Received 1 April 2024, accepted 23 April 2024, date of publication 30 April 2024, date of current version 7 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3395517

METHODS

Sequence-to-Sequence Stacked Gate Recurrent Unit Networks for Approximating the Forward Problem of Partial Differential Equations

ZHAOYANG ZHANG AND QINGWANG WANG, (Member, IEEE)
Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China

Corresponding author: Qingwang Wang (wangqingwang@kust.edu.cn)

This work is funded in part by the Yunnan Fundamental Research Projects under Grant 202401AW070019, 202101BE070001-008 and 202301AV070003, in part by the Youth Project of the National Natural Science Foundation of China under Grant 62201237, in part by the Major Science and Technology Projects in Yunnan Province under Grant 202302AG050009 and 202202AD080013.

ABSTRACT We proposed an optimisation algorithm based on the sequence-to-sequence (Seq2Seq) stacking of the gate recurrent unit (GRU) model to characterise and approximate the forward problem of partial differential equations (PDEs). Unlike traditional methods based on mesh differential approximation and no parameters, this is a meshless approach based on parametric semi-supervised learning. Specifically, the algorithm employs the ability of deep feedback neural networks to approximate continuous dynamical systems, which is enhanced by stacked GRU modules to capture the evolution of the PDEs over time and thus enrich the representations of the sequences in the hidden space. The loss function of the model incorporates partial physical knowledge as an a priori condition to guide the optimisation direction, i.e., transforming the numerical iterative problem into a non-convex optimisation problem. In addition, each round of training of the model incorporates data resampling to prevent it from overfitting. We evaluated the ability of the proposed algorithm to solve mathematical physics equations for a variety of differential operators and constraints, including the heat, wave, Burgers, Schrodinger, diffusion, and Kovasnay flow equations. The experimental results confirmed the outstanding prediction precision and generalisation capability of the proposed algorithm.

INDEX TERMS Forward problem of partial differential equations, deep learning, gate recurrent unit, sequence-to-sequence.

I. INTRODUCTION

The forward problem of partial differential equations (PDEs) can describe different physical phenomena, that is, different physical phenomena can be governed by the same laws, reflecting the comparability of various physical phenomena. Consequently, its study is of major significance to mathematics and theoretical physics. However, as PDEs are nonlinear or contain complex special functions in several practical situations, typically, analytical solutions are nonexistent or difficult to determine. Thus, appropriate numerical algorithms must be designed to approximate the numerical solution of the forward problem. Traditional numerical algorithms for solving forward problems include the finite element method (FEM [1]), finite difference method (FDM [2]), and finite volume method (FVM [3]). Owing to the requirement of dividing the solution domain into discrete grids, the method and scale of mesh delineation can restrict the precision and efficacy of the computation. Moreover, instable and non-convergent numerical solutions may result from improper mesh selection.

In the past decades, deep learning based on deep neural networks has made impressive achievements in the fields of natural language processing, computer vision, and

The associate editor coordinating the review of this manuscript and approving it for publication was Ángel F. García-Fernández.

multimodal scene modelling. It has also demonstrated its potential and advantages in processing high-dimensional data and solving complex nonlinear problems [4], [5], [6]. The classic concept of function approximation has been included in the approximate solution of the forward problem of PDEs. The theory of function approximation is an essential branch of contemporary mathematics. Wierstrass proved that continuous functions can be approximated using polynomial functions [7], and Chebyshev formulated the best approximation theorem [8]. In addition, approximation theories based on wavelet bases [9], algebraic or trigonometric polynomials [10], and deep neural networks have been rapidly developed. The universal approximation theorem of neural networks [11], [12] has shown that in the case of a sufficiently wide and deep network, any continuous and complex function can be approximated with arbitrary precision. Thus, deep neural networks exhibit a very good nonlinear approximation ability. Its essence is to express and approximate unary or multivariate functions via the expansion of nonlinear parameters. Therefore, compared to general approximation, deep neural networks exhibit superior nonlinear approximation abilities. According to the PAC theory [13], deep learning algorithms can learn a nearly correct hypothesis (local optimal solution) with a certain probability. This is expressed as:

$$P(|\mathcal{R}(f) - \hat{\mathcal{R}}(f)| \leq \epsilon) \geq 1 - \delta, \tag{1}$$

where $\mathcal{R}(f)$ is the expected risk, $\hat{\mathcal{R}}(f)$ is the empirical risk of the model, and f denotes the machine learning model. Most of the local optimal solutions are equivalent as the size of the network increases during optimization [14], while the number of training samples N is related to the error precision ε and the probability δ of approximate correctness as follows:

$$N(\varepsilon, \delta) \geq \frac{1}{2\varepsilon^2}(\log|\mathcal{F}| + \log\frac{\delta}{2}), \tag{2}$$

where $\mathcal{F}$ denotes the function cluster of the model and $|\mathcal{F}|$ denotes the hypothesis space size. Based on the optimization principles of deep learning, researchers have developed numerous outstanding algorithms for solving forward problems of PDEs. The known physical constraints were added to the training and optimization process of the neural network to facilitate its adaptation to and learning of real physical phenomena while simultaneously using the feed-forward neural network to optimize the approximate solution of the PDEs [15], while the correlation capture of the PDEs itself was relatively weak. In contrast to traditional finite element methods, the Deep Galerkin Method (DGM) need not discretize PDEs, and it uses a feedforward neural network and a long short-term memory (LSTM) [16] network to approximate the approximate solution of PDEs [17]. It also uses the Monte Carlo method [18] to approximate the second derivative, and exhibits beneficial computational efficiency for PDEs in high-dimensional data instances. In addition, operator theory-based deep learning methods may also aid in the development of the forward and backward problems of PDEs [19]. This study's neural operators are composed of a linear integral operator and a point-by-point nonlinear activation function, which use two sub-networks (backbone and branch networks) to capture the relationship between the input function and the local operator, thus closely approximating the nonlinear operator. Fourier transform-based deep learning methods are based on operator learning and approximate the solution of PDEs using frequency domain information [20]. It can also learn the generalized functions between inputs and outputs, which can partially solve the retraining problem of PINNs. In addition, there exists a method based on neural differential equations that expresses the PDEs as a set of coupled ordinary differential equations and obtains the numerical solution of the PDEs by solving this set of ordinary differential equations [21]. In terms of correcting the loss function, [22] proposed a convolutional neural network based technique to reconstruct the loss function thereby improving the prediction accuracy. In terms of application, [23] utilized physically informative neural networks to predict excess pore water pressure in two-dimensional soil consolidation and successfully applied it in real railroads. These studies [24], [25], [26], [27], [28], and [29] are based on certain research variants of PINN [15] and applications in different fields. Furthermore, certain scholars are working on theoretical studies of recurrent neural network (RNN)-like linear system approximation capabilities [30], [31], [32], [33].

However, these studies do not expressly reflect the temporal correlation of time-containing PDEs, and their sampling of training samples lacks diversity. To this end, we propose a stacked GRU network optimization algorithm in the sequence-to-sequence(Seq2Seq) context, based on the capability of deep feedback neural networks to approximate continuous dynamical systems, in order to characterize and approximate the forward problem of time-dependent PDEs. The direction of optimisation is constrained by using the PDE constraints as a priori knowledge of the loss function for the network. Calculating PDE derivatives using automatic differentiation methods [34], which do not produce truncation errors due to differential computation, requires vectors of spatio-temporal sequences to train that have been randomly sampled under the associated constraints, avoiding the granularity issue of the grid. The trained model is able to address a variety of time-dependent forward problem contexts more effectively while also having high generalisation and modifiability since the GRU structure of the hidden layer is used to capture the correlation of time in sequences.

The contributions of this study are as follows:

1) We proposed an optimization algorithm based on Seq2Seq superimposed GRU networks for characterizing and approximating the forward problem of PDEs. The constraints of the PDE were jointly constituted as penalty terms in the loss function of the model, where the initialization and boundary conditions served as the direct prior knowledge to modify the model,

and the operator conditions served as the indirect prior knowledge to guide the model, thus transforming the solution of the PDE from a numerical iterative problem into a semi-supervised learning problem for non-convex optimization.

2) The proposed algorithm is parameter learning-based and meshless, and its numerical inference is without differential iterations. Moreover, unlike general deep feedforward neural networks, the proposed algorithm increases the feature of capturing the temporal correlation between PDEs input sequences, which enriches the expressiveness of hidden variable features and the generalization ability of the model.

3) The proposed algorithm was well-suited for solving various types of elliptic, parabolic, and hyperbolic PDEs, and both time-dependent and non-time-dependent variations, including heat conduction, string vibrations, Burgers equation, Schrodinger equation, diffusion equation, and Kovasznay flow in fluid dynamics. The numerical errors were typically in the range of 1×10^{-5} -1×10^{-7}, which confirmed the accuracy and generalizability of the proposed model.

The remainder of the paper is structured as follows. Section II introduces the related approximation lemmas and proposes an optimization algorithm based on a synchronized Seq2Seq stacked GRU network model to characterize and approximate the PDE of problem. Section III presents the application of the proposed algorithm to a variety of classical mathematical physics scenarios with various types of operators and constraints to evaluate and verify the prediction accuracy and generalization of the proposed algorithm. Section IV discusses the accuracy and generalization of the proposed model in experiments, and some reflections on the pending improvements of the model in the computational process. In Section V, relevant conclusions and future work are summarized. Moreover, the proof procedure of the lemmas presented in this paper is given in the appendix.

II. METHODOLOGY

In this section, we present a brief summary of the capability of deep feedback neural networks to approximate nonlinear continuous dynamical systems from the perspective of function approximation. Consequently, we propose a Seq2Seq based stacked GRU network model to characterize and approximate the forward problem of time-dependent PDEs.

A. CAPACITY OF RNN TO APPROXIMATE CONTINUOUS DYNAMIC SYSTEMS

In an RNN, neurons can receive information from other neurons and from themselves; that is, the feedback connection from hidden layer to hidden layer is added, which forms a network structure with loops. First, a general approximation theorem [11] is considered when describing the evolution of a dynamical system over time t. For multilayer feedforward neural networks, during nonlinear continuous dynamic systems with input, the output trajectory over a finite time period can be approximated with arbitrary precision using a class of RNNs. A continuous RNN is of the form:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = -\alpha \boldsymbol{x}(t) + A\sigma(\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)), \tag{3}$$

where $x \in \mathbb{R}^L$ is the state vector of L neurons, $u \in \mathbb{R}^m$ is the input vector of the network model, $A \in \mathbb{R}^{L\times L}$ is the connection weight matrix between L neurons, $B \in \mathbb{R}^{L\times m}$ is the connection weight matrix of the input signal to each neuron, and $\alpha \in \{0, 1\}$ is the delay factor, and $\sigma(.)$ is a nonlinear vector function with continuous first-order derivatives. After learning the weight matrix, the continuous nonlinear dynamical system can be arbitrarily approximated as:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{4}$$

if F is Lipschitz continuous, then the following lemma can be obtained:

Lemma 1: For any $\epsilon > 0$, let S, U be the open sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, $F, \tilde{F} : S \times U \to \mathbb{R}^n$ are Lipschitz continuous and continuous mappings respectively, L is the Lipschitz constant of $F(\boldsymbol{x}, \boldsymbol{u})$ over $S \times U$ for $\boldsymbol{x}$, and for all $\boldsymbol{x} \in S$ and $\boldsymbol{u} \in U$ satisfy

$$||F(\boldsymbol{x}, \boldsymbol{u}) - \tilde{F}(\boldsymbol{x}, \boldsymbol{u})|| < \epsilon,$$

if $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ are solutions of equations

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t), u(t))$$

and

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}(t)}{\mathrm{d}t} = \tilde{F}(\tilde{\boldsymbol{x}}(t), u(t))$$

satisfying initial condition $\boldsymbol{x}(t_0) = \tilde{\boldsymbol{x}}(t_0) \in S$ on the interval T, respectively, then we have

$$||\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}(t)|| \leq \frac{\epsilon}{L}(\exp L|t - t_0| - 1).$$

Lemma 2: For any $\epsilon > 0$, let S, U be the open sets in $\mathbb{R}^n$ and $\mathbb{R}^m$, X and D_U are tight sets in S and U, respectively, $F : S \times U \to \mathbb{R}^n$ is a vector function with a continuous first-order derivative function, consider the nonlinear dynamic system

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t), \boldsymbol{u}(t)), \boldsymbol{x} \in S, \boldsymbol{u} \in U, t \in [0, T], \tag{5}$$

provided its initial condition $\boldsymbol{x}(0) \in X$, then for any $\epsilon > 0$ and any bounded input $\boldsymbol{u}(t) \in D_U, t \in [0, T]$, there exists a natural number r and an RNN 3 with appropriate initial condition $\boldsymbol{s}_0$ such that

$$\max_{0\leq t\leq T} ||\boldsymbol{x}(t) - \boldsymbol{s}_n(t)|| < \epsilon, \tag{6}$$

where $\boldsymbol{s}_n$ is the state vector of the first n output neurons of the RNN, that is $\boldsymbol{s}_L = \begin{pmatrix} s_n \\ h \end{pmatrix} \in \mathbb{R}^L, h \in \mathbb{R}^{L-n}$.

According to Lemma 1 and 2, when $F(\boldsymbol{x}(t), \boldsymbol{u}(t))$ has a certain special structure, RNNs with a simpler structures can be used to approximate its output track.

Corollary 1: Let S be an open set in $\mathbb{R}^n$, X be a compact set in S, $F : S \times \mathbb{R} \rightarrow \mathbb{R}^n$ is a vector function with a continuous first-order derivative function, consider the nonlinear dynamic system

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t), t), \boldsymbol{x} \in S, t \in [0, T], \tag{7}$$

provided its initial condition $\boldsymbol{x}(0) \in X$, then for any $\epsilon > 0$, there exists a natural number r and an RNN of the following form with appropriate initial condition $\boldsymbol{s_0}$:

$$\frac{\mathrm{d}\boldsymbol{s}(t)}{\mathrm{d}t} = -\alpha \boldsymbol{s} + W_1 \sigma(\boldsymbol{s}), \boldsymbol{s} \in \mathbb{R}^L, \tag{8}$$

such that

$$\max_{0 \leq t \leq T} ||\boldsymbol{x} - \boldsymbol{s_n}|| < \epsilon, \tag{9}$$

where $\boldsymbol{s} = \begin{pmatrix} \boldsymbol{s_n} \\ h \end{pmatrix} \in \mathbb{R}^L, \boldsymbol{s_n} \in \mathbb{R}^n, h \in \mathbb{R}^r$.

The time-dependent PDE is also a continuous linear or nonlinear complex dynamical system, and its forward problem can be modeled by the continuous RNN model in the sequence-to-sequence task. Therefore, we proposed a Seq2Seq-based stacked GRU network model to approximate the solution of PDEs, which is both a meshless parameter learning model and captures the correlation features between the timing of input sequences of PDEs, thus enriching the feature expressiveness of the model's hidden variables and allowing the model to be efficiently generalized to solve PDEs of various physical problems.

B. SEQ2SEQ-STACKED GRU ARCHITECTURE FOR PDES

Consider a forward problem of a time-dependent nonlinear PDE with a spatial dimension of d as follows:

$$\begin{aligned}
&\frac{\partial^a u}{\partial t^a}(t, \boldsymbol{x}) - \mathcal{N} u(t, \boldsymbol{x}) = 0, (t, \boldsymbol{x}) \in (0, T) \times \Omega \\
&u(t = 0, \boldsymbol{x}) = \phi(\boldsymbol{x}) \\
&u_t(t = 0, \boldsymbol{x}) = \varphi(\boldsymbol{x}), \boldsymbol{x} \in \Omega \\
&u(t, \boldsymbol{x}) = \omega(t, \boldsymbol{x}) \\
&u_x(t, \boldsymbol{x}) = \upsilon(t, \boldsymbol{x}), (t, \boldsymbol{x}) \in [0, T] \times \partial\Omega,
\end{aligned} \tag{10}$$

where $a \in \{1, 2\}$ is the first-order or second-order PDEs under the time dimension; $t \in \mathbb{R}^1$ is the time dimension, $\boldsymbol{x} \in \mathbb{R}^d$ is the d-dimensional space dimension; $u(t, \boldsymbol{x})$ is the real analytical solution of PDEs (it is unknown), Ω is the internal value range of $\boldsymbol{x}$, $\partial\Omega$ is the boundary value range of $\boldsymbol{x}$, and $\mathcal{N}$ is a linear or nonlinear operator. The first formula represents a differential operator equation, which can be a hyperbolic, parabolic, or elliptic equation. The second and third formulas represent the initialization conditions. Finally, the fourth and fifth formulas represent the boundary conditions, which can be a combination of Dirichlet or Neumann boundary conditions. To solve the forward problem for PDEs more effectively, we constructed the network model depicted in Figure 1 based on the relevant lemmas presented in Section II-A. Assuming that the PDEs has S conditional constraints, a batch of the model training contains S groups of sequences, and each group of sequences contains N samples, corresponding to sample sets that satisfy different conditional constraints. These sequence samples are randomly generated by a random sampling algorithm (such as Stratified [35] or Latin Hypercube Sampling [36]). The trick employs independent data resampling as a semi-supervised training set D^{train} for each round of model learning, thereby preventing grid formation.

$$\begin{aligned}
D^d &= \{(t_d, x_d) \in (0, T) \times \Omega\} \sim p_d \\
D^i &= \{(t_i, x_i) \in [0] \times \Omega\} \sim p_i \\
D^b &= \{(t_b, x_b) \in [0, T] \times \partial\Omega\} \sim p_b \\
D^{train} &= \{D^d, D^i, D^b\},
\end{aligned} \tag{11}$$

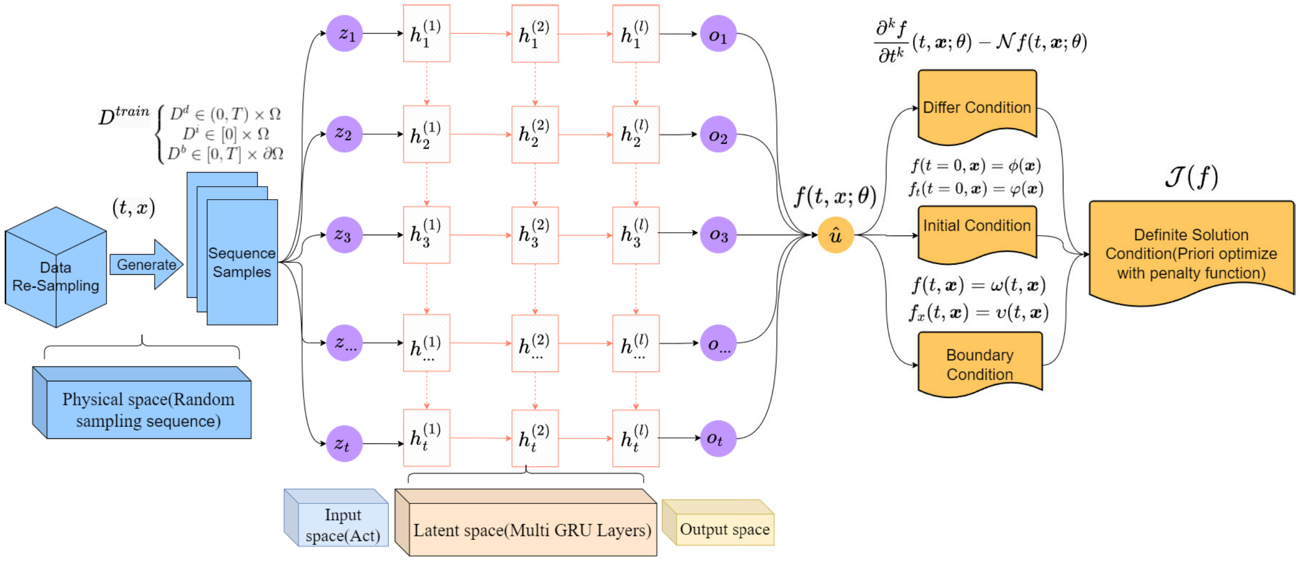
where D^d is the sample set sampled on the differential operator equation, D^i is the sample set sampled on the initial condition, D^b is the sample set sampled on the boundary condition, t_* and x_* denote the temporal and non-temporal features sampled in each sample set, respectively, and p_* is the probability density function in the sampling method. Unlike PINN [15], [37], the input sequences to our algorithm do not require any experimental data as loss terms for the model, except for sampling out sequences (D^{train}) that satisfy physical constraints.

As $u(t, x)$ is unknown in the differential operator equation, D^d is equivalent to unsupervised learning throughout the entire training procedure. Moreover, because the functions $\phi(\boldsymbol{x}), \varphi(\boldsymbol{x}), \omega(t, \boldsymbol{x}), \upsilon(t, \boldsymbol{x})$ are known, D^i and D^b can be utilized as part of supervised learning to correct model learning errors. Thus, the entire training process is equivalent to semi-supervised learning.

The hidden layer adopts a synchronous Seq2Seq stacking structure and recursively learns the correlation of different sequences in the time dimension through the hidden state $h_t^{(l)}$ ($h_t^{(l)}$ shares historical information in the time dimension) of the lth layer. We used a GRU [38] to update the hidden state of the input sequence simultaneously by updating gate $z_t^{(l)}$ and resetting gate $r_t^{(l)}$:

$$\begin{aligned}
z_t^{(l)} &= \sigma(W_z^{(l)} x_t + U_z^{(l)} h_{t-1} + b_z^{(l)}), z_t^{(l)} \in \{0, 1\} \\
r_t^{(l)} &= \sigma(W_r^{(l)} x_t + U_r^{(l)} h_{t-1} + b_r^{(l)}), r_t^{(l)} \in \{0, 1\} \\
\widetilde{h}_t^{(l)} &= \tanh(W_h^{(l)} x_t + U_h^{(l)} (r_t^{(l)} \odot h_{t-1}^{(l)}) + b_h^{(l)}) \\
h_t^{(l)} &= z_t^{(l)} \odot h_{t-1}^{(l)} + (1 - z_t^{(l)}) \odot \widetilde{h}_t^{(l)},
\end{aligned} \tag{12}$$

where $W_*^{(l)}$, $U_*^{(l)}$, and $b_*^{(l)}$ are learnable parameters, $W_*^{(l)} \in \mathbb{R}^{r \times h}$ is the state-input weight matrix, $U_*^{(l)} \in \mathbb{R}^{h \times h}$ is the state-state weight matrix, $b_*^{(l)} \in \mathbb{R}^{1 \times h}$ is the bias vector, and $\widetilde{h}_t^{(l)}$ is the candidate state at the current moment t, the operator $\odot$ denotes the Hadamard product operator for vectors, which multiplies each element. Except for the output layer, each layer of neurons in the model employed the same activation function to finish the nonlinear mapping, such as the GELU function [39]. The algorithmic procedure is described in the subsequent subsection.



D^{train} $D^d \in (0,T)\times\Omega$, $D^i \in [0]\times\Omega$, $D^b \in [0,T]\times\partial\Omega$
(t,x)
Data Re-Sampling
Generate
Sequence Samples
Physical space(Random sampling sequence)
Input space(Act)
Latent space(Multi GRU Layers)
Output space
$f(t,x;\theta)$
$\hat{u}$
$\frac{\partial^k f}{\partial t^k}(t,\boldsymbol{x};\theta) - \mathcal{N}f(t,\boldsymbol{x};\theta)$
Differ Condition
$f(t=0,\boldsymbol{x})=\phi(\boldsymbol{x})$
$f_t(t=0,\boldsymbol{x})=\varphi(\boldsymbol{x})$
Initial Condition
$f(t,\boldsymbol{x})=\omega(t,\boldsymbol{x})$
$f_x(t,\boldsymbol{x})=v(t,\boldsymbol{x})$
Boundary Condition
$\mathcal{J}(f)$
Definite Solution Condition(Priori optimize with penalty function)

FIGURE 1. Seq2Seq stacked GRU model architecture: (a) Generate a collection of sequences in physical space that satisfy the corresponding constraints by independent resampling of the data. (b) In the input space, a multilayer perceptron with a nonlinear activation function maps the original input into the hidden space. (c) A multi-layer gated neural network mines the correlations between sequences in the hidden space, which in turn enriches the representation of sequences. (d) Finally, the projection operation on the output sequence is accomplished in the output space by a multilayer perceptron without an activation function. (e) The updating of the model parameters is done by optimising the loss function embedded in the physical prior knowledge.

C. ALGORITHM PROCESS

Assuming that the solution $u(t,\boldsymbol{x})$ of the forward problem is Lipschitz continuous and bounded, we approximated u with the algorithm model f established in Section II-B, i.e., if $u \in L^p[\Omega_T], 1 \leq p \leq \infty$, then there must exist $f^* = f(t,\boldsymbol{x},\theta^*) \in \mathcal{F}$, such that $||u - f^*||_p = \inf_{f^{(n)}\in\mathcal{F}} ||u - f^{(n)}||_p$, where f and u both belong to the linearly normed finite space, θ is the parameter vector controlling f, $||f||_p = (\int_{\Omega_T} |f(t,\boldsymbol{x};\theta)|^p \mathrm{d}x)^{\frac{1}{p}}$, and different parameters θ form a function cluster $\mathcal{F} = \{f(t,\boldsymbol{x};\theta^{(1)}), f(t,\boldsymbol{x};\theta^{(2)}), \ldots, f(t,\boldsymbol{x};\theta^{(n)})\}$, each f in $\mathcal{F}$ has a smooth, continuous, and differentiable expression, and its partial derivatives of each order can be derived using the automatic differentiation method [34]. The numerical solution of a set of forward problems was calculated using the forward process, and then the conditional residual($\mathcal{J}_*(f)$) of PDEs was used as the prior knowledge or penalty function of the model loss function $\mathcal{J}(f)$ to reverse optimize the model parameters using the learning rate decay algorithm Adam with momentum [41]. The learning of model parameters is a non-convex optimization problem, and it is typically challenging to determine the global optimal solution; therefore, we simultaneously introduced randomness into the sampling process and gradient calculation such that the algorithm could escape the saddle point to the best extent possible. The flow is shown in Algorithm 1, where $\mathcal{J}_d(f)$ is the differential operator residual of PDEs, $\mathcal{J}_i(f)$ and $\mathcal{J}_b(f)$ are the residuals of the PDEs constraint, the hyperparameter λ_* is the penalty factor for each residual term, $\jmath$ is a differentiable function that quantifies the residual(such as MSE or Log-Cosh function), $\nabla_\theta \mathcal{J}(f)$ is the gradient of the current f to θ, α is the learning rate with momentum decay, and the f^* trained by $\mathcal{J}(f)$ can maximally satisfy the differential operator, boundary conditions, and initial condition constraints of PDEs. We verified the convergence of $\mathcal{J}(f)$ in both the numerical fitting experiments in Section III. In addition, we enrich the property of having consistent approximation by embedding the low-dimensional continuous dynamical system into a continuous feedback neural network using a constructive approach in the appendix.

III. NUMERICAL EXAMPLES

In this section, we consider a series of forward problems in classical mathematical physics to evaluate our algorithm. We modeled and numerically solved these forward problems using our algorithm to validate the accuracy and generalization capability of the algorithm as well as its ability to capture actual physical phenomena. In Section III-A, we presents five one-dimensional forward problems to validate the accuracy of the algorithm; in Section III-B, three two-dimensional forward problems are presented to demonstrate the generality of the algorithm.

Within each experiment in this Section, the evaluation object of the algorithm is the exact mathematical solution or the conventional numerical solution on the test set. Following the flow of Algorithm 1, for the residual function $\jmath$ of the constrained conditions, we uniformly selected MSE [42] and used the Adam optimizer [41] to optimize the loss function(i.e., $\theta^* = \arg\min_{\theta} \mathcal{J}(f)$). Moreover, the activation function selected was GELU [39]. Regarding the data Re-Sampling technique, we used Latin hypercube sampling [36] to generate a set of training sequences that satisfied the corresponding constraints. The learning rate is 0.0001, the number of samples is 5000, the sample size of each batch

Algorithm 1 Overview of the Method

```
Data: Number of samples $N$ sampled for each sequence, sample size $n$ for each batch, number of layers of GRU $L$,
      number of neurons $N_{neurons}$ for each layer, input dimension $r$ and output dimension $s$ of the sample, total number
      of training sessions of the model $N_{epoch}$, the hyperparameter learning rate $\alpha$, $\lambda_1, \lambda_2, \lambda_3$ is employed to balance
      the loss term, acceptable error accuracy $\epsilon$;
Result: optimal model $f^*$;
Initialize: $f \leftarrow$ MLP, MultilayerGRU $\leftarrow \theta = \{W_*, U_*, b_*\} \leftarrow$ Xavier initialization [40];
while $epoch \leq N_{epoch}$ do
    $X^d, X^i, X^b \leftarrow$ The set of sequences satisfying $(0, T] \times \Omega$, $\Omega$, $(0, T] \times \partial\Omega$ corresponding constraints is generated as
      the training set by the data resampling method, where $X^d$, $X^i$ and $X^b$ denote the set of sequences on the interior,
      initial and boundary conditions of the PDE, respectively;
    $D^{train}, D^{val} \leftarrow$ Divide $X^d, X^i, X^b$ into batches, where each batch contains $S$ group sequences and each sequence
      contains $n$ samples. The $D^{train}$ denotes the training set of sequences providing model training and $D^{val}$ denotes the
      validation set of sequences providing model validation.;
    for $\{D^d, D^i, D^b\} \leftarrow D^{train}$ do
        $\mathcal{J}_d(f) \leftarrow \frac{1}{|D^d|} \sum_{(t_d, x_d) \in D^d} \jmath(\frac{\partial^k f}{\partial t^k}(t_d, x_d; \theta), \mathcal{N}f(t_d, x_d; \theta))$;
        $\mathcal{J}_i(f) \leftarrow \frac{1}{|D^i|}[\sum_{(t_i, x_i) \in D^i} \jmath(f(t_i, x_i; \theta), \phi(x_i)) + \jmath(f_t(t_i, x_i; \theta), \varphi(x_i))]$;
        $\mathcal{J}_b(f) \leftarrow \frac{1}{|D^b|}[\sum_{(t_b, x_b) \in D^b} \jmath(f(t_b, x_b; \theta), \omega(t_b, x_b)) + \jmath(f_x(t_b, x_b; \theta), \upsilon(t_b, x_b))]$;
        $\mathcal{J}(f) \leftarrow \lambda_1 \mathcal{J}_d(f) + \lambda_2 \mathcal{J}_b(f) + \lambda_3 \mathcal{J}_i(f)$;
        $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{J}(f)$;
    Evaluate the error of the current model $f$ on the $D^{val}$ to get the validation error $err_{val}$;
    if $err_{val} \leq \epsilon$ then
        break;
Return $f^*$
```

is 1024, the number of training times is 10000, the number of layers of the GRU is 4, and the number of neurons is 50. The NVIDIA RTX 3060 GPU card was utilised in each of the following experiments.

A. ONE-DIMENSIONAL FORWARD PROBLEM

In this subsection, we test the accuracy and flexibility of the algorithm in five forward problems: the heat conduction, wave, Burgers, Schrodinger, and diffusion equations.

1) ONE-DIMENSIONAL HEAT CONDUCTION EQUATION

First, we evaluated the capacity of the algorithm to solve first-order linear PDEs. Heat conduction is a common physical phenomenon wherein, in case of an uneven temperature distribution within an object, heat spontaneously travels from high to low temperatures. The resolution $u(t, x)$ represents the temperature distribution at any point within an object at any instant in time. We considered the following linear heat conduction scenario for a bounded rod:

$$\frac{\partial u}{\partial t} = \frac{1}{\pi^2}\frac{\partial^2 u}{\partial x^2}, t \in (0, T), x \in (0, L)$$
$$u|_{t=0} = 1 + \cos\frac{2\pi x}{L}, x \in [0, L]$$
$$\frac{\partial u}{\partial x}|_{x=0} = 0, \frac{\partial u}{\partial x}|_{x=L} = 0, t \in (0, T). \tag{13}$$

This PDE is linear and has a parabolic form in mathematics. Its boundary conditions are Neumann boundary conditions,

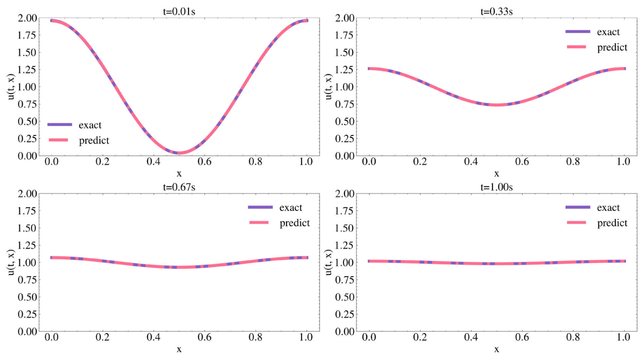


FIGURE 2. One-dimensional heat conduction equation: Compare the prediction and actual temperature of the algorithm for four time samples at $t = (0.1, 0.33, 0.66, 1)$.

implying that the two ends of the bounded rod stay adiabatic. Thus. the temperature gradient at both ends is zero. Figure 2 shows the prediction results of the algorithm on four unequal time snapshots and compares them with the mathematical analytical solution of the PDEs. Explicitly, the algorithm fit the theoretical exact value very well, and it also learned the physical fact that when $t = 1$, the entire heat conduction system gradually tended to the steady state temperature $C_0 = 1$. Figure 3 portrays a visual comparison of the predicted and actual temperature of the system across the entire space-time domain, demonstrating that the algorithm had a reasonable fitting effect across the entire space-time

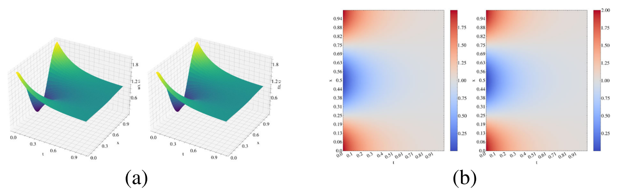


FIGURE 3. One-dimensional heat conduction equation: Intuitive comparison of algorithm-predicted and actual temperature changes over the entire space-time region. (a) On the image of the function. (b) On the heat map.

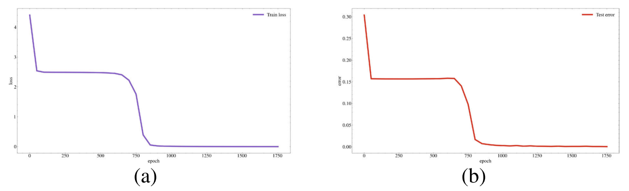


FIGURE 4. One-dimensional heat conduction equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

domain. The training process of the model observation in Figure 4. Obviously, as the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, while the error calculated by the model on the test set steadily diminished. Moreover, it exhibited high accuracy for the prediction accuracy of the first-order linear PDEs.

2) ONE-DIMENSIONAL STRING VIBRATION EQUATION

Next, we evaluated the capacity of the algorithm to solve the forward problem for second-order linear PDEs. The wave equation describes the time-dependent evolution of the wave function. Consider a string of length L placed horizontally, with the horizontal direction of the string as the x-axis. We used the function $u(t, x)$ to represent the amplitude of any position $x(0 < x < L)$ on the string at any time t:

$$\frac{\partial^2 u}{\partial t^2} = \frac{1}{\pi^2}\frac{\partial^2 u}{\partial x^2}, t \in (0, T), x \in (0, L)$$
$$u|_{t=0} = \cos\frac{\pi x}{2L}, \frac{\partial u}{\partial t}|_{t=0} = 0, x \in [0, L]$$
$$\frac{\partial u}{\partial x}|_{x=0} = 0, u|_{x=L} = 0, t \in (0, T). \tag{14}$$

The PDE is also linear and mathematically has a hyperbolic shape. Here, $u|_{t=0}$ and $\frac{\partial u}{\partial t}|_{t=0}$ in the constraints are the initial displacement and initial velocity, respectively; thus, the boundary constraints for this PDE are a combination of Neumann boundary conditions and Dirichlet boundary conditions. Figure 5 shows the prediction results of the algorithm on four unequal time snapshots and compares them with the mathematical analytical solution of the PDEs. It is obvious that the algorithm fit the theoretical exact value very well. Moreover, the algorithm discovered that the string was fixed at the $x = L$ end and vibrated simply between $[-1, 1]$ at the $x = 0$ end. The visual comparison between the predicted and actual amplitudes of the system in the entire

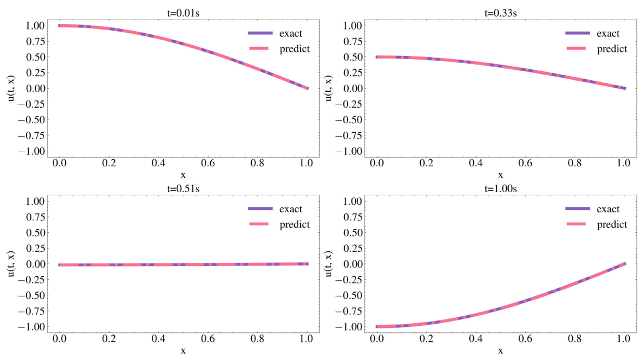


FIGURE 5. One-dimensional string vibration equation: Compare the prediction and actual amplitude of the algorithm for four time samples at t = (0.1, 0.33, 0.66, 1).

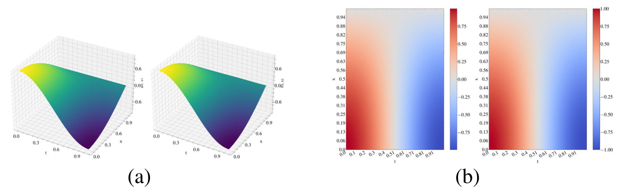


FIGURE 6. One-dimensional string vibration equation: Intuitive comparison of algorithm-predicted and actual amplitude changes over the entire space-time region. (a) On the image of the function. (b) On the heat map.

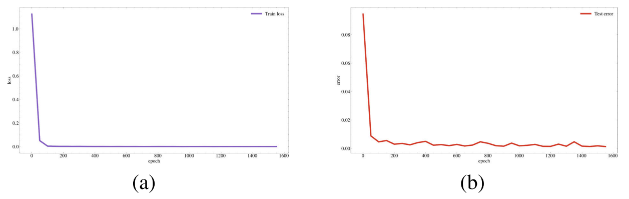


FIGURE 7. One-dimensional string vibration equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

space-time domain, demonstrating that the algorithm exerted a reasonable fitting effect on the entire space-time domain is shown in Figure 6. Figure 7 shows the training process of the model. With the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished. Moreover, it exhibited high accuracy for the prediction accuracy of the second-order linear PDEs.

3) ONE-DIMENSIONAL BURGERS EQUATION

Next, we tested the ability of the algorithm to solve the forward problem for first-order nonlinear PDEs. The Burgers equation is a nonlinear PDE that models shock wave propagation and reflection. As the strong shock wave in this equation is difficult to deal with using traditional numerical methods, we considered the following scenarios:

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \frac{0.01}{\pi}\frac{\partial^2 u}{\partial x^2}, t \in (0, T), x \in (-1, 1)$$

$$u|_{t=0} = -\sin \pi x, x \in [-1, 1]$$
$$u|_{x=-1} = 0, u|_{x=1} = 0, t \in (0, T). \tag{15}$$

This is a nonlinear PDE with Dirichlet boundary conditions.

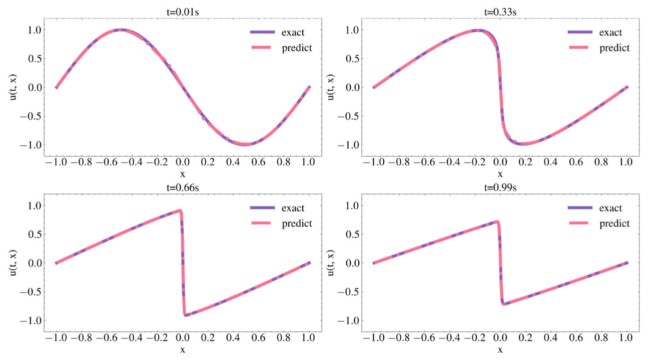


FIGURE 8. **One-dimensional Burgers equation: Compare the prediction and actual shock of the algorithm for four time samples at $t = (0.1, 0.33, 0.66, 1)$.**

The prediction results of the algorithm on four unequal time snapshots and compares them with the traditional numerical solution of the PDEs is shown in Figure 8. Explicitly, the algorithm fit the numerical reference value very well, and it also learned the changes of strong shock waves.

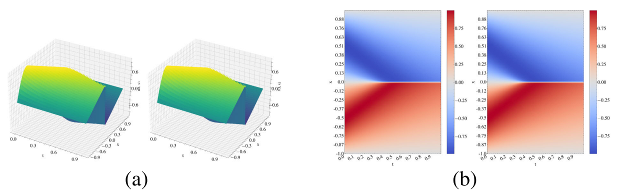


FIGURE 9. **One-dimensional Burgers equation: Intuitive comparison of algorithm-predicted and actual shock wave changes over the entire space-time region. (a) On the image of the function. (b) On the heat map.**

As is illustrated in Figure 9, a visual comparison between the predicted and actual shock wave of the system in the entire space-time domain demonstrates that the algorithm exhibited a reasonable fitting effect within the entire space-time domain. Figure 10 shows the training process of the model. Obviously, as the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished, Moreover, it exhibited high accuracy for the prediction accuracy of the first-order nonlinear PDEs.

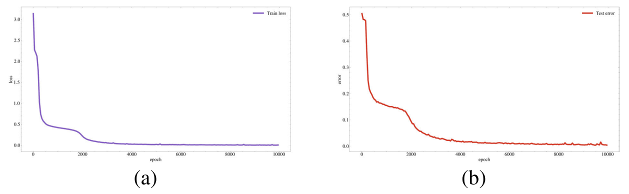


FIGURE 10. **One-dimensional Burgers equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.**

4) ONE-DIMENSIONAL SCHRODINGER EQUATION

Next, we tested the ability of the algorithm to solve the forward problem for complex-valued first-order nonlinear PDEs. Following the uncertainty principle, the momentum and position of a microscopic particle are described by the wave function $\psi(t, x)$, which satisfies the complex solution of the Schrodinger equation. The statistical interpretation of the wave function reveals that for a large number of particles, the mode $\psi\psi^*$ of the wave function represents the particle density in space, whereas for a single particle, $\psi\psi^*$ represents the probability of the particle appearing at a particular location in space. We considered the following scenarios:

$$\mathrm{i}\frac{\partial \psi}{\partial t} = -\frac{1}{2}\frac{\partial^2 \psi}{\partial x^2} - |\psi|^2\psi, t \in (0, \frac{\pi}{2}), x \in (-5, 5)$$
$$\psi|_{t=0} = \frac{2}{\exp(x) + exp(-x)}, x \in [-5, 5]$$
$$\psi|_{x=-5} = 0, \psi|_{x=5} = 0, t \in (0, \frac{\pi}{2})$$
$$\frac{\partial \psi}{\partial x}|_{x=-5} = 0, \frac{\partial \psi}{\partial x}|_{x=5} = 0, t \in (0, \frac{\pi}{2}), \tag{16}$$

where i is the imaginary unit, the wave function $\psi(t, x) = u(x, t) + iv(x, t)$ is a complex solution, $u(x, t)$ is the real part solution, and $v(x, t)$ is the imaginary part solution. When separating the real and imaginary components, the following PDE equations were obtained. Its constraints acted on both the real and imaginary components, and its boundary constraints were a combination of Neumann and Dirichlet boundary conditions. Therefore, the output of the model had two dimensions: the real part function $u(x, t)$ and the imaginary part function $v(x, t)$ respectively:

$$\frac{\partial u}{\partial t} = -\frac{1}{2}\frac{\partial^2 v}{\partial x^2} - (u^2 + v^2)v, t \in (0, \frac{\pi}{2}), x \in (-5, 5)$$
$$\frac{\partial v}{\partial t} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + (u^2 + v^2)u$$
$$u|_{t=0} = \frac{2}{\exp(x) + exp(-x)}, x \in [-5, 5]$$
$$v|_{t=0} = 0$$
$$u|_{x=-5} = 0, u|_{x=5} = 0, t \in (0, \frac{\pi}{2})$$
$$v|_{x=-5} = 0, v|_{x=5} = 0$$
$$\frac{\partial u}{\partial x}|_{x=-5} = 0, \frac{\partial u}{\partial x}|_{x=5} = 0, t \in (0, \frac{\pi}{2})$$
$$\frac{\partial v}{\partial x}|_{x=-5} = 0, \frac{\partial v}{\partial x}|_{x=5} = 0. \tag{17}$$

Figure 11 shows the prediction results of the algorithm on four unequal time snapshots, where the ordinate is the modulus length of the real and imaginary parts of the algorithm output($|\psi(x, t)| = \sqrt{u(x, t)^2 + v(x, t)^2}$), and compares it with the traditional numerical solution of the PDEs. Obviously, the algorithm fit the numerical reference value very well and also reflects the process of particles collapsing from a superposition state to an intrinsic state. Figure 12 shows a visual comparison of the predicted and

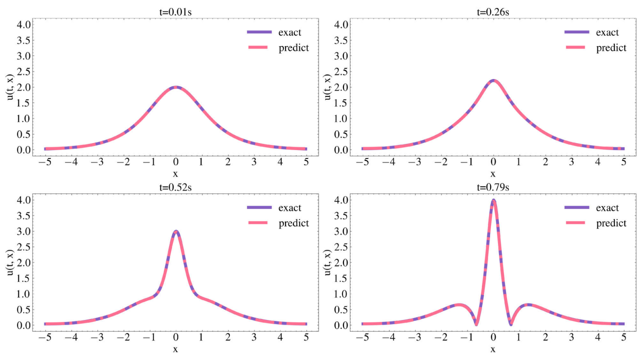


FIGURE 11. One-dimensional Schrodinger equation: Compare the prediction and actual probability value of the algorithm for four time samples at $t = (0.008, 0.259, 0.518, 0.785)$.

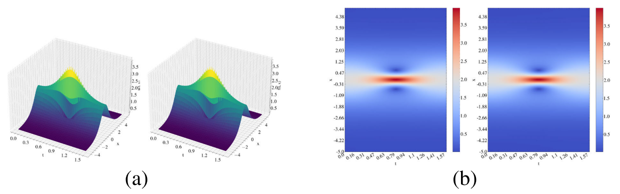


FIGURE 12. One-dimensional Schrodinger equation: Intuitive comparison of algorithm-predicted and actual probability value over the entire space-time region. (a) On the image of the function. (b) On the heat map.

actual probability change of the system across the entire space-time domain, demonstrating that the algorithm had a reasonable fitting effect across the entire space-time domain. The training process of the model is shown in Figure 13.

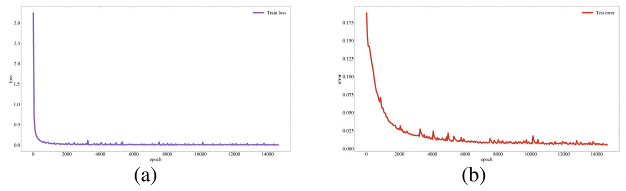


FIGURE 13. One-dimensional Schrodinger equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

Clearly, with the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished. Complex-valued solutions to first-order nonlinear PDEs were predicted with high precision using this method.

5) ONE-DIMENSIONAL DIFFUSION EQUATION

Next, we tested the ability of the algorithm to solve PDEs in the field of fluid mechanics using the diffusion equation as an example. This equation, which combines mass conservation and Fick's law, demonstrates that the first derivative of concentration with respect to time is proportional to its second derivative with respect to space. Moreover, it implies that when the difference between a point in the field and the average concentration of surrounding points is greater, the speed of concentration diffusion is also faster. We considered the following scenarios:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} - e^{-t}(\sin(\pi x) - \pi^2 \sin(\pi x))\\ u|_{t=0} &= \sin(\pi x), x \in [-1, 1]\\ u|_{x=-1} &= 0, u|_{x=1} = 0, t \in (0, 1).\end{aligned} \tag{18}$$

This is also a nonlinear PDE with Dirichlet boundary conditions. Figure 14 shows the prediction results of the algorithm on four unequal time snapshots and compares them with the mathematical analytical solution of the PDEs. It is obvious that the algorithm fit the theoretical exact value very well. The visual comparison of the predicted and actual diffusion of the system in the entire space-time domain is shown in Figure 15. Obviously, the algorithm learned the diffusion process of the system over time. Figure 16 shows the training process of the model. As the model continued

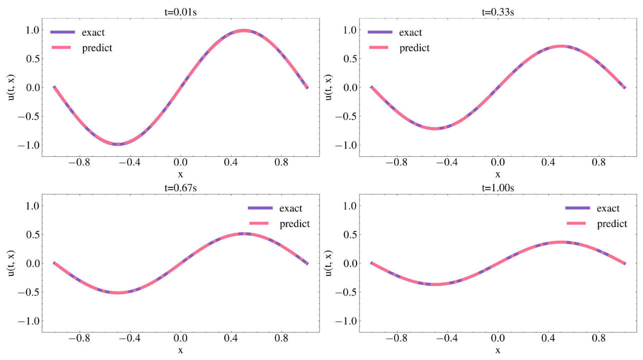


FIGURE 14. One-dimensional diffusion equation: Compare the prediction and actual diffusion of the algorithm for four time samples at $t = (0.1, 0.33, 0.66, 1)$.

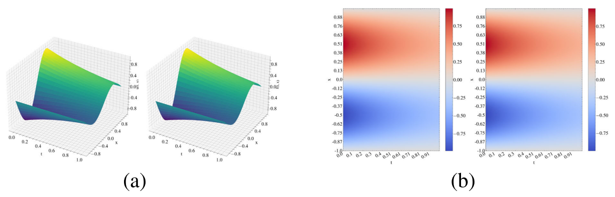


FIGURE 15. One-dimensional diffusion equation: Intuitive comparison of algorithm-predicted and actual diffusion changes over the entire space-time region. (a) On the image of the function. (b) On the heat map.

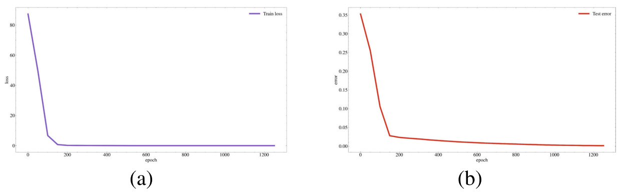


FIGURE 16. One-dimensional diffusion equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished. Moreover, it exhibited good predictive accuracy for the diffusion process in fluid mechanics.

B. TWO-DIMENSIONAL FORWARD PROBLEM

In this subsection, we apply the algorithm to forward problems in three two-dimensional spaces (the heat conduction, wave, and Kovasznay flow equations) to demonstrate the generalizability of the algorithm from one-dimensional space to two-dimensional space.

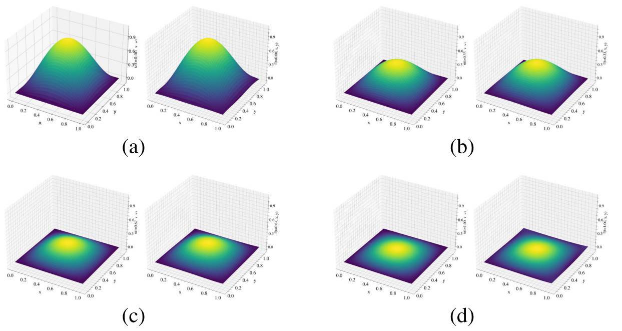


FIGURE 17. Two-dimensional heat conduction equation: (a)-(d) Compare the prediction and actual temperature of the algorithm for four time samples at $t = (0.1, 0.33, 0.66, 1)$.

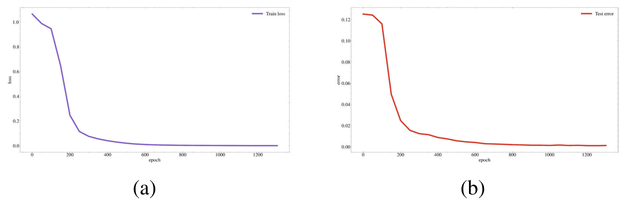


FIGURE 18. Two-dimensional heat conduction equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

1) TWO-DIMENSIONAL HEAT CONDUCTION EQUATION

We considered the following heat conduction scenario in two-dimensions:

$$\frac{\partial u}{\partial t} = \frac{1}{\pi^2}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), t \in (0,1), x \in (0,1),$$
$$y \in (0,1)$$
$$u|_{t=0} = \sin(\pi x)\sin(\pi y), x \in [0,1], y \in [0,1]$$
$$u|_{x=0} = 0, u|_{x=1} = 0, t \in (0,1), y \in [0,1]$$
$$u|_{y=0} = 0, u|_{y=1} = 0, t \in (0,1), x \in [0,1]. \quad (19)$$

This is a first-order two-dimensional linear PDE with Dirichlet boundary conditions; the sample of the input sequence is time t and two-dimensional space $\boldsymbol{x} = (x, y)$; Figure 17 shows an intuitive comparison between the predicted temperature (right) and the actual temperature (left) of the algorithm on four unequal time snapshots, demonstrating that the algorithm could still cover the exact value in two-dimensions; As predicted by the algorithm. The training process of the model is shown in Figure 18. Obviously, with the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished. Moreover, it exhibited high predictive accuracy for the first-order linear PDEs in two-dimensions. Figure 19 presents

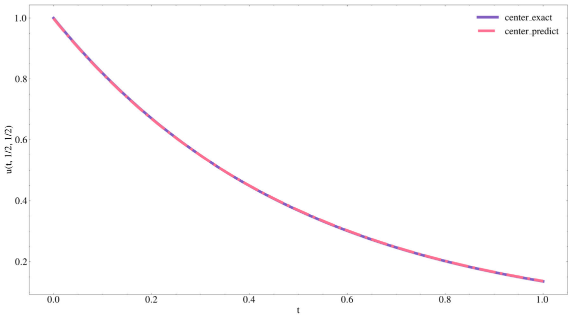


FIGURE 19. Two-dimensional heat conduction equation: Compare the predicted and actual cooling processes of the algorithm at the centre point $(x, y) = (\frac{1}{2}, \frac{1}{2})$.

the cooling process over time at the centre of the film, which corresponded to the theoretical cooling process.

2) TWO-DIMENSIONAL WAVE EQUATION

We considered the following fluctuation scenario in two-dimensional:

$$\frac{\partial^2 u}{\partial t^2} = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), t \in (0,1), x \in (0,1),$$
$$y \in (0,1)$$
$$u|_{t=0} = \sin(\pi x)\sin(\pi y), x \in [0,1], y \in [0,1]$$
$$\frac{\partial u}{\partial t}|_{t=0} = 0, x \in [0,1], y \in [0,1]$$
$$u|_{x=0} = 0, u|_{x=1} = 0, t \in (0,1), y \in [0,1]$$
$$u|_{y=0} = 0, u|_{y=1} = 0, t \in (0,1), x \in [0,1]. \quad (20)$$

This is a two-dimensional linear PDE of the second order

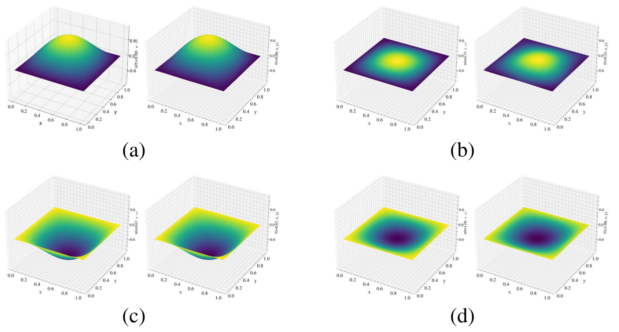


FIGURE 20. Two-dimensional wave equation: (a)-(d) Compare the prediction and actual amplitudes of the algorithm for four time samples at $t = (0.1, 0.33, 0.66, 1)$.

with Dirichlet boundary conditions. Figure 20 shows the intuitive comparison of the predicted amplitude (right) and the actual amplitude (left) of the algorithm on four unequal time snapshots. Evidently, the algorithm covered the actual vibration of the membrane in two-dimensions perfectly. Figure 21 shows the training process of the model. Obviously, with the model continued to learn, the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished. Moreover, it exhibited high predictive accuracy for the second-order linear PDEs in two-dimensions. The algorithm accurately captured the

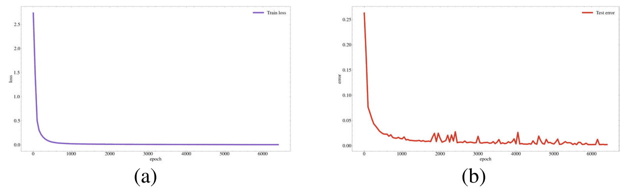


FIGURE 21. Two-dimensional wave equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

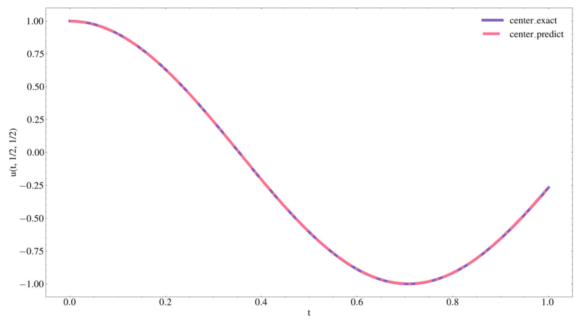


FIGURE 22. Two-dimensional wave equation: Compare the algorithm predicted amplitude and actual amplitude at the center point of $(x, y) = (\frac{1}{2}, \frac{1}{2})$ over time.

vibration of the center point $(x, y) = (\frac{1}{2}, \frac{1}{2})$ of the rectangular membrane over time, as shown in Figure 22. Moreover, the displacement was not strictly periodic, which is consistent with the theoretical vibration process.

3) TWO-DIMENSIONAL KOVASZNAY FLOW EQUATION

We considered the following Kovasznay flow equation scenario:

$$
\begin{aligned}
&u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), x \in (0, 1),\\
&\qquad y \in (0, 1)\\
&u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\\
&\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \qquad (21)
\end{aligned}
$$

where $u(x, y)$ and $v(x, y)$ are the velocity of the velocity field in the x and y directions, respectively, $p(x, y)$ is the pressure at each point in the flow field, and Re is the Reynolds number of the flow. Kovasznay flow corresponds to an exact solution of the Navier-Stokes equations and is interpreted to characterize the flow behind a two-dimensional grid. In the end, the output of the model had three dimensions: $\{\hat{u}(x, y), \hat{v}(x, y), \hat{p}(x, y)\}$, and their boundary values complied with the Dirichlet conditions. As is illustrated in Figure 23,

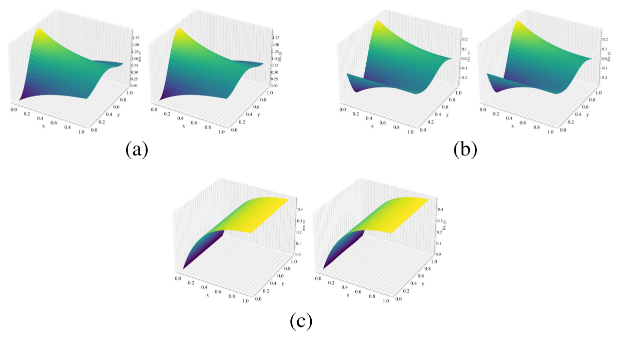


FIGURE 23. Two-dimensional Kovasznay flow equation: The comparison between the results predicted by the algorithm and the actual results in the entire space domain. (a) Velocity in the x-direction. (b) Velocity in the y-direction. (c) Pressure.

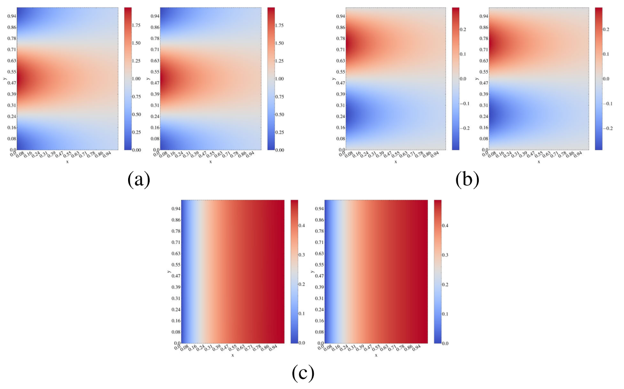


FIGURE 24. Two-dimensional Kovasznay flow equation: Comparison of heatmaps in the entire spatial domain. (a) Velocity in the x-direction. (b) Velocity in the y-direction. (c) Pressure.

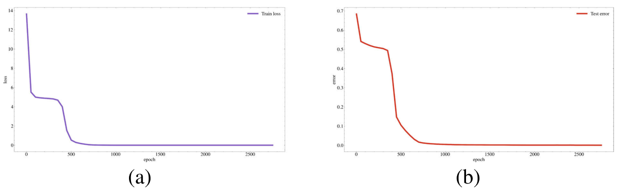


FIGURE 25. Two-dimensional Kovasznay flow equation: Evaluation results for training and test sets. (a) Loss function convergence results for the training set. (b) Model error results for the test set.

the velocity and pressure fields predicted by the model for the Kovasznay fluid. On the left, the exact results are presented, and on the right, the predicted results are shown. It is obvious that the algorithm provided a superior fit for all three physical quantities. Figure 24 shows a comparison of the velocity field and the pressure field on the heatmap, demonstrating that all physical quantities predicted by the model were close to the actual situation. Figure 25 shows the training process of the model. Obviously, as the model continued to learn,

TABLE 1. Comparison of MSE results for different PDEs numerical calculation experiments.

PDEs	PINN [15]	SSL-PINN [37]	Our Model
One-dimensional heat conduction equation	1.5×10^{-6}	1.18×10^{-6}	6.84×10^{-7}
One-dimensional string vibration equation	9.78×10^{-6}	4.72×10^{-6}	3.10×10^{-6}
One-dimensional Burgers equation	1.61×10^{-3}	5.50×10^{-4}	8.39×10^{-5}
One-dimensional Schrodinger equation	1.58×10^{-4}	9.12×10^{-5}	7.02×10^{-5}
One-dimensional diffusion equation	5.28×10^{-6}	3.68×10^{-6}	2.58×10^{-6}
Two-dimensional heat conduction equation	6.84×10^{-6}	3.05×10^{-6}	2.66×10^{-6}
Two-dimensional wave equation	1.82×10^{-5}	7.12×10^{-6}	6.18×10^{-6}
Two-dimensional Kovasznay flow equation	5.28×10^{-6}	3.68×10^{-6}	1.70×10^{-7}

the $\mathcal{J}(f)$ on the training set continued to converge, and the error calculated by the model on the test set steadily diminished.

IV. DISCUSSION

To verify the applicability and generalization of the proposed method, some comparative methods based on deep learning to solve PDEs are added in Table 1. The evaluation metric we chose is $MSE = \frac{1}{|D^{test}|}\sum_{i=1}^{|D^{test}|}(y_i - \hat{y}_i)^2$, where $y_i = u(t_i, x_i)$ is the exact or numerical solution of the problem on the test set, and $\hat{y}_i = f(t_i, x_i; \theta)$ is the prediction result of our model on the test set, and $|D^{test}|$ is the size of the test set.

The objective of the proposed model needs to be both to accurately solve the numerical solutions of specific PDEs and to be applicable to PDEs of different types or complexity in various physical problems. As demonstrated in Table 1, the proposed model has less error in all numerical inference results than other methods, and it can be effectively generalized to different types of PDEs (e.g., elliptic, parabolic, or parabolic-shaped PDEs, or PDEs with multi-output physical quantities), which demonstrates the superiority and feasibility of the proposed model. From the experimental results, it is observed that the error of the numerical inference results based on the multilayer feed-forward neural network is higher than the proposed model, which is mainly because PDEs with temporal evolution can have strong temporal order, and the feedforward neural network does not consider the correlation character between samples when extracting the hidden variable features of the input sequences, thereby leading to the lack of correlation features of the hidden variable sequences in the hidden layer. This additionally validates the ability of the stacked GRU structural model to capture the temporal correlation of the input sequences, which compensates for the lack of feature expressiveness of previous deep learning methods based on multi-layer feed-forward neural networks for solving PDEs. Moreover, unlike the numerical computation method based on iteration, the proposed model is based on a feedback neural network to approximate the solution of the PDE, which is a semi-supervised deep learning method based on parameter learning. After the proposed model has been trained, the numerical results can be quickly inferred for any tested sample set simply by the forward computation process of the model, which does not require the iterative process of gridding. The proposed algorithm is a deep learning model based on stacked GRU, where the complexity of the model is proportional to the number of parameters. Although the GRU model involves two control gates and one candidate hidden state, the number of parameters is 25 percent less than the LSTM model [16]. The number of parameters involved in the experiments in this paper is about 40,000. The ADAM algorithm [41] is employed to optimize the loss function to a minimum to obtain the optimal parameters of the model. The model training takes about 3.5 to 5 hours, and the inference time of the trained model on the test set is about 1.9 to 2.8 seconds. The GPU-based asynchronous execution mechanism is beneficial to improving the computational efficiency of the model; therefore, we recommend that model training and inference be performed on GPUs.

In addition, we found some potential problems in the course of the experiment. First, the proposed model resamples the input sequences in each round of training. This step reduces the dilemma of the model falling into the saddle point in the nonconvex optimization process, but it also aggravates the training burden of the model. Therefore, we consider subsequent local resampling for the sequence where the penalty term that causes the error of the loss function to become larger after each round of training is completed. Second, the training of the model could be interrupted due to some objective factors. For this reason, we consider subsequently porting the proposed model to a parameter server for training [43]. The architecture is to distribute the training of the neural network over multiple GPU nodes to speed up the training, and each node receives the updated model from the parameter server asynchronously before computing the gradient of the parameters.

V. CONCLUSION

This study proposed an optimization algorithm based on synchronous sequence-to-sequence stacked GRU networks for characterizing and approximating the forward problem of PDEs. The constraints of the PDEs were utilized as a priori knowledge of the loss function of the network to constrain the direction of the optimization or as a penalty function for partially regularized terms to prevent optimization overfitting. The proposed model was not grid-based, and its input was a vector of randomly sampled space-time sequences that satisfied the corresponding constraints. The GRU structure of the hidden layer was used to capture and learn the correlation of timing between different input sequences. Following the training of the model, it only required its forward process to swiftly deduce the expected output physical quantity in the corresponding physical scene. Our algorithm was applied to various types of mathematical physics equations.

Physics-informed machine learning [44] is the emerging paradigm used to solve PDEs. Much of the physical knowledge can be embedded into the optimisation goals of deep learning, such as symmetry and conservation. According to the research in this paper, we should follow up with in-depth exploration in the following two aspects: On the one hand, to enhance the representation ability of the model to sequence data, such as capturing the time-series correlation features and non-time-series hybrid features of the sequence, respectively, and dynamically calculating the degree of their contribution to the output. On the other hand, to improve the penalty function of multi-objective optimization, such as by combining the Lagrange multiplier method to optimize the penalty term. We will also investigate the use of some more general experiments to represent the performance of the optimized algorithms, such as the chemical master equation (CME). In our future work, we further consider the above tasks.

APPENDIX A PROOFS OF RELATED LEMMAS

This section lists the relevant proofs on the lemmas and corollaries presented in Section II-A. Referring to previous work [30], [31], [45], [46], we embed a low-dimensional continuous dynamical system into a continuum-type RNN using a constructive approach, thereby proving that for nonlinear continuous dynamical systems with inputs, the output trajectory in finite time can be arbitrarily approximated by a class of continuum-type RNN output neurone state vectors.

A. PROOF OF LEMMA 1

For any $t \in T$, there is:

$$\begin{aligned}|x(t)-\tilde{x}(t)| &\leq \int_{t_0}^{t} |F(x(s),u(s))-\tilde{F}(\tilde{x}(s),u(s))|ds \\ &\leq \int_{t_0}^{t} |F(x(s),u(s))-F(\tilde{x}(s),u(s))|ds \\ &\quad+\int_{t_0}^{t} |F(\tilde{x}(s),u(s))-\tilde{F}(\tilde{x}(s),u(s))|ds \\ &\leq \int_{t_0}^{t} L|x(s)-\tilde{x}(s)|ds+\int_{t_0}^{t} \epsilon ds \\ &\leq L\int_{t_0}^{t} (|x(s)-\tilde{x}(s)|+\frac{\epsilon}{L})ds,\end{aligned}$$

and thus

$$|x(s)-\tilde{x}(s)|+\frac{\epsilon}{L} \leq \frac{\epsilon}{L}+L\int_{t_0}^{t}(|x(s)-\tilde{x}(s)|+\frac{\epsilon}{L})ds,\ t\in T,$$

from Gronwall's inequality [46], it follows that:

$$|x(s)-\tilde{x}(s)|+\frac{\epsilon}{L} \leq \frac{\epsilon}{L}\exp L|t-t_0|,$$

that is

$$||x(t)-\tilde{x}(t)|| \leq \frac{\epsilon}{L}(\exp L|t-t_0|-1).$$

B. PROOF OF LEMMA 2

Since $F : S \times U \to \mathbb{R}^n$ is a vector function with continuous first-order derivatives, for any initial conditions $\boldsymbol{x}(0) \in X$, by the continuity and boundedness of the output trajectory of the nonlinear continuous system($\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t)), \boldsymbol{u}(t)$), the set D_S of points of its output trajectory on the interval $[0, T]$ is a tight subset of S.

Let d_1 be the distance between the boundaries of the tight set D_S and the open set S, and d_2 be the distance between the boundaries of the tight set D_U and the open set U. For a given $\epsilon > 0$, take η such that it satisfies $0 < \eta < \min(\epsilon, d_1, d_2)$, suppose that

$$\begin{aligned}S_\eta &= \{x \in \mathbb{R}^n | \exists x_1 \in D_S, s.t.\ ||x-x_1|| \leq \eta\},\\ U_\eta &= \{u \in \mathbb{R}^m | \exists u_1 \in D_U, s.t.\ ||u-u_1|| \leq \eta\},\end{aligned}$$

then S_η and U_η are also tight sets and $D_S \subset S_\eta \subset S, D_U \subset U_\eta \subset U$, $F(x,u)$ is Lipschitz continuous on $S_\eta \times U_\eta$ with respect to x. Let L_F be the Lipschitz constant of F on $S_\eta \times U_\eta$ with respect to x. Take ϵ_1 to satisfy

$$0 < \epsilon_1 < \frac{\eta L_F}{2(\exp L_F T-1)}, \tag{22}$$

then on the tight set $S_\eta \times U_\eta$, according to [11] and [12], for any constant α, there exist a natural number r, an r-dimensional threshold vector θ and matrices $A \in \mathbb{R}^{n\times r}$, $B_1 \in \mathbb{R}^{r\times n}$, $B_2 \in \mathbb{R}^{r\times m}$ such that

$$\max_{x\in S_\eta, u\in U_\eta} ||F(x,u)+\alpha\cdot x-A\sigma(B_1x+B_2u+\theta)|| < \epsilon_1, \tag{23}$$

where σ is a Sigmoid vector function with continuous first order derivatives. Choose α to satisfy the condition

$$||\alpha\cdot\theta|| < \frac{\eta L_{\tilde{G}}}{2(\exp L_{\tilde{G}}T-1)},\ |\alpha| < \frac{L_{\tilde{G}}}{2}, \tag{24}$$

where $L_{\tilde{G}}$ is the Lipschitz constant for the mapping $\tilde{G} : \mathbb{R}^{n+r} \times \mathbb{R}^m \to \mathbb{R}^{r+n}$. Let $x(t)$ and $\tilde{x}(t)$ be solutions of the equations

$$\frac{dx(t)}{dt} = F(x(t),u(t)) \tag{25}$$

and

$$\frac{d\tilde{x}(t)}{dt} = -\alpha\tilde{x}(t)+A\sigma(B_1\tilde{x}(t)+B_2u(t)-\theta) \tag{26}$$

respectively, they all satisfy the initial condition $x(0) = \tilde{x}(0) = x_0 \in X$. According to Lemma 1 and Equation (22), we get

$$\max_{0\le t\le T} ||x(t) - \tilde{x}(t)|| \le \frac{\epsilon_1}{L_F}(\exp L_F T - 1) < \frac{\eta}{2}. \tag{27}$$

Let $\tilde{y} = B_1\tilde{x}+\theta$, $\tilde{s} = (\tilde{x}, \tilde{y})'$, then from Equations (5) and (26), we have

$$\frac{d\tilde{s}(t)}{dt} = \tilde{G}(\tilde{s}(t), u(t)) = -\alpha\tilde{s}(t) + W_1\sigma(\tilde{s}(t) + W_2u(t)) + \alpha\theta_1, \tag{28}$$

where $W_1 = \begin{pmatrix} 0 & A \\ 0 & B_1A \end{pmatrix} \in \mathbb{R}^{(n+r)\times(n+r)}$, $W_2 = \begin{pmatrix} 0 \\ B_2 \end{pmatrix} \in \mathbb{R}^{(n+r)\times m}$, $\theta_1 = \begin{pmatrix} 0 \\ \theta \end{pmatrix} \in \mathbb{R}^{n+r}$.

The first n elements of the solution vector of Equation (28) are equivalent to the solution of Equation (5). Since $\sigma(.)$ is a Sigmoid vector function with continuous first-order derivatives, the mapping $W_1\sigma(\tilde{s} + W_2u)$ is Lipschitz continuous on $\mathbb{R}^{(n+r)\times m}$ with respect to $\tilde{s}$. Let $\frac{L_{\tilde{G}}}{2}$ be the Lipschitz constant for $W_1\sigma(\tilde{s} + W_2u)$. Moreover, from Equation (24), $\frac{L_{\tilde{G}}}{2}$ is also the Lipschitz constant for $-\alpha\tilde{s}$, so $L_{\tilde{G}}$ is the Lipschitz constant of the mapping

$$\tilde{G}(\tilde{s}, u) = -\alpha\tilde{s} + W_1\sigma(\tilde{s} + W_2u) - \alpha\theta_1 \tag{29}$$

over $\mathbb{R}^{n+r} \times \mathbb{R}^m$ with respect to $\tilde{s}$, then for the following RNN

$$\frac{ds(t)}{dt} = G(s(t), u(t)) = -\alpha s + W_1\sigma(s + W_2u), \tag{30}$$

where $s = \begin{pmatrix} s_n \\ h \end{pmatrix} \in \mathbb{R}^{n+r}$, $s_n \in \mathbb{R}^n$ is the internal state of its n output neurons, $h \in \mathbb{R}^r$ is the internal state of its r hidden neurons. By Equations (23),(24) and (25), for any $s \in \mathbb{R}^{n+r}$, there are

$$||\tilde{G}(s, u) - G(s, u)|| = ||\alpha\theta|| < \frac{\eta L_{\tilde{G}}}{2(\exp L_{\tilde{G}}T - 1)}, \tag{31}$$

therefore, let s and $\tilde{s}$ represent the respective RNN Equation (30) and dynamical system Equation (25) solutions for the input u, and let them take the same initial conditions, that is

$$s_n(0) = x(0), h(0) = B_1x(0) + \theta,$$

then from Lemma 1 and Equation (27), we have

$$\max_{0\le t\le T} ||\tilde{x} - s_n|| \le \max_{0\le t\le T} ||\tilde{s} - s|| \le \frac{\eta}{2}, \tag{32}$$

from Equations (27) and (32), we have

$$\max_{0\le t\le T} ||x - s_n|| \le \max_{0\le t\le T} ||x - \tilde{x}|| + \max_{0\le t\le T} ||\tilde{x} - s_n|| \le \frac{\eta}{2} + \frac{\eta}{2} = \eta < \epsilon. \tag{33}$$

C. PROOF OF COROLLARY 1

Let $\bar{x} = \begin{pmatrix} x \\ t \end{pmatrix} \in \mathbb{R}^{n+1}$, then the nonlinear system can be expressed as

$$\frac{d\bar{x}}{dt} = F(\bar{x}(t)), \tag{34}$$

where $F(\bar{x}(t)) = \begin{pmatrix} F(x(t), t) \\ 1 \end{pmatrix}$, and the initial condition is $\bar{x}(0) = \begin{pmatrix} x(0) \\ 0 \end{pmatrix}$. When $x(0) \in X$, $\bar{x}(0)$ belongs to a tight subset of $\mathbb{R}^{n+1}$. The system is a nonlinear continuous system without inputs. According to Lemma 2, the weight matrix $W_2 = 0$ in the RNN used to approximate Equation (34), i.e., there exists an RNN of the form Equation (30) such that

$$\max_{0\le t\le T} ||\boldsymbol{x} - \boldsymbol{s_n}|| < \epsilon. \tag{35}$$

ACKNOWLEDGMENT

The authors would like to thank the editor and the reviewers for their helpful comments and suggestions.

REFERENCES

[1] C. A. Taylor, T. J. R. Hughes, and C. K. Zarins, "Finite element modeling of blood flow in arteries," *Comput. Methods Appl. Mech. Eng.*, vol. 158, nos. 1–2, pp. 155–196, May 1998.
[2] Y. Zhang, "A finite difference method for fractional partial differential equation," *Appl. Math. Comput.*, vol. 215, no. 2, pp. 524–529, Sep. 2009.
[3] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," in *Handbook of Numerical Analysis*, vol. 7. Amsterdam, The Netherlands: Elsevier, 2000, pp. 713–1018.
[4] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 12159–12168.
[5] T. B. Brown et al., "Language models are few-shot learners," in *Proc. NIPS*, 2020, pp. 1877–1901.
[6] J. Cho, J. Lei, H. Tan, and M. Bansal, "Unifying vision-and-language tasks via text generation," in *Proc. 38th Int. Conf. Mach. Learn.*, vol. 139, Jul. 2021, pp. 1931–1942.
[7] M. H. Stone, "The generalized weierstrass approximation theorem," *Math. Mag.*, vol. 21, no. 4, p. 167, Mar. 1948.
[8] J. Descloux, "Approximations in L^p and Chebyshev approximations," *J. Soc. Ind. Appl. Math.*, vol. 11, no. 4, pp. 1017–1026, 1963.
[9] M. Unser and T. Blu, "Wavelet theory demystified," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 470–483, Feb. 2003.
[10] I. I. Sharapudinov, "Approximation of functions in by trigonometric polynomials," *Izv. Math.*, vol. 77, no. 2, p. 407, 2013.
[11] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.
[12] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.
[13] D. Haussler and M. Warmuth, "The probably approximately correct (PAC) and other learning models," in *Foundations of Knowledge Acquisition*, vol. 195. Boston, MA, USA: Springer, 1993, pp. 291–312.
[14] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," in *Proc. 18th Int. Conf. Artif. Intell. Statist.*, 2015, pp. 192–204.
[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.
[16] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.
[17] J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," *J. Comput. Phys.*, vol. 375, pp. 1339–1364, Dec. 2018.

[18] D. P. Kroese and R. Y. Rubinstein, "Monte Carlo methods," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 4, no. 1, pp. 48–58, 2012.

[19] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," *Nature Mach. Intell.*, vol. 3, no. 3, pp. 218–229, Mar. 2021.

[20] N. Kovachki, S. Lanthaler, and S. Mishra, "On universal approximation and error bounds for Fourier neural operators," *J. Mach. Learn. Res.*, vol. 22, no. 1, pp. 13237–13312, 2021.

[21] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.

[22] P. Zhi, Y. Wu, C. Qi, T. Zhu, X. Wu, and H. Wu, "Surrogate-based physics-informed neural networks for elliptic partial differential equations," *Mathematics*, vol. 11, no. 12, p. 2723, Jun. 2023.

[23] Y. Lu and G. Mei, "A deep learning approach for predicting two-dimensional soil consolidation using physics-informed neural networks (PINN)," *Mathematics*, vol. 10, no. 16, p. 2949, Aug. 2022.

[24] L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," *J. Comput. Phys.*, vol. 425, Jan. 2021, Art. no. 109913.

[25] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, Jan. 2021.

[26] J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 34, pp. 8505–8510, Aug. 2018.

[27] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Sci. Adv.*, vol. 3, no. 4, Apr. 2017, Art. no. e1602614.

[28] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *J. Sci. Comput.*, vol. 92, no. 3, p. 88, Sep. 2022.

[29] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data," *J. Comput. Phys.*, vol. 394, pp. 56–81, Oct. 2019.

[30] L. Jin, P. N. Nikiforuk, and M. M. Gupta, "Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks," *IEEE Trans. Autom. Control*, vol. 40, no. 7, pp. 1266–1270, Jul. 1995.

[31] K.-I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Netw.*, vol. 6, no. 6, pp. 801–806, Jan. 1993.

[32] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, "Deep neural network approximation theory," *IEEE Trans. Inf. Theory*, vol. 67, no. 5, pp. 2581–2623, May 2021.

[33] P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier, "Deep relaxation: Partial differential equations for optimizing deep neural networks," *Res. Math. Sci.*, vol. 5, no. 3, pp. 1–30, Sep. 2018.

[34] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," *J. Mach. Learn. Res.*, vol. 18, pp. 1–43, Apr. 2018.

[35] C. Tong, "Refinement strategies for stratified sampling methods," *Rel. Eng. Syst. Saf.*, vol. 91, nos. 10–11, pp. 1257–1265, Oct. 2006.

[36] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Rel. Eng. Syst. Saf.*, vol. 81, no. 1, pp. 23–69, Jul. 2003.

[37] Y. Li and F. Mei, "Deep learning-based method coupled with small sample learning for solving partial differential equations," *Multimedia Tools Appl.*, vol. 80, no. 11, pp. 17391–17413, May 2021.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 2067–2075.

[39] M. Lee, "GELU activation function in deep learning: A comprehensive mathematical analysis and performance," 2023, *arXiv:2305.12073*.

[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

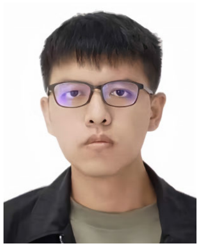
[42] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Comput. Sci.*, vol. 7, p. e623, Jul. 2021.

[43] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–12.

[44] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu, "Physics-informed machine learning: A survey on problems, methods and applications," 2022, *arXiv:2211.08064*.

[45] G. Lewicki and G. Marino, "Approximation by superpositions of a sigmoidal function," *Zeitschrift für Anal. und Ihre Anwendungen*, vol. 22, no. 2, pp. 463–470, Jun. 2003.

[46] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, nos. 1–4, pp. 107–123, Mar. 2000.



ZHAOYANG ZHANG received the B.E. degree from Kunming University of Science and Technology, China, in 2015, where he is currently pursuing the master's degree with the School of Information and Automation. His research interests include wide-ranging, machine learning, optimization theory, and the research of deep neural networks fusing numerical iterative methods to solve partial differential equations.

QINGWANG WANG (Member, IEEE) received the B.E. and Ph.D. degrees in electronics and information engineering and information and communication engineering from Harbin Institute of Technology, Harbin, China, in 2014 and 2020, respectively. From 2020 to 2021, he was a Senior Engineer with Huawei Technology Company Ltd., to study autonomous driving. He joined Kunming University of Technology with a high-level talent. His research interests include machine learning and its application to remote sensing data analysis, autonomous driving, and partial differential equation solving.