carried out for 400 time steps (total time $T = 100$ s) from only the v portion of the

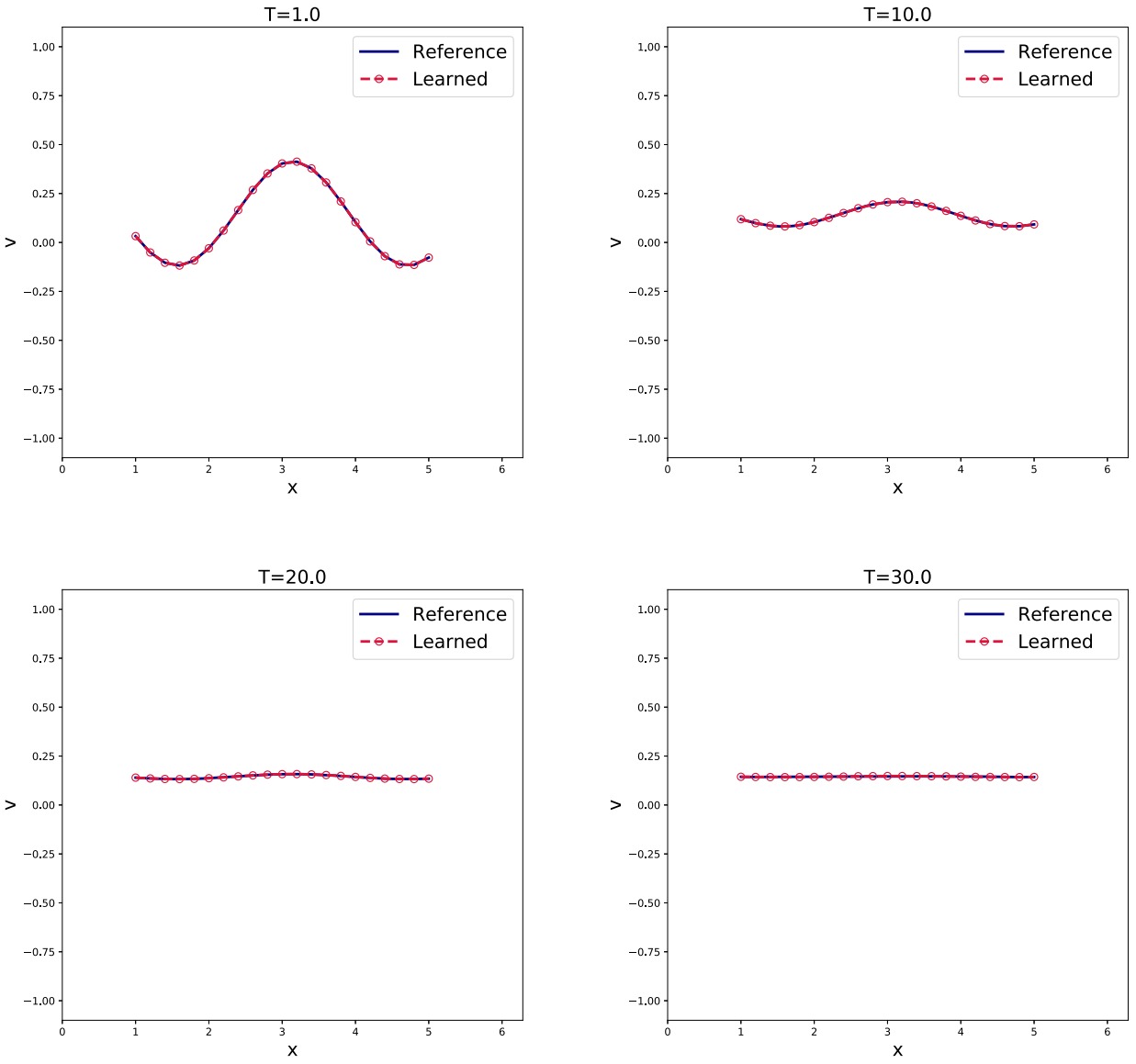


Fig. 4.7. DNN prediction of the 4th-order PDE in sub-domain $[1, 5]$, where the complete domain is $[0, 2\pi]$. (Data are missing in $\sim 36\%$ of the domain.)

new initial conditions (and the required memory). The results are shown in Fig. 4.3 for an example test trajectory, and in Fig. 4.4 for the mean absolute ℓ_2 error over 100 test trajectories. Again we see that a longer memory improves accuracy.

4.2.2. Two dimensions

In the following two-dimensional example, we set the parameters as $D_v = 0.1$, $D_w = 1.6$, $\epsilon = 0.08$, $b = 0.7$, $c = 0.8$, $I = 0.35$. Equation (4.4) is solved using a high-order numerical solver in the domain $[0, 10] \times [0, 10]$ on a uniform grid of $N = 50^2 = 2,500$ points and observed up to time $T = 30$ s with time step $\Delta t = 1$ (30 time steps), using $1,000$ randomized initial conditions in the form of random noise,

$$\begin{aligned} v(\mathbf{x}_{i,j}, 0) &\sim U[-1.1, -0.9] \\ w(\mathbf{x}_{i,j}, 0) &\sim U[-0.4, -0.2] \end{aligned} \tag{4.7}$$

for $i, j = 1, \ldots, N$. These initial conditions along with periodic boundary conditions generate a phenomenon whereby one or more blobs amass from the random noise and then grow and decay, eventually settling to a steady state.

A memory length of $n_M = 20$ is used with a recurrent loss length of $K = 10$, resulting in a DNN structure with input length $m \cdot N \cdot (n_M + 1) = 52,500$. The $J = 5$ disassembly block channels each have 1 hidden layer with $n_c = 100$ neurons. Using such a small number of neurons allows us to keep the total number of parameters lower given the large input size.

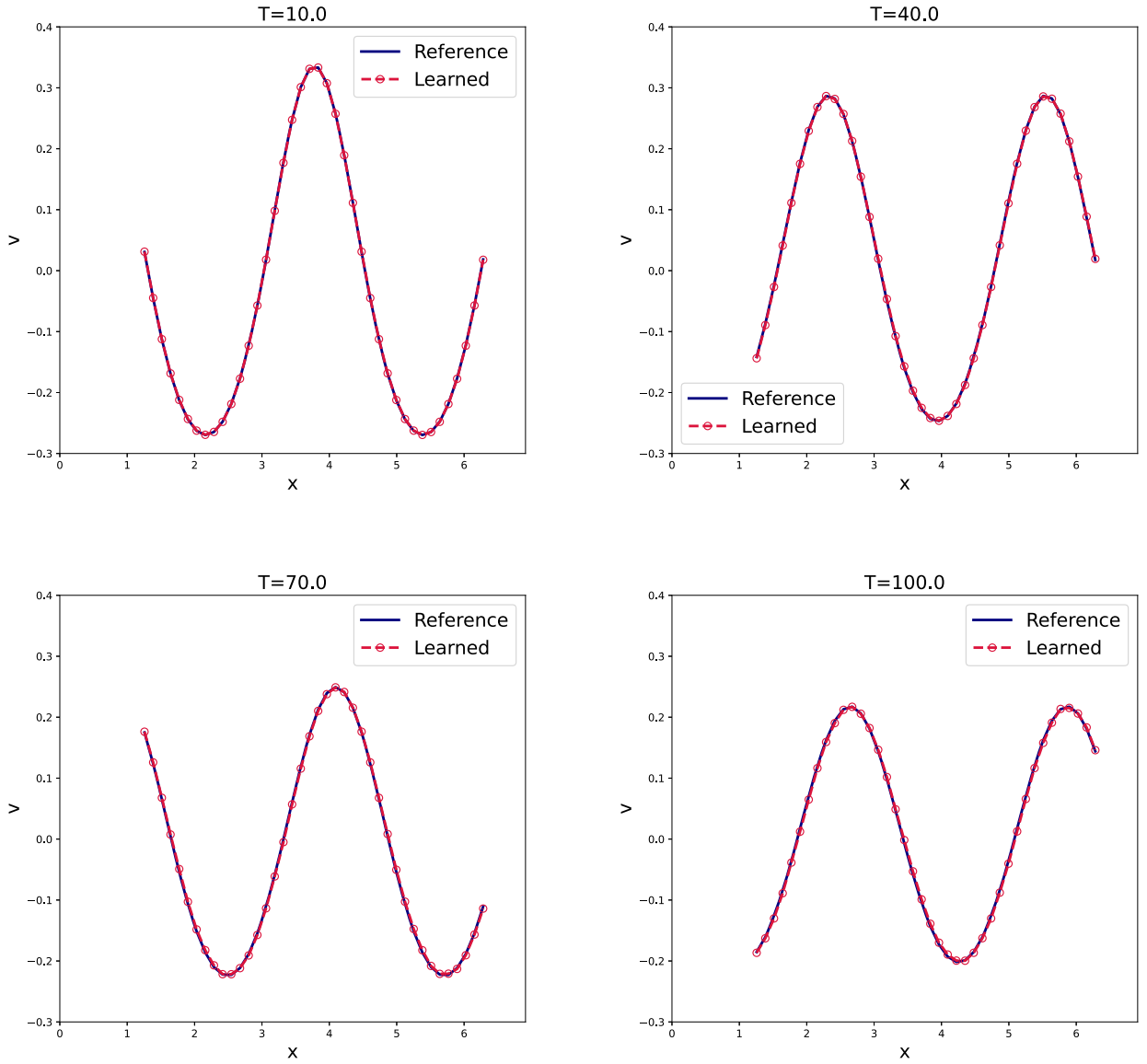


Fig. 4.8. Advection-diffusion with 20% of missing domain. DNN prediction with the initial condition $u_0^{(1)}(x)$ in sub-domain $[\frac{2}{5}\pi, 2\pi]$.

The assembly layer has $J = 5$ neurons. Hyperbolic tangent is used as the activation function for each layer. 10 DNN models are trained for $10,000$ epochs each with a constant learning rate of 10^{-3}.

Testing data is created in the same way as training data (albeit with new random numbers). Ensemble prediction [3] using the 10 trained DNN models is carried out for 50 time steps (total time $T = 50$ s) from only the v portion of the new initial conditions (and the required memory). The results are shown in Fig. 4.5 for an example test trajectory and in Fig. 4.6 for the mean absolute ℓ_2 error over 100 test trajectories. For this memory level, we see good accuracy that levels out over a long time period.

4.3. Missing domain: fourth-order equation

We consider the following 4th-order equation:

$$\frac{\partial u}{\partial t} + c\frac{\partial^4 u}{\partial x^4} = 0, \; x \in [0, 2\pi], \tag{4.8}$$

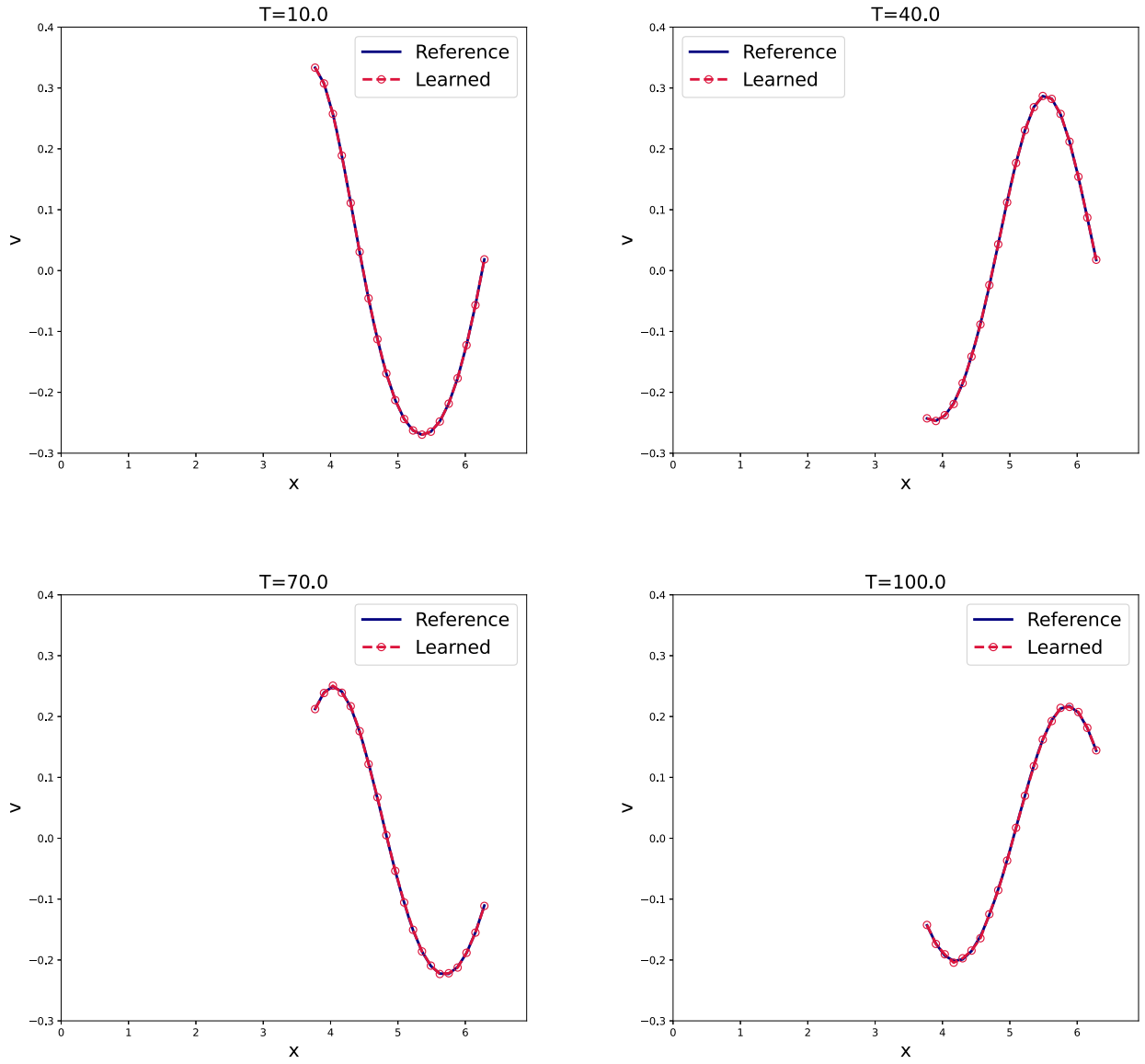


Fig. 4.9. Advection-diffusion with 60% of missing domain. DNN prediction with the initial condition $u_0^{(1)}(x,0)$ in sub-domain $[\frac{6}{5}\pi, 2\pi]$.

with periodic boundary conditions and $c = 10^{-2}$. The equation is solved numerically over a uniform grid with $\Delta x = 0.05$ and randomized initial conditions. We assume the solution data are only available in the sub-domain $[1, 5]$, which is about $4/2\pi \approx 64\%$ of the entire domain. In other words, solution data are missing from about 36% of domain.

The training data are generated by initial conditions as randomized Fourier series

$$u(x,0) = a_0 + \sum_{n=1}^{N_c} (a_n \cos(nx) + b_n \sin(nx)), \tag{4.9}$$

where $N_c = 7$, $a_0 \sim U[-1/2, 1/2]$ and $a_n^{(i)}, b_n^{(i)} \sim U[-1/n, 1/n]$, $1 \le n \le N_c$. For the time march step, we take $\Delta t = 0.1$ and end time $T = 5$. With the training data recorded only in the sub-domain $[1, 5]$, we employ our DNN structure by setting the recurrent loss step $K = 10$, $J = 5$ disassembly block channels each have 2 hidden layers with 21 neurons.

After the DNN model is trained, we perform various validation tests. Here we present the results based on an initial condition of $u(x,0) = \exp(-\sin^2 x) - 1/2$ up to a termination time $T = 50$. We present the results for memory length $n_M = 20$ in Fig. 4.7. We observe very good agreement between the DNN prediction and the reference solutions.

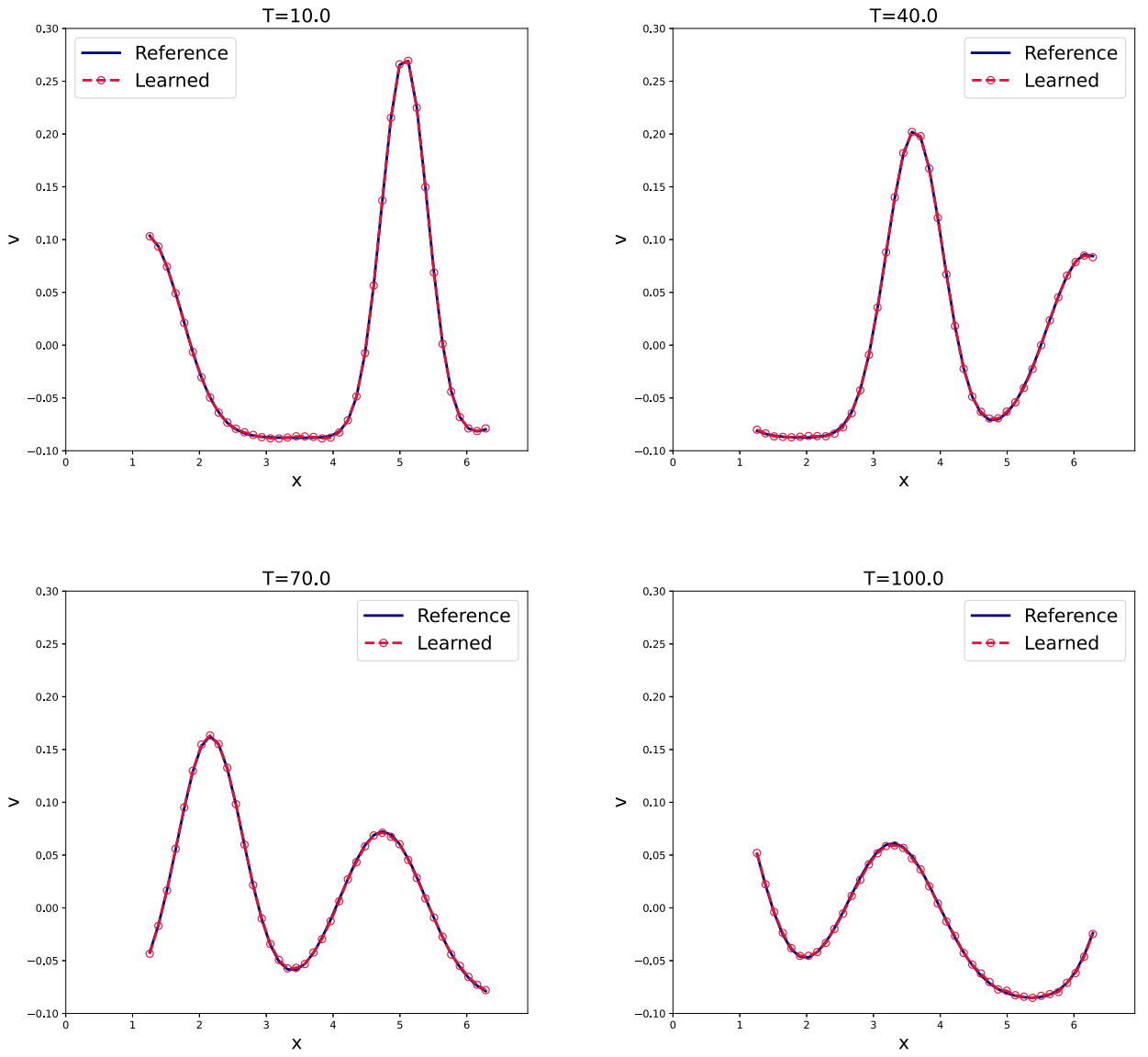


Fig. 4.10. Advection-diffusion with 20% of missing domain. DNN prediction with the initial condition $u_0^{(2)}(x)$ in sub-domain $[\frac{2}{5}\pi, 2\pi]$.

4.4. Missing domain: advection-diffusion equation

We consider the following advection-diffusion equation:

$$\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x} + \kappa \frac{\partial^2 u}{\partial x^2}, \qquad x \in [0, 2\pi], \tag{4.10}$$

with periodic boundary conditions, and $\kappa = 10^{-3}$.

We consider two cases: (i) solution data are available in the sub-domain $[\frac{2}{5}\pi, 2\pi]$, and (ii) solution data are available in the sub-domain $[\frac{6}{5}\pi, 2\pi]$. The first case represents 20% of the domain missing, and the second case has 60% of the domain missing.

The training data are generated by solving the equation using the same randomized Fourier series from (4.9) in the previous section as initial conditions. The time step is set at $\Delta t = 10^{-1}$ and the end time at $T = 10$. The DNN structure has $J = 5$ disassembly block channels, each of which has 1 hidden layers with 40 neurons. The recurrent loss length is $K = 20$.

For validation tests, we consider two initial conditions:

$$u_0^{(1)}(x) = e^{-\sin^2 x} - \frac{1}{2}, \quad u_0^{(2)}(x) = \frac{2}{5} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{1}{5} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} - \frac{1}{5},$$

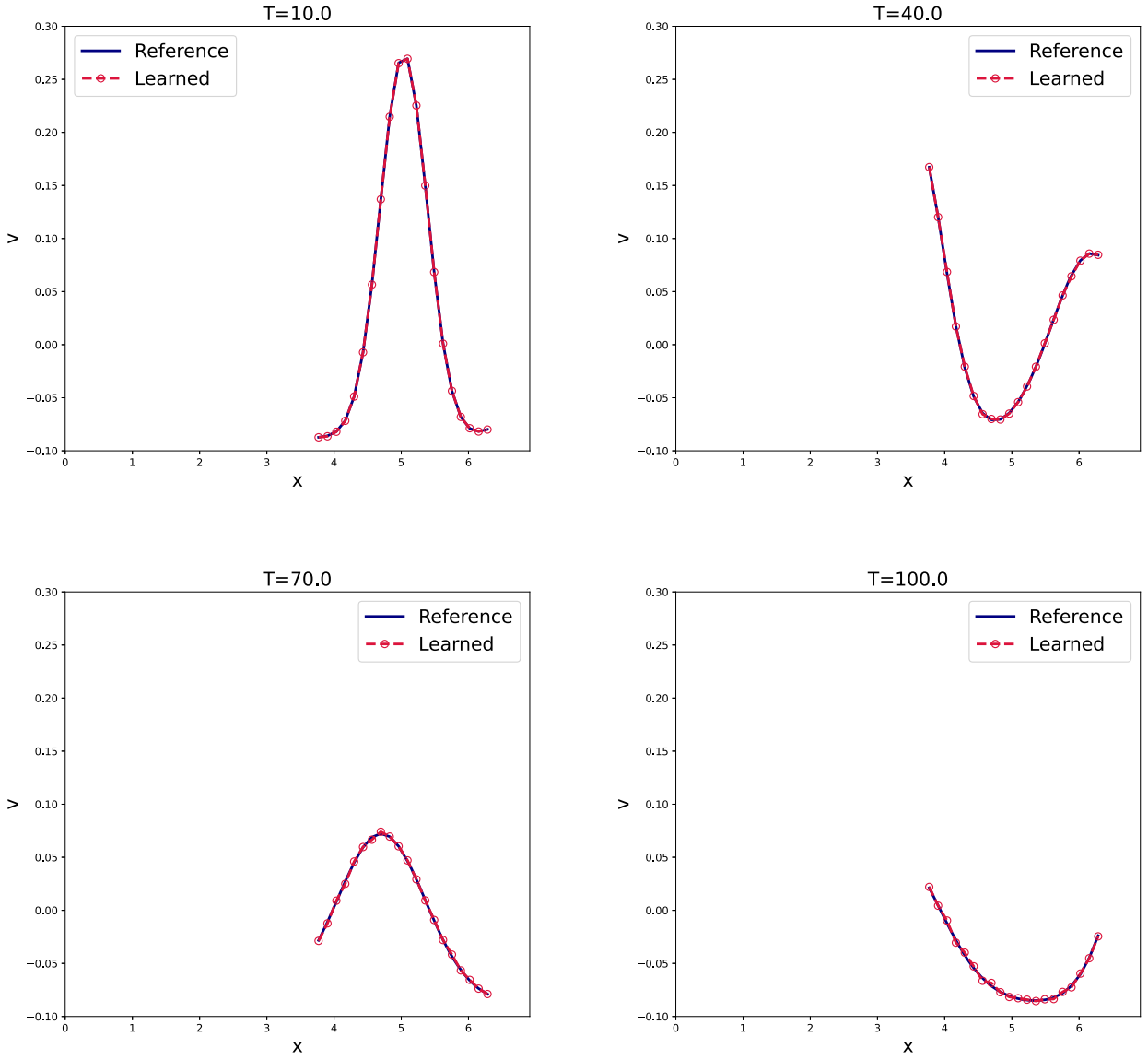


Fig. 4.11. Advection-diffusion with 60% of missing domain. DNN prediction with the initial condition $u_0^{(2)}(x,0)$ in sub-domain $[\frac{6}{5}\pi, 2\pi]$.

where $\mu_1 = \frac{2}{5}\pi, \mu_1 = \frac{6}{5}\pi, \sigma_1 = 0.3, \sigma_2 = 0.5$ The DNN prediction results for the first initial condition are shown in Figs. 4.8, 4.9, for the case of sub-domain $[\frac{2}{5}\pi, 2\pi]$ (20% missing domain) and sub-domain $[\frac{6}{5}\pi, 2\pi]$ (60% missing domain), respectively. Compared to the reference solutions, we note the DNN model produces fairly accurate predictions, even in the case when 60% of the domain is without any observational data. The results for the 2nd initial conditions are shown in Figs. 4.10, 4.11, for the case of sub-domain $[\frac{2}{5}\pi, 2\pi]$ (20% missing domain) and sub-domain $[\frac{6}{5}\pi, 2\pi]$ (60% missing domain), respectively. We again observe accurate predictions made by the DNN model.

4.5. Missing domain: viscous Burger's equation

We now consider the viscous Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \qquad x \in [-\pi, \pi], \tag{4.11}$$

with periodic boundary conditions and viscosity set to be $\nu = 0.1$.

We consider the case when only data in sub-domain $[-\frac{3}{5}\pi, \frac{4}{5}\pi]$ are observed. This amounts to 30% of the domain missing. The training data are generated on a uniform grid with $\Delta x = \pi/25$, using the same randomized initial conditions

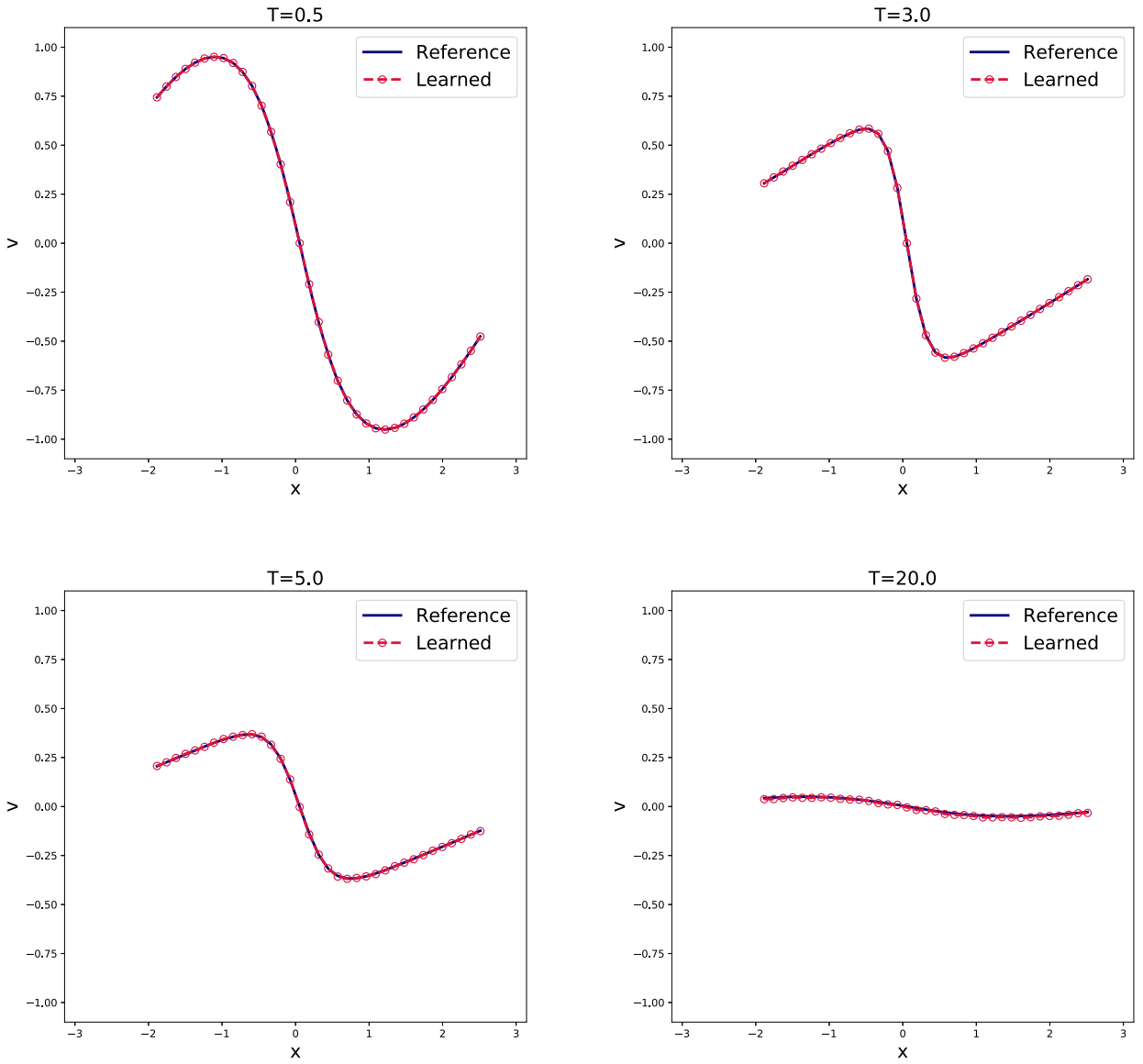


Fig. 4.12. Viscous Burger's equation with 30% missing domain. DNN prediction in sub-domain $[-\frac{3\pi}{5}, \frac{4\pi}{5}]$.

as (4.9) in the previous section, over a time step $\Delta t = 0.1$ up to a termination time $T = 3$. The DNN structure again has $J = 5$ disassembly block channels, each of which has 1 hidden layers with 40 neurons, and the recurrent loss length is $K = 20$.

Validation results using an initial condition $u_0(x) = -\sin x$ are shown in Fig. 4.12, for time up to $T = 100$. We observe good agreement of the DNN results when compared to the reference solution.

4.6. Missing domain: inviscid Burger's equation

We now consider the inviscid Burgers equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \qquad x \in [-\pi, \pi], \tag{4.12}$$

with periodic boundary conditions. This is a more challenging case, as the solution may develop shocks within finite time. Same as before, the equation is solved over uniform grids and a time step $\Delta = 0.05$, by using the same randomized initial conditions as before (4.9). We consider two cases: data are available in the sub-domain $[-\frac{3}{5}\pi, \pi]$ (20% of the domain missing) and in the sub-domain $[\frac{1}{5}\pi, \pi]$ (60% of the domain missing). Our DNN structure has $J = 5$ disassembly block channels, each of which has 1 hidden layer with 50 neurons. The recurrent loss length is set at $K = 20$.

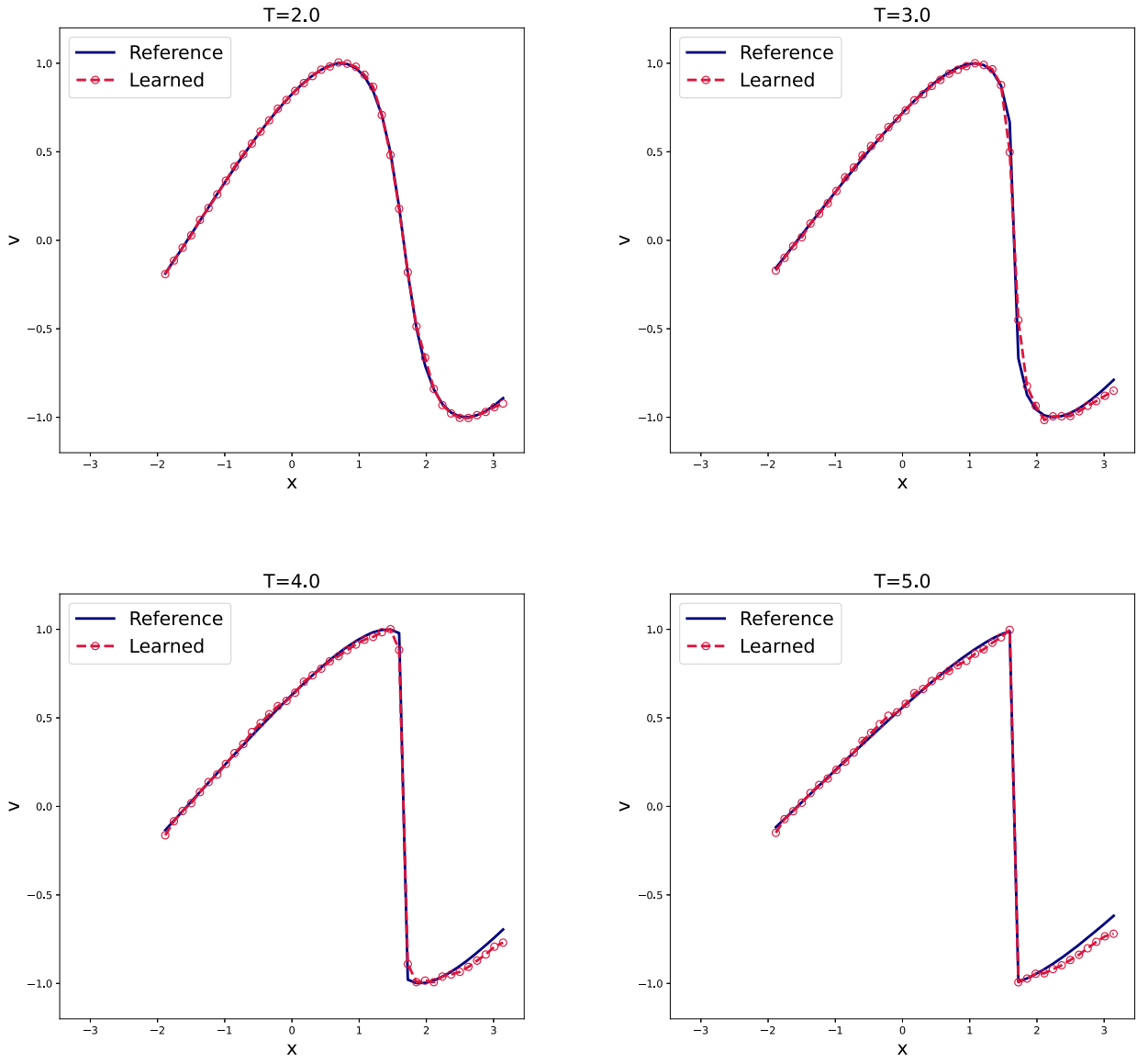


Fig. 4.13. Learned results of Inviscid Burgers equation in partial domain $[-\frac{3\pi}{5}, \pi]$, the complete physical domain is $[-\pi, \pi]$.

For validation, we present the DNN prediction results using an initial condition $u_0(x) = \cos x$, for up to a termination time $T = 6$. The results of the first case with 20% missing domain are shown in Fig. 4.13. We observe that the DNN model captures the development of the shock with fairly good accuracy. For the second case of 60% missing domain, the results are shown in 4.14. Again, the DNN model correctly captures the development of the shock. Although the overall accuracy is relatively lower, compared to the other results we have presented here, we remark that this is a very challenging case: a nonlinear problem with shocks with a large portion of the domain 60% without data.

5. Conclusion

We have presented a new approach for learning unknown partially-observed PDEs from their solution data using DNNs by making significant generalizations to the existing work on learning dynamical systems as well as PDEs. Our method is a notable advancement of data driven modeling, as it allows one to model PDEs when data are not available for certain state variables or in certain parts of the domain. Future work will focus on accelerating training of large models for two-dimensional examples through learning reduced representations.

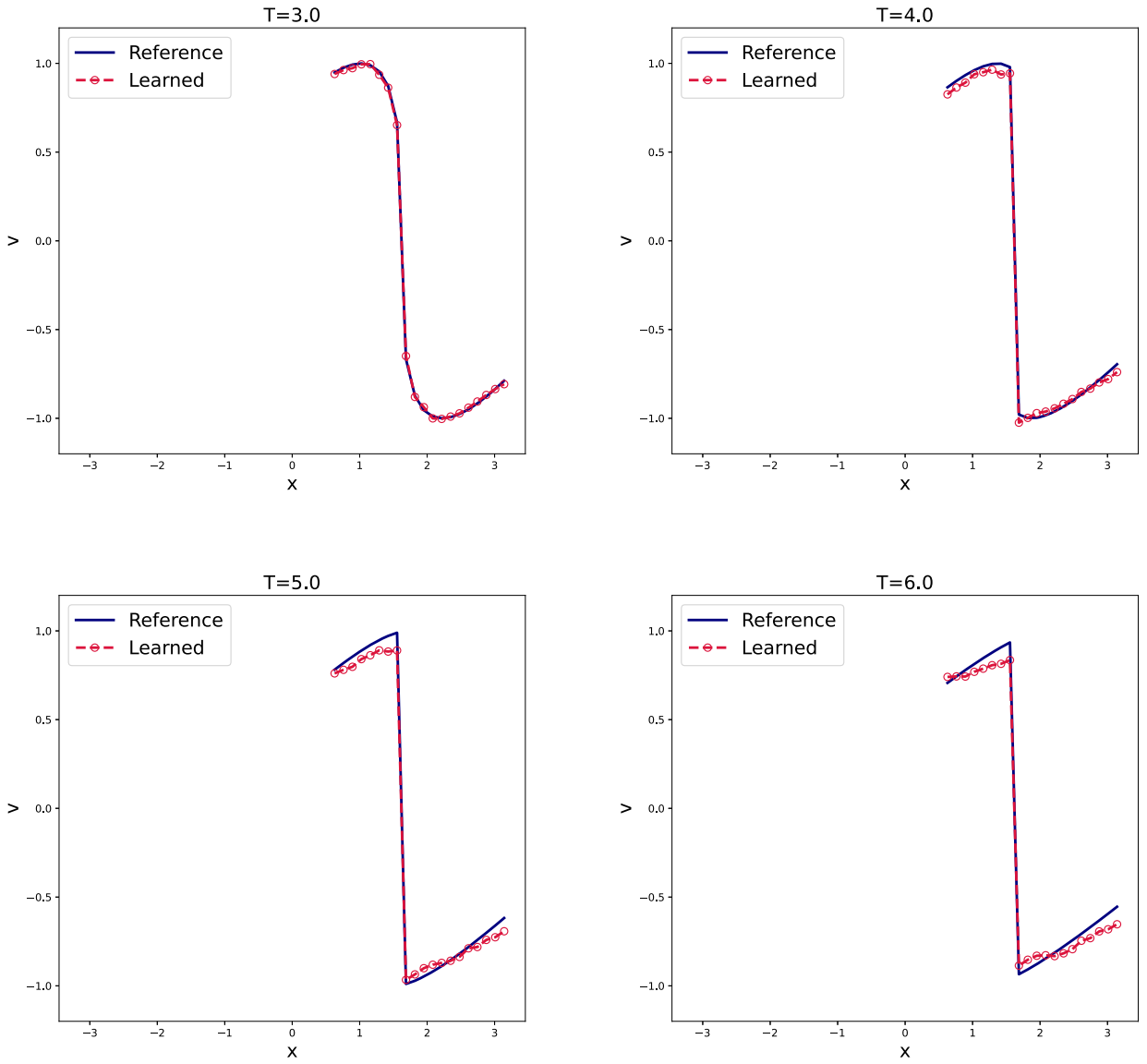


Fig. 4.14. Learned results of Inviscid Burgers equation in partial domain $[\frac{1\pi}{5}, \pi]$, the complete physical domain is $[-\pi, \pi]$.

CRediT authorship contribution statement

The authors have approximately equal contributions to the paper. Each author contributed to all aspects of the work.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Dongbin Xiu reports financial support was provided by Air Force Office of Scientific Research FA9550-22-1-0011.

Data availability

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

References

[1] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. USA 113 (2016) 3932–3937.
[2] Z. Chen, V. Churchill, K. Wu, D. Xiu, Deep neural network modeling of unknown partial differential equations in nodal space, J. Comput. Phys. 449 (2022) 110782.
[3] V. Churchill, S. Manns, Z. Chen, D. Xiu, Robust modeling of unknown dynamical systems via ensemble averaged learning, arXiv preprint, arXiv:2203.03458, 2022.
[4] V. Churchill, D. Xiu, Deep learning of chaotic systems from partially-observed data, arXiv preprint, arXiv:2205.08384, 2022.
[5] V. Churchill, D. Xiu, Learning fine scale dynamics from coarse observations via inner recurrence, arXiv preprint, arXiv:2206.01807, 2022.
[6] X. Fu, L.-B. Chang, D. Xiu, Learning reduced systems via deep neural networks with memory, J. Mach. Learn. Model. Comput. 1 (2020) 97–118.
[7] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[8] E.M. Izhikevich, R. FitzHugh, Fitzhugh-Nagumo model, Scholarpedia 1 (2006) 1349.
[9] S.H. Kang, W. Liao, Y. Liu, IDENT: identifying differential equations with numerical time evolution, arXiv preprint, arXiv:1904.03538, 2019.
[10] N. Kovachki, S. Lanthaler, S. Mishra, On universal approximation and error bounds for Fourier neural operators, J. Mach. Learn. Res. 22 (2021).
[11] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: learning maps between function spaces, arXiv preprint, arXiv:2108.08481, 2021.
[12] Z. Li, N.B. Kovachki, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, et al., Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2020.
[13] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, arXiv preprint, arXiv:1812.04426, 2018.
[14] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: learning PDEs from data, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 3208–3216.
[15] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229.
[16] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, SIAM Rev. 63 (2021) 208–228.
[17] H. Mori, Transport, collective motion, and Brownian motion, Prog. Theor. Phys. 33 (1965) 423–455.
[18] T. Qin, Z. Chen, J. Jakeman, D. Xiu, Deep learning of parameterized equations with applications to uncertainty quantification, Int. J. Uncertain. Quantificat. 11 (2) (2021) 63–82, https://doi.org/10.1615/Int.J.UncertaintyQuantification.2020034123.
[19] T. Qin, Z. Chen, J. Jakeman, D. Xiu, Data-driven learning of non-autonomous systems, SIAM J. Sci. Comput. 43 (2021) A1607–A1624.
[20] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[21] M. Raissi, Deep hidden physics models: deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (2018) 1–24.
[22] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10561, 2017.
[23] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (Part II): data-driven discovery of nonlinear partial differential equations, arXiv preprint, arXiv:1711.10566, 2017.
[24] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv preprint, arXiv:1801.01236, 2018.
[25] S.H. Rudy, S.L. Brunton, J.L. Proctor, J.N. Kutz, Data-driven discovery of partial differential equations, Sci. Adv. 3 (2017) e1602614.
[26] S.H. Rudy, J.N. Kutz, S.L. Brunton, Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, J. Comput. Phys. 396 (2019) 483–506.
[27] H. Schaeffer, Learning partial differential equations via data discovery and sparse optimization, Proc. R. Soc. Lond. A, Math. Phys. Eng. Sci. 473 (2017).
[28] H. Schaeffer, S.G. McCalla, Sparse model selection via integral terms, Phys. Rev. E 96 (2017) 023302.
[29] H. Schaeffer, G. Tran, R. Ward, Extracting sparse high-dimensional dynamics from limited data, SIAM J. Appl. Math. 78 (2018) 3279–3295.
[30] Y. Sun, L. Zhang, H. Schaeffer, NeuPDE: neural network based ordinary and partial differential equations for modeling time-dependent data, arXiv preprint, arXiv:1908.03190, 2019.
[31] G. Tran, R. Ward, Exact recovery of chaotic systems from highly corrupted data, Multiscale Model. Simul. 15 (2017) 1108–1129.
[32] Q. Wang, N. Ripamonti, J. Hesthaven, Recurrent neural network closure of parametric POD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, J. Comput. Phys. 410 (2020) 109402.
[33] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[34] R. Zwanzig, Nonlinear generalized Langevin equations, J. Stat. Phys. 9 (1973) 215–220.