System and Control Graduation Report(45 EC)

PHYSICS-INFORMED LEARNING OF THE INKJET PRINTER FIXATION PROCESS

Haoyue Yang
1727095
Supervisor:
prof. dr. Amritam Das

Control System Group, Electrical Engineering Department
Eindhoven University of Technology, August 26, 2024
This report was made in accordance with the TU/e Code of Scientific Conduct for the Master thesis.

CONTENTS

Physics-Informed Learning of the Inkjet Printer Fixation Process *

Haoyue Yang
Eindhoven University of Technology
Eindhoven, Netherlands

***Abstract*—With the continuous advancement of machine learning technologies, various neural networks have demonstrated significant potential in approximating the behavior of complex nonlinear systems. Among them, Physics-Informed Neural Networks (PINNs) have been recently proven to effectively integrate domain-specific physics knowledge with data to enhance generalizatbility. In this research, we develop a PINN framework applicable for monitoring and maintenance of a fixation unit in commercial inkjet printers. The process of fixation describes complex thermo-fluidic phenomena that occur on a newly printed sheet of paper that is subjected to drying. Such process is governed by a system of partial differential equations (PDEs) where the physical coefficients of the PDEs exhibit spatial and temporal dependencies. Irrespective of the PDE system's dimension, our method adeptly manages these dependencies, accurately predicts the spatio-temporal behaviour of the fixation process and estimates corresponding physical parameters that are typically represented by matrix valued multi-variable functions. We conduct extensive numerical experiments and determine the optimal hyperparameters and network configuration, thereby achieving a systematic framework to perform identification of the spatio-temporal dynamics for a given fixation process. Finally, we validate our model using an experimental dataset from a commercially available inkjet printer's fixation unit.**

***Index Terms*—PINN, machine learning, PDE, system identification, parameter identification**

I. Introduction

The integration of physical principles and empirical data plays a crucial role in system simulation. Physical principles provide dynamic models that adhere to the fundamental laws of physics, while empirical data consists of observations and measurements collected from the system. Combining these sources of information can significantly enhance the accuracy and reliability of model identification. In this paper, we develop this concept for the fixation process in inkjet printers. To this end, first, we review several existing data-driven methodologies for model identification, highlighting their strengths and limitations.
[1]

A. Introduction of the Fixation Process

The main process of inkjet printing involves the physical integration of a solid medium and a liquid material, such as paper and ink. This process consists of two primary stages: jetting and fixation. During the jetting stage, liquid ink is precisely deposited onto the solid medium, creating the desired pattern. Once the pattern is printed, the material must undergo drying in the fixation stage to ensure the ink is permanently adhered to the medium.

In the fixation unit, depicted in Figure1, the printed material is transported to the fixation unit on a solid platform which is the belt in the printer. The material passes through a heater and an air impingement unit, which work together to dry the ink. Temperature and moisture sensors distributed on the platform monitor the material's conditions throughout the fixation process, ensuring optimal drying. The printed material in this context can be viewed as a composite material, which is a non-homogeneous substance constructed by interconnecting layers of different materials. The unique properties of composite materials, such as varying thermal and moisture absorption rates across different layers, inputs through the boundary, imperfect contact between the platform and the composite material, present serious challenges in capturing thermo-fluidic behaviour during the fixation process.

Temperature and moisture sensors distributed on the platform monitor the material's conditions throughout the fixation process, ensuring optimal drying. The data can be collected from the sensors distributed from the belt.

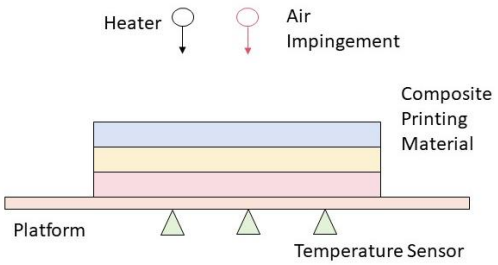


Fig. 1: Overview of Fixation Unit

B. Objective of the thesis

Accurately simulating the fixation process and predicting future states are crucial for optimizing the drying conditions. Effective simulation can help identify the most efficient settings for temperature, airflow, and time required for drying, thereby enhancing the overall print process. This optimization not only ensures high print quality by preventing defects such as smudging and incomplete drying but also reduces waste by minimizing energy consumption and material usage.

[1] ChatGPT is used for grammar and paragraph correction

Moreover, advanced predictive models can contribute to real-time adjustments in the printing process, accommodating variations in environmental conditions and material properties. By integrating physics-based principles with data-driven approaches, these models can provide a comprehensive understanding of the fixation process, leading to continuous improvements in efficiency and quality. Ultimately, the ability to precisely control and optimize the fixation stage is essential for achieving superior performance in inkjet printing technology.

Therefore, the primary objective of this research is to build a surrogate model of the thermo-fluidic phenomena in the fixation process that can accurately predict the spatio-temporal evolution of the temperature and moisture content of printed material in real time.

C. Organization of the Thesis

This thesis is organized into seven comprehensive sections, the structure of the thesis is as follows:

The first section introduces the fundamental principles of printers and the fixation process, followed by a presentation of the primary research objectives.

The second section introduces some preliminaries including the notation and some preliminaries on the standard neural network.

The third section is the problem formulation, which describes the dynamic model of the fixation process and its boundary conditions as constraints. It also details the data sources used in this research. Based on the physics model and the data, we conduct a thorough analysis of the main research problem, breaking it down into several sub-problems, each mathematically formulated. Detailed descriptions of these sub-problems are provided to elucidate their interrelations.

The next section reviews existing approaches relevant to our research and concludes by outlining the specific contributions of this thesis.

Then the next section delves into the methodology for solving the problems. We begin by discussing the optimization principles underlying traditional PINNs and then present our proposed improvements to address each identified sub-problem. Each improved network is explained in detail, emphasizing their theoretical foundations and practical implementations.

In the next section, we present the results of a series of numerical experiments designed to evaluate the performance of our network. The experiments assess various aspects of the network's effectiveness, including its accuracy, and convergence. We also apply the network to real data from the fixation model to demonstrate its practical utility. Detailed analyses of the results are provided, supported by comprehensive tables and figures.

The final section offers a summary of the thesis, encapsulating the key findings and contributions. While our model has shown promising results, it also presents areas that require further refinement and improvement. Additionally, we propose directions for future research, identifying areas where further exploration and development are warranted.

II. Preliminaries

A. Notation

Vector $\begin{bmatrix} x_1^\top & x_2^\top & \dots & x_n^\top \end{bmatrix}^\top$ is denoted by $\mathrm{col}(x_1, x_2, \dots, x_n)$. By $\mathrm{diag}(A, B)$, we denote the block-diagonal matrix $\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$. Similarly, $\mathrm{diag}(B, A_i)_{i=1}^M$ denotes the matrix constructed by putting the matrices $B, A_1, \dots, A_M$ in a block-diagonal fashion, respectively. For a full matrix $\mathbb{R}^{2\times 3} \ni A := \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$, their $(i,j)^{\text{th}}$ entry is denoted by $[A]_{i,j}$, i.e., $[A]_{i,j} = a_{i,j}$ for $i = 1, 2, j = 1, 2, 3$. The same matrix can also represented in terms of its entries as $A = [a_{i,j}]_{i=1,j=1}^{i=2,j=3}$. For a vector $x \in \mathbb{R}^n$, its Euclidean norm is denoted as $||x||_2 := \sqrt{x^\top x}$.

B. Prelimianries on neural networks

In this paper, a *(feedforward) neural network* is any function $H : \mathbb{R}^m \to \mathbb{R}^n$ which can be written as

$$H_k(u) = C_k \sigma_p(A_k u + b_k),\ u \in \mathbb{R}^m \tag{1}$$

for $A_k \in \mathbb{R}^{n_k \times m}$, $b_k \in \mathbb{R}^{n_k}$, $C_k \in \mathbb{R}^{n \times n_k}$. Here $\sigma_k : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ is a diagonal mapping such that the *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ is applied to each coordinate, i.e., $\sigma_k(v) = (\sigma(v_1), \dots, \sigma(v_k))$ for $v \in \mathbb{R}^{n_k}$. In practical applications, these are usually known as networks with *one hidden layer*. Extension to multiple hidden layered neural network architecture simply involve compositions of (1) with different C_k, A_k, b_k. An example of multi-layer neural network would be:

$$F_p(u) := H_{L-1}(H_{L-2}(\dots(H_1(u)))) + d, \tag{2}$$

for specific integer value of L. Learning such a neural network amounts to estimating the (sub)optimal set of parameters p consisting of all the C_k, A_k, b_k and d for a chosen value of L and chosen activation function σ. Such an estimation process involves setting a nonlinear regression problem that minimizes the difference between data and the output of multi-layered neural networks by finding (sub)optimal values of the coefficients C, A, b using backpropagation and stochastic gradient descent.

We let $\mathfrak{N}_{\sigma,p}^{m,n}$ be the class of such networks and define $\mathfrak{N}_\sigma^{m,n} := \cup_{p=1}^\infty \mathfrak{N}_{\sigma,p}^{m,n}$. Throughout the paper, we shall assume that $\sigma : \mathbb{R} \to \mathbb{R}$ is a fixed bounded, continuous and nonconstant function. Under this assumption it is well-known [1] that $\mathfrak{N}_\sigma^{n,m}$ is dense in $C_n(K)$ for any compact set $K \subset \mathbb{R}^m$. That is, for any continuous function $f : K \to \mathbb{R}^n$ and $\varepsilon > 0$ there is a network $h \in \mathfrak{N}_\sigma^{n,m}$ such that $\sup_{x\in K} \|f(x) - h(x)\| < \varepsilon$.

It is also important to note that, provided σ is differentiable, the backpropagation algorithm makes it possible to symbolically computes any derivative of the neural network h and in particular any $\frac{\partial h}{\partial x}$ [2]. This is particularly advantageous when the neural network is used as a surrogate model for design optimization, controller synthesis or solving inverse problem that involves fast computation of the gradients on the model.

III. Problem Formulation

We first address the available resources on how physics laws can be used to mathematically model such a process and, the availability of data. Subsequently, we address the challenges of utilizing the model directly and formulate the research. question.

A. Available Information about the fixation process

1) Physics Model of the Fixation Process: The fixation process is a spatial-temporal thermal-fluidic process which is governed by coupled mass and heat transfer processes varying across both space and time. This provides insights into the transfer and distribution of heat and moisture at different locations within the system and captures how these thermal and moisture transfer phenomena change and evolve over specific time intervals. In Ch.2 of [3], the mathematical model for the fixation process is derived in the graph-theoretic setting. Here, we provide a brief summary of that model. At each layer of the composite printing material, see Fig.1, the governing PDE of is modeled by a partial derivative equation as (3).

$$\begin{aligned}\frac{\partial \mathbf{u_m}\left(s_m,t\right)}{\partial t} =&\mathbf{D}_m\left(s_m,t\right)\frac{\partial^2 \mathbf{u_m}\left(s_m,t\right)}{\partial s_m^2}\\ &+\mathbf{U}_m\left(s_m,t\right)\frac{\partial \mathbf{u_m}\left(s_m,t\right)}{\partial s_m}\\ &+\mathbf{K}_m\left(s_m,t\right)\mathbf{u_m}\left(s_m,t\right)+\mathbf{P}_m\left(s_m,t\right)q(t).\end{aligned} \tag{3}$$

This equation describes the temporal and spatial evolution of the state variable $\mathbf{u_m}\left(s_m,t\right)$ which potentially represents temperature or moisture and some other contents such as ink pigments. Here, s_m is the space variable that takes value from an interval $[s_{\iota,m}, s_{\upsilon,m}]$ where $s_{\upsilon,m} > s_{\iota,m}$, and $s_{\upsilon,m} - s_{\iota,m}$ is the thickness of one layer of the material. Similarly, t is the temporal variable that also takes value from a compact set $\mathbb{T} \subset \mathbb{R}$. Furthermore, $\mathbf{u_m} : [s_{\iota,m}, s_{\upsilon,m}] \times \mathbb{T} \to \mathbb{R}^{n_m}$. In the equation, m is the index of the composite layers as

$$m \in \{1, 2, ..., M\} \tag{4}$$

Here, M is the number of layers. Furthermore, $\mathbf{D}_m(s_m,t) \in \mathbb{R}^{n_m\times n_m}$ means the diffusion coefficient, $\mathbf{U}_m(s_m,t) \in \mathbb{R}^{n_m\times n_m}$ means the transport coefficient and $\mathbf{K}_m(s_m,t) \in \mathbb{R}^{n_m\times n_m}$ means the reaction coefficient in m^{th} layer. $\mathbf{P}_m(s_m,t) \in \mathbb{R}^{n_m\times n_{q,m}}$ is the coefficient for the input $q(t) \in \mathbb{R}^{n_{q,m}}$ for the m^{th} layer.

In [3], a normalization procedure is presented to translate the space coordinate of each layer into the same interval $[s_\iota, s_\upsilon]$, so finally, the dynamics of all layers can be represented by a set of coupled PDEs in the same spatial domain, given by (5).

$$\frac{\partial \mathbf{u}(s,t)}{\partial t} - \mathsf{N}[\mathbf{u};\boldsymbol{\Theta}](s,t) = 0 \quad \forall s \in [s_\iota, s_\upsilon]\,, t \in \mathbb{T}. \tag{5}$$

Here, $\mathbf{u}(s,t) := \text{col}(\mathbf{u}_1(s,t), \ldots, \mathbf{u}_m(s,t))$. The operator $\mathsf{N}[\mathbf{u};\boldsymbol{\Theta}](s,t)$ is the differential operator that represents the relationship between the derivatives of $\mathbf{u}$ with respect to space and time, defined as (6).

$$\mathsf{N}[\mathbf{u};\boldsymbol{\Theta}](s,t) = \boldsymbol{\Theta}(s,t)\begin{bmatrix}\frac{\partial^2 \mathbf{u}}{\partial s^2}(s,t)\\ \frac{\partial \mathbf{u}}{\partial s}(s,t)\\ \mathbf{u}(s,t)\\ q(t)\end{bmatrix} \tag{6}$$

The $\boldsymbol{\Theta}(s,t) \in \mathbb{R}^{\sum_{m=1}^{M} n_m \times \sum_{m=1}^{M} 3n_m + n_{q,m}}$ is the set of parameters in the differential operator represented as

$$\boldsymbol{\Theta} = \begin{bmatrix}\text{diag}(\mathbf{D}_m) & \text{diag}(\mathbf{U}_m) & \text{diag}(\mathbf{K}_m) & \text{diag}(\mathbf{P}_m)\end{bmatrix}. \tag{7}$$

Boundary conditions are essentially to constrain for solving PDE problems. By considering the interface boundary conditions between two layers as well as the external boundary conditions of two extremum layer, and, subsequently, applying the normalization process as in [3], the entire set of boundary conditions can be rewritten as

$$A_{bc}\begin{bmatrix}\mathbf{u}(s_\iota,t)\\ \mathbf{u}(s_\upsilon,t)\\ \frac{\partial \mathbf{u}(s_\iota,t)}{\partial s}\\ \frac{\partial \mathbf{u}(s_\upsilon,t)}{\partial s}\end{bmatrix} = 0 \tag{8}$$

where $A_{bc} \in \mathbb{R}^{2\sum_{m=1}^{M} n_m \times 4\sum_{m=1}^{M} n_m}$ is a constant matrix with full-rank of $2\sum_{m=1}^{M} n_m$. The full rankness is due to the requirements on the sufficient number of boundary conditions that makes (5) wellposed. More discussions on the wellposedness of the considered model is out of scope of this thesis.

Given that the model of the composite material in the fixation process is a PDE, identifying the model parameters $\boldsymbol{\Theta}(s,t)$ can be transformed into an inverse problem of the PDE. This inverse problem involves determining the unknown parameters within the PDE based on observed data, which is crucial for accurately describing the fixation process and its underlying mechanisms.

2) Sources of measurement, simulation data: The measurement data used in this study originate from sensors embedded within a real printer. Another student has been employing these measurements to develop a simulation model using the first principles. Consequently, we have data from both real-world measurements and the physics-based simulation model. Both sets of data serve as prior knowledge in our data-driven approach, facilitating a comprehensive understanding of the system's behavior.

Remark 1: In reality, the physics-based simulation model can be replaced by more complex FEM solvers or already developed mathematical models at the industrial R&D. Since, we only have access to the physics-based model, that is used in this thesis.

Remark 2: In the current simulation fixation model, only constant parameters are considered. However, this is a simplification for now and a discussion with the experts from the industry revealed that, in reality, these coefficients can be spatially and temporally varying.

B. Main Research Question

In the previous subsection, we introduced the physics model along with the unknown parameters that need to be identified, as well as the known data derived from measurements and the simulation model. This leads to the research question of this thesis to which we seek an answer: *How to rapidly simulate the thermo-fluidic processes in an inkjet printer's fixation process and estimate the unknown diffusion, transport and reaction parameters as well as the input profile while taking into account their spatial-temporal dependencies?*

This approach necessitates the integration of measurement data, simulation data, and the physical model of the fixation process. By combining physics and data, we can obtain a convenient and efficient way to get both the solution of the physics model and identify the unknown parameters in the model.

C. Finding the subproblems

There are two main problems related to addressing the above research question. One of them is the data-driven solution and the other is the data-driven discovery.

Data-driven solution involves finding the solution of a specific PDE given its boundary conditions. Data-driven discovery deals with the inverse problem, where the form of the PDE is known, but some parameters within the PDE are unknown. By combining the observed data, the model, along with the boundary and initial conditions, our objective is to, not only, simulate the process to predict its evolution over space and time, but also, infer the unknown parameters of the PDE model.

The physics model is introduced in the previous section as (5), the data we can obtain from the model is the space sampling points in the space domain $[a, b]$

$$\bar{s} = \{s_i | i = 1, 2, ..., N\} \tag{9}$$

the temporal sampling points from the time domain $[0, T_f]$

$$\bar{t} = \{t_i | i = 1, 2, ..., N\} \tag{10}$$

and the corresponding output with the points $[s_i, t_i]$.

$$\mathbf{u}(s_i, t_i), (s_i, t_i) \in \bar{s} \times \bar{t} \tag{11}$$

The objective is to combine the empirical data with the physical model (5) to solve the equation and determine the unknown parameters $\mathbf{\Theta}$.

In this thesis, we make clear distinction between two specific cases on the structure of $\mathbf{\Theta}$. In one case, it is a constant matrix, and in another case, it is a matrix-valued function of s and t. These cases, in turn, help us formulate two subproblems and the combined answers of them will provide a solution to the proposed research question.

1) Simulation and Constant Parameter Estimation of Fixation process: In this case, the parameters $\mathbf{\Theta}$ are in the form of constant matrices which are represented as

$$\mathbf{\Theta} = \Theta^*, \Theta^* \in \mathbb{R}^{\sum_{m=1}^{M} n_m \times \sum_{m=1}^{M} 3n_m + n_{q,m}}$$

and

$$\Theta^* = \begin{bmatrix} D^* & U^* & K^* & P^* \end{bmatrix}$$

Here, D^*, U^*, K^*, P^* are all constant matrices as follows

$$\begin{aligned} D^* = \mathrm{diag}(D_m^*), U^* =& \mathrm{diag}(U_m^*), K^* = \mathrm{diag}(K_m^*), \\ P^* =& \mathrm{diag}(P_m^*). \end{aligned} \tag{12}$$

For each $m \in \{1, 2, ...M\}$, these matrices can be represented in terms of its entries as

$$\begin{aligned} D_m^* = [d_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, U_m^* = [u_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, \\ K_m^* = [k_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, P_m^* = [p_{i,j}]_{i=1,j=1}^{i=n_m, j=n_{q,m}}. \end{aligned} \tag{13}$$

2) Simulation and Estimation of Spatial-Temporal Dependency of Physical Parameters in Fixation Process: Here, $\mathbf{\Theta}$ is more intricate than a simple constant matrix; instead, they depend on both spatial coordinates and time:

$$\mathbf{\Theta} : [s_\iota, s_\upsilon] \times \mathbb{T} \to \mathbb{R}^{\sum_{m=1}^{M} n_m \times \sum_{m=1}^{M} 3n_m + n_{q,m}}$$

and

$$\mathbf{\Theta} = \begin{bmatrix} \mathrm{diag}(\mathbf{D}_m) & \mathrm{diag}(\mathbf{U}_m) & \mathrm{diag}(\mathbf{K}_m) & \mathrm{diag}(\mathbf{P}_m) \end{bmatrix}.$$

Here, for each $m \in \{1, 2, ...M\}$, these matrices can be represented in terms of its entries as

$$\begin{aligned} \mathbf{D}_m(s,t) =& [\mathbf{d}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m, j=n_m}, \\ \mathbf{U}_m(s,t) =& [\mathbf{u}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m, j=n_m}, \\ \mathbf{K}_m(s,t) =& [\mathbf{k}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m, j=n_m}, \\ \mathbf{P}_m(s,t) =& [\mathbf{p}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m, j=n_{q,m}}. \end{aligned} \tag{14}$$

This means that each parameter within the matrix can vary independently with respect to both space and time. This subproblem aims to identify the parameter $\mathbf{\Theta}(s,t)$ and also solve the corresponding PDE model (5).

IV. State of the Art and Main Contributions

In this section, we review existing approaches relevant to our research, including neural networks, data-driven methods for system identification, and approaches that integrate model-based and data-driven techniques. The section concludes by outlining the specific contributions of this thesis.

A. Modeling Approach Based on Data

System identification is a key area in modeling, where the goal is to develop mathematical models of dynamical systems based on observed data [4]. Traditional methods for system identification have relied on statistical techniques [5] which differ depending on the prior choice of model class, typically selected based on physical principles. With the advent of machine learning and neural networks, the domain of system identification perceives a paradigm shift where complex systems are identified using neural networks as the model class and a plethora of off-the-shelf computational tools are aiding to this end.

Machine learning offers a potent avenue for addressing data-driven tasks and leveraging data to simulate models [6].

Serving as a formidable tool in data-driven endeavors, machine learning enables exhaustive data utilization [7], facilitating pattern recognition [8] and relationship discernment, as well as predictive model formulation [7] and informed decision-making [9].

The initial applications of machine learning to system identification focused on neural networks in the late 20th century. Narendra and Parthasarathy [10] demonstrated the use of neural networks for nonlinear system identification and control and showed that neural networks could approximate nonlinear mappings without explicit knowledge of the system's dynamics, offering a powerful alternative to traditional parametric methods.

With the development of deep learning, deep neural networks (DNNs), [11], [12], [13], and recurrent neural networks (RNNs) [14] have been extensively used for modeling dynamic systems. More recently, Long Short-Term Memory (LSTM) networks [15], [16] and Gated Recurrent Units (GRUs) [17], [18] have been employed to capture temporal dependencies in system dynamics.

Gaussian Process (GP) regression [19] is another powerful tool for system identification [20], particularly for systems with uncertain and noisy data. Rasmussen and Williams offered a comprehensive treatment of GPs for machine learning, demonstrating their applicability to system identification [21]. Sparse regression techniques, such as the Sparse Identification of Nonlinear Dynamical Systems (SINDy) algorithm developed by Brunton [22], also have gained popularity for identifying governing equations of dynamical systems.

Despite the significant advancements, several challenges remain in applying machine learning to system identification. The most crucial one of them is ensuring the interpretability and explainability of machine learning models, particularly in safety-critical and engineering systems where principles of physics are at the core of design and function of the system. Integrating domain knowledge with data to improve model generalization is therefore an ongoing area of research.

B. Physics-Informed Neural Network

Solving partial differential equations (PDEs) is another sector where machine learning has made significant inroads. PDEs are fundamental to modeling a wide range of physical phenomena, including fluid dynamics [23], heat transfer [24], and quantum mechanics [25]. Traditional numerical methods for solving PDEs, such as finite difference [26], finite element [27], and finite volume [28] methods, often face limitations related to computational cost as well as scalability. Machine learning approaches offer a promising alternative by combining a physics model and data, leveraging data-driven models to approximate PDE solutions while facilitating the higher degree of explainability and interpretability.

The application of machine learning to PDEs began with the use of neural networks for function approximation. Lee proposed that differential equations can be solved by neural algorithms in 1990 [29]. Lagaris et al. were also among the pioneers, proposing a neural network-based method to solve boundary value problems for PDEs (c.f. [30], [31]). Their approach involved constructing a trial solution that inherently satisfies the boundary conditions, reducing the problem to training a neural network to approximate the remaining part of the solution.

In 2011, a new technique was developed by Ladislav, named Differential Polynomial Neural Network (D-PNN) [32] which estimates a multi-parametric function by generating and solving unknown PDEs. The D-PNN, compared to a regular artificial neural network, allows each neuron in the network to directly calculate the network's overall output. In 2013, Ladislav showed that D-PNNs could solve complex mathematical problems [33]. In 2015, Ladislav et al. showed a recurrent neural network (RNN) of one layer for time series predictions [34].

In 2017, Raissi et al. proposed hidden physics models (machine learning of nonlinear PDEs) [35], the models can exploit underlying physical laws expressed by time dependency and nonlinear PDEs. In the same year, Raissi et al. introduced the prototype of physics-informed neural networks (PINNs) [36]. The algorithms can efficiently use data and encode the underlying physical constraints as prior knowledge.

In 2018, Raissi et al. proposed another technique named "multistep neural networks for the data-driven discovery of nonlinear dynamical systems" [37]. The technique is a machine-learning strategy for identifying nonlinear dynamical systems using data, combined with traditional numerical analysis methods with potent nonlinear function approximations. In the same period, Raissi also proposed a new technique called Deep-Hidden Physics Models [38] to discover nonlinear PDEs using deep learning.

In early 2019, Raissi et al. presented the full version of a physics-informed neural network(PINN) as a "Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations" [39] that can solve the PDEs solution and inverse problems.

In recent years, numerous engineering applications have leveraged Physics-Informed Neural Networks (PINNs), such as combustion optimization systems for coal-fired boilers [40] and water wave simulation [41].

Significant advancements have been made to enhance the PINN framework. Various studies have integrated PINNs with other neural network structures and algorithms. For instance, convolutional neural networks (CNNs) have been utilized in PINN to solve PDEs , [42], recurrent neural networks (RNNs) have been applied for system identification and energy buffer modeling [43], and PINNs have also been employed in reinforcement learning contexts [44].

Researchers have also focused on extending the functionality and improving the performance of PINNs. Jagtap et al. developed a conservative physics-informed neural network that extends PINNs to complex problems, thereby enhancing their generalization capabilities (c.f. [45], [46]). Liu et al. introduced Bayesian physics-informed neural networks, which improve both the accuracy and speed of PINN computations. [47]. Pu et al. advanced PINNs for localized wave solutions of the

nonlinear Schrödinger equation in complex space, resulting in an optimal network with faster convergence [48]. Additionally, Colby et al. proposed an Extended PINN method, which is more effective and accurate for solving larger PDE problems [49].

However, existing research on the inverse problems of PINNs remains limited, particularly for more complex PDE systems. Most studies addressing inverse problems have focused on simpler PDE systems [50] [51] [52] [53], leaving a gap in the application of PINNs to more intricate and high-dimensional inverse problems. This highlights the need for further investigation into extending PINNs to handle more complex inverse problems, which would significantly broaden their applicability and effectiveness in solving real-world engineering and scientific challenges.

C. Main Contributions

In this work, we present a novel method to solve complex partial differential equations (PDEs) with spatial and temporal dependencies. Our contributions can be summarized as follows:

We introduce a robust approach to accurately identify the unknown parameters within the PDEs. This method leverages available data, including the solution of the PDE in specific spatial and temporal intervals, initial conditions, and boundary conditions, to approximate the parameters. Unlike traditional approaches where parameters are often assumed to be constants, our method allows parameters to extend to higher dimensions, enabling the solution of complex systems involving multiple PDEs.

Our method is capable of handling complex dependencies, dealing with PDE systems where parameters are represented as matrices with elements that vary independently across both spatial and temporal dimensions. This capability is crucial for modeling and understanding complex physical systems where parameter behavior is non-uniform and dependent on specific conditions.

To validate our method, we conduct numerical tests and fine-tune the parameters and network structure to identify the optimal hyper-parameters and network configuration for improved results. This iterative process ensures the robustness and accuracy of our model.

Furthermore, our approach is designed to be practically applicable to a wide range of physical systems, such as the dynamic model of the fixation process. We apply our model to the fixation dataset to simulate, analyze, and understand practical systems with complex spatial-temporal dependencies.

Overall, this work represents a significant advancement in the field of PDE analysis by providing a structured and effective method for parameter identification in complex systems. This approach facilitates more accurate and detailed modeling of dynamic spatial-temporal phenomena, enhancing predictive capabilities and optimization of solutions for thermal-fluidic and related problems.

V. Methodology

In this section, we present the method designed to solve the problems in those PDE systems.

A. Physics-informed Neural Network

Let F_p be a neural network according to (2) for a given L and σ that approximates the function u, i.e.,

$$F_p(s,t) \approx \mathbf{u}(s,t). \tag{15}$$

Conventional machine learning entails finding a (sub)optimal network parameters $p^\star$ by solving the following optimization problem on the locations $(s_i,t_i) \in (\bar{s}\times\bar{t})$:

$$p^\star = \arg\min_p \frac{1}{N}\sum_{i=1}^{N} || \ F_p(s_i,t_i) - \mathbf{u}(s_i,t_i) \ ||_2^2 \ . \tag{16}$$

Here $F_p(s_i,t_i)$ represents the prediction value of the network with corresponding input s_i and t_i, $\mathbf{u}(s_i,t_i)$ is the ground truth according to (11). The objective is to minimize the average loss function over a dataset comprising N samples.

Machine learning is a purely data-driven approach that only relies on the data, but does not truly "understand" the physics model given by (5) which represents a partial derivative model in space interval $[s_\iota, s_\upsilon]$ and time interval $\mathbb{T}$.

By integrating machine learning optimizers with differential equations, we define a new constrained optimization problem as shown in Equation (17). This forms a learning problem under specific constraints.

$$\begin{aligned} p^\star &= \arg\min_p \frac{1}{N}\sum_{i=1}^{N} || \ F_p(s_i,t_i) - \mathbf{u}(s_i,t_i) \ ||_2^2, (s_i,t_i) \in (\bar{s}\times\bar{t}) \\ s.t. &\frac{\partial F_p(s,t)}{\partial t} - \mathsf{N}[F_p;\mathbf{\Theta}](s,t) \approx 0 \quad \forall s \in [s_\iota, s_\upsilon] \ , t \in \mathbb{T}. \end{aligned} \tag{17}$$

To solve this new learning problem, we first transform the constraint into an integral form as shown in Equation (18).

$$\begin{aligned} p^\star &= \arg\min_p \frac{1}{N}\sum_{i=1}^{N} || \ F_p(s_i,t_i) - \mathbf{u}(s_i,t_i) \ ||_2^2 \\ s.t. &\iint\limits_{[s_\iota,s_\upsilon]\times\mathbb{T}} || \ \frac{\partial F_p(s,t)}{\partial t} - \mathsf{N}[F_p;\mathbf{\Theta}](s,t) \ ||_2^2 \ \mathrm{d}t\,\mathrm{d}s \approx 0. \end{aligned} \tag{18}$$

Using a Lagrange multiplier $\lambda \geq 0$, (18) can be transformed into the following regularised optimization problem

$$\begin{aligned} p^\star = \arg\min_p \max_{\lambda\geq 0} &\frac{1}{N}\sum_{i=1}^{N} || \ F_p(s_i,t_i) - \mathbf{u}(s_i,t_i) \ ||_2^2 \\ &+ \lambda \iint\limits_{[s_\iota,s_\upsilon]\times\mathbb{T}} || \ \frac{\partial F_p(s,t)}{\partial t} - \mathsf{N}[F_p;\mathbf{\Theta}](s,t) \ ||_2^2 \ \mathrm{d}t\,\mathrm{d}s \end{aligned} \tag{19}$$

To solve the problem, we discretize the integral operator using Riemann sum over a grid $(s_j,t_j) \in \underline{\mathrm{s}} \times \underline{\mathrm{t}}$ where $\underline{\mathrm{s}} =$

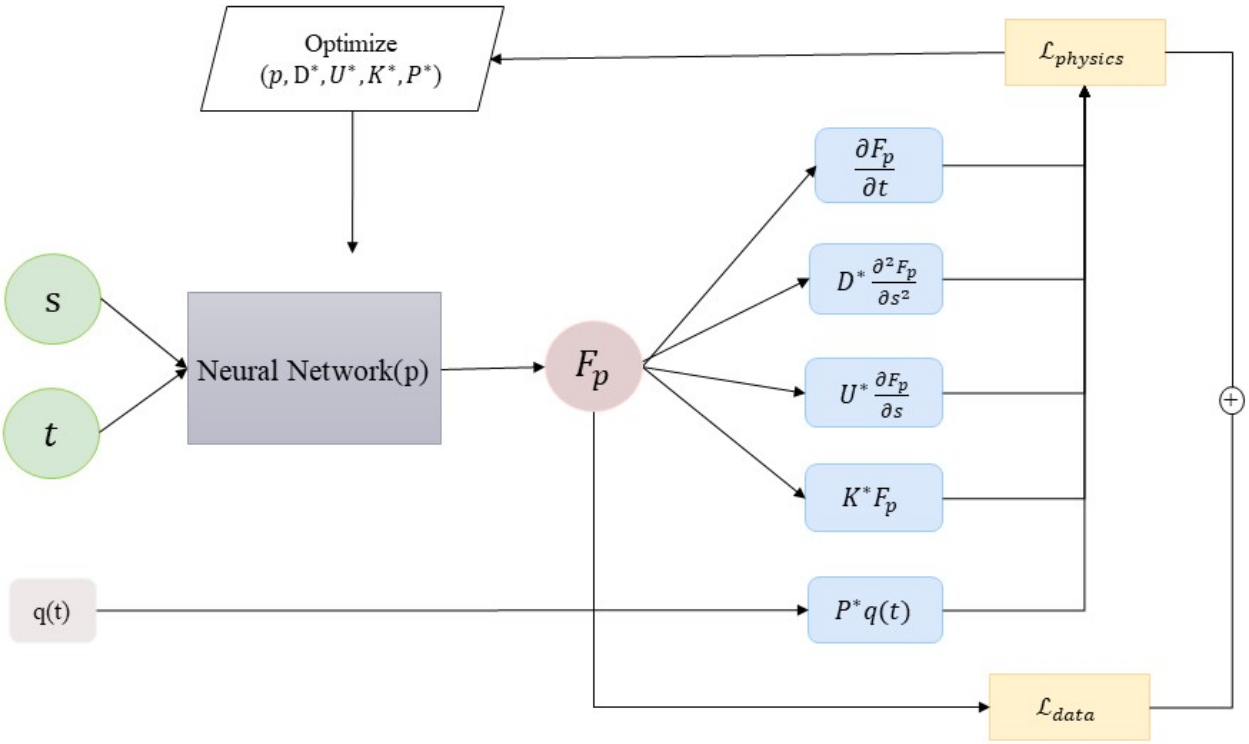


Fig. 2: The structure of the PINN for simulation and constant parameter estimation

$\{s_j | j = 1, 2, ..., N_{\text{phy}}\}, \underline{t} = \{t_j | j = 1, 2, ..., N_{\text{phy}}\}$, leading to the final form of the optimization problem:

$$\begin{aligned} p^\star &= \arg\min_{p} \max_{\lambda \geq 0} \frac{1}{N} \sum_{i=1}^{N} || F_p(s_i, t_i) - \mathbf{u}(s_i, t_i) ||_2^2 \\ &+ \lambda \frac{1}{N_{\text{phy}}} \sum_{j=1}^{N_{\text{phy}}} || \frac{\partial F_p(s_j, t_j)}{\partial t} - \mathsf{N}[F_p; \boldsymbol{\Theta}](s_j, t_j) ||_2^2 \\ &= \text{argmin}_{\boldsymbol{p}} \max_{\lambda \geq 0} \mathcal{L}_\lambda(p) \end{aligned} \tag{20}$$

In (20), $\mathcal{L}_\lambda(p)$ is the loss function of the neural network. There are two items in the loss function. The first item in the equation is called data loss which corresponds to the discrepancy between the predicted and observed data points. The second term is the so-called physics loss which enforces compliance with the underlying physical laws. Two distinct datasets, referred to as data points and physics points, are utilized to train the neural network. N represents the number of data points and N_{phy} represents the number of physics points. The parameter λ acts as a regularization factor in the loss function, balancing the influence of the data loss and the physics cost on the neural network. By optimizing the (20), we can obtain a network that approximates the function $\mathbf{u}(s, t)$.

For the parametric estimation problem of the PDE, the parameter $\boldsymbol{\Theta}(s, t)$ is unknown. Then, the optimization problem (20) is reformulated as

$$\begin{aligned} &\arg\min_{p, \boldsymbol{\Theta}} \max_{\lambda \geq 0} \frac{1}{N} \sum_{i=1}^{N} ||F_p(s_i, t_i) - \mathbf{u}(s_i, t_i)||_2^2 \\ &+ \frac{1}{N_{\text{phy}}} \sum_{j=1}^{N_{\text{phy}}} ||F_p(s_i, t_i) - \mathsf{N}\left[F_p; \boldsymbol{\Theta}\right](s_j, t_j)||_2^2 . \end{aligned} \tag{21}$$

B. Simulation and Constant Parameter Estimation Using Physics Informed Neural Network(PINN)

For this case, We consider the continuous time model as (5). With the constant parameters as

$$\Theta^* = \begin{bmatrix} D^* & U^* & K^* & P^* \end{bmatrix}.$$

Here, D^*, U^*, K^*, P^* are all constant matrices as follows

$$\begin{aligned} D^* = \text{diag}(D_m^*), U^* &= \text{diag}(U_m^*), K^* = \text{diag}(K_m^*), \\ P^* &= \text{diag}(P_m^*). \end{aligned} \tag{22}$$

For each $m \in \{1, 2, ...M\}$, these matrices can be represented in terms of its entries as

$$\begin{aligned} D_m^* &= [d_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, U_m^* = [u_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, \\ K_m^* &= [k_{i,j}]_{i=1,j=1}^{i=n_m, j=n_m}, P_m^* = [p_{i,j}]_{i=1,j=1}^{i=n_m, j=n_{q,m}}. \end{aligned} \tag{23}$$

If we consider (21) as

$$\arg\min_{p, \Theta^*} \max_{\lambda \geq 0} \mathcal{L}_{data}(s_i, t_i; p) + \lambda \mathcal{L}_{\text{physics}}(s_j, t_j; p, \Theta^*), \tag{24}$$

then the data loss in PINN is defined as (25).

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N} ||F_p\left(s_i, t_i\right) - \mathbf{u}(s_i, t_i)||_2^2 \tag{25}$$

The physics loss, on the other hand, is defined as (30)

$$\begin{aligned}\mathcal{L}_{\text{physics}} &= \frac{1}{N_{\text{phy}}}\sum_{j=1}^{N_{\text{phy}}}\left|\left|\frac{\partial F_p}{\partial t}(s_j,t_j) - \mathsf{N}[F_p;\Theta^*](s_j,t_j)\right|\right|_2^2 \\ &= \frac{1}{N_{\text{phy}}}\sum_{j=1}^{N_{\text{phy}}}\sum_{m=1}^{M}\sum_{\ell=1}^{n_m}\sum_{k=1}^{n_m}(d_{\ell,k}\frac{\partial^2 F_{p,k}}{\partial s^2}(s_j,t_j) \\ &+ u_{\ell,k}\frac{\partial F_{p,k}}{\partial s}(s_j,t_j) + k_{\ell,k}F_{p,k}(s_j,t_j) \\ &+ \sum_{h=1}^{n_{q,m}} p_{\ell,h}q_h(t_j) - \frac{\partial F_{p,k}}{\partial t}(s_j,t_j))^2\end{aligned} \tag{26}$$

The structure of the network with multiple outputs is depicted in Figure2. Here, q_h is the h^{th} element of q and $F_{p,k}$ is the k^{th} element of the neural network's output vector. During training, the network minimizes the loss (24) which is the summation of data loss and physics loss to learn both network parameters p and the model parameters $d_{\ell,k}$, $u_{\ell,k}$, $k_{\ell,k}$, $p_{\ell,h}$.

C. Simulation and Estimation of Parameters with Spatial-Temporal Dependency Using Physics Informed Neural Network(PINN)

In systems with constant parameters, these parameters are generally treated as trainable within the neural network framework. However, when parameters vary as functions of space and time, direct updating through standard network training techniques is not feasible. To overcome this challenge, we leverage the neural networks' capability to approximate any continuous function given sufficient neurons and layers. Our approach involves designing additional sub-neural networks specifically to approximate those variable parameters. This methodology integrates a Physics-Informed Neural Network (PINN) with the parameter networks dedicated to approximating the spatial and temporal variations of the parameters to a combined network, then we can get both the model and the parameters in the system.

Here, we define a continuous time model as (5) with the parameters with spatial-temporal dependency as

$$\mathbf{\Theta} : [s_\iota, s_\upsilon] \times \mathbb{T} \to \mathbb{R}^{\sum_{m=1}^{M} n_m \times \sum_{m=1}^{M} 3n_m + n_{q,m}}$$

and

$$\mathbf{\Theta} = \begin{bmatrix}\text{diag}(\mathbf{D}_m) & \text{diag}(\mathbf{U}_m) & \text{diag}(\mathbf{K}_m) & \text{diag}(\mathbf{P}_m)\end{bmatrix}.$$

Here, for each $m \in \{1, 2, ...M\}$, these matrices can be represented in terms of its entries as

$$\begin{aligned}\mathbf{D}_m(s,t) &= [\mathbf{d}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m,j=n_m}, \\ \mathbf{U}_m(s,t) &= [\mathbf{u}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m,j=n_m}, \\ \mathbf{K}_m(s,t) &= [\mathbf{k}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m,j=n_m}, \\ \mathbf{P}_m(s,t) &= [\mathbf{p}_{i,j}(s,t)]_{i=1,j=1}^{i=n_m,j=n_{q,m}}.\end{aligned} \tag{27}$$

Let F_{Φ_D},F_{Φ_U},F_{Φ_K}, and F_{Φ_P} be a neural network according to (2) for a given L and σ that approximates the function $\mathbf{\Theta}(s,t)$ such that

$$\mathbf{\Theta}(s,t) \approx \begin{bmatrix}F_{\Phi_D}(s,t) & F_{\Phi_U}(s,t) & F_{\Phi_K}(s,t) & F_{\Phi_P}(s,t)\end{bmatrix}.$$

In this situation, we can directly use (21) that is decomposed in the following way:

$$\begin{aligned}\arg \min_{p,\Phi_D,\Phi_U,\Phi_K,\Phi_P} \max_{\lambda\geq 0} &\mathcal{L}_{data}(s_i,t_i,p) \\ &+ \lambda\mathcal{L}_{\text{physics}}(s_j,t_j;p,\Phi_D,\Phi_U,\Phi_K,\Phi_P).\end{aligned} \tag{28}$$

The data loss $\mathcal{L}_{data}$ remains identical to before:

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{i=1}^{N} ||F_p\left(s_i, t_i\right) - \mathbf{u}(s_i, t_i)||_2^2 \tag{29}$$

The physics loss $\mathcal{L}_{\text{physics}}$ can be reformulated as

$$\mathcal{L}_{\text{physics}} = \frac{1}{N_{\text{phy}}}\sum_{j=1}^{N_{\text{phy}}}\left|\left|\frac{\partial F_p}{\partial t}(s_j,t_j) - \hat{\mathbf{\Theta}}(s_j,t_j)\begin{bmatrix}\frac{\partial^2 F_p}{\partial s^2}(s_j,t_j) \\ \frac{\partial F_p}{\partial s}(s_j,t_j) \\ F_p(s_j,t_j) \\ q(t_j)\end{bmatrix}\right|\right|_2^2. \tag{30}$$

Here,

$$\hat{\mathbf{\Theta}}(s,t) = \begin{bmatrix}F_{\Phi_D}(s,t) & F_{\Phi_U}(s,t) & F_{\Phi_K}(s,t) & F_{\Phi_P}(s,t)\end{bmatrix}. \tag{31}$$

In the total loss, Φ_D,Φ_D,Φ_D, and Φ_D are the sub-network weight of the parameter networks, which can be updated with the main physics-informed neural network during training.

The structure of the combined network is shown in Figure3. This figure represents an integrated network designed to identify parameters with spatial-temporal dependencies. The combined network consists of a primary PINN and four parameters networks which are used to approximate the four parameters in the system respectively. The inputs to the parameter networks are the spatial coordinate s and time t, and the outputs are the corresponding parameter values. The spatial and temporal coordinates are also the input of the main physics-informed neural network.

The structure of the primary physics-informed neural network remains consistent with the standard PINN. However, it incorporates the outputs from the parameter networks into the physics loss. The combined loss function, which includes both data loss and physics loss is used to update the primary PINN and the parameter networks simultaneously.

Remark 3: The composite structure of the fixation process and the corresponding sparsity in the coefficient matrices induced it (i.e. the block-diagonal structure of the matrices) is not exploited in this thesis. This choice is deliberate and intended to keep the identification framework generic for applications beyond fixation process. If exploiting the sparsity structure is mandatory, one can directly exploit (27) to approximate $\mathbf{d}_{i,j}$, $\mathbf{u}_{i,j}$, $\mathbf{k}_{i,j}$, $\mathbf{p}_{i,j}$ with neural networks.

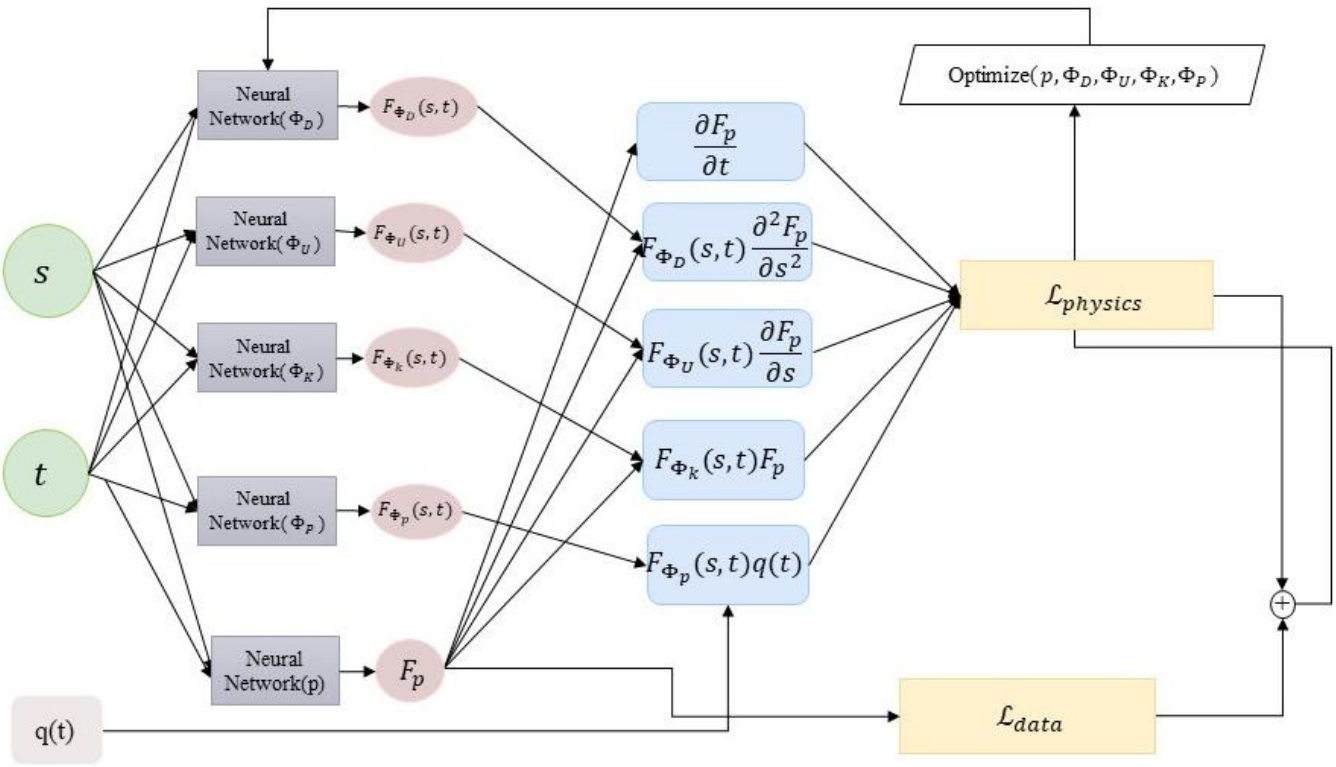


Fig. 3: The structure of the PINN for simulation and estimation of parameters with spatial-temporal dependency

VI. Implementation

This section shows the results of the numerical test we did to evaluate the model in section IV. The application for our network is used in simulation and identification in the fixation model.

A. Numerical Example

we conduct a series of numerical experiments to evaluate the performance of the neural network discussed in the previous section. These experiments aim to rigorously assess the network's effectiveness in identifying various kinds of models. Additionally, we perform an in-depth analysis of the network, examining the influence of different architectural configurations and hyperparameters on its performance.

Throughout the entire implementation process, we consider the Cartesian x-coordinate to define the spatial configuration of the underlying PDEs.

1) Dataset Introduction: The dataset used in our experiments consists of two distinct types of points: data points and physics points. Data points are sampled in both space x and time t, with their corresponding output u derived from the model. In contrast, physics points include only the spatial and temporal coordinates without requiring the output. Consequently, these two datasets are generated using different methods.

Data points necessitate the solution of partial differential equations (PDEs), and thus were generated using the MATLAB PDE solver. On the other hand, physics points were generated using the Latin Hypercube Sampling (LHS) method.

a) MATLAB PDE Solver: The data points are generated using MATLAB's PDE solver,$pdepe$. This solver is capable of handling systems of partial differential equations (PDEs) of the form:

$$c(x,t,u,\frac{\partial u}{\partial x})\frac{\partial u}{\partial t} = x^{-m}\frac{\partial}{\partial x}\left(x^m f(x,t,u,\frac{\partial u}{\partial x})\right) + s(x,t,u,\frac{\partial u}{\partial x})$$

The solver can get the solution of the single PDE system with constant parameters or parameters with spatial dependence. function in MATLAB is invoked as follows

$$sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan, options)$$

m specifies the symmetry of the problem, $m = 0$ means the system is slab symmetry, $m = 1$ means the system is cylindrical symmetry, $m = 2$ means the system is spherical symmetry. $pdefun$ represents the function of the system. $icfun$ and $bcfun$ represent the initial and boundary condition respectively. $xmesh$ is a vector specifying the spatial mesh points, which discretize the spatial domain. $tspan$ specifies the time points at which the solution is computed, which discretizes the temporal domain. $options$ specifies the solver options.

This versatile solver is also capable of solving systems with multiple PDEs. The typical MATLAB code structure for such a system is as follows

```
sol = pdepe(m,@pdefun,@pdeic,@pdebc,x,t);
```

Here, `@pdefun` specifies the function handle to the PDE system definition., `@pdeic` and `@pdebc` specify the function handle for the initial condition and boundary condition respectively.

Our tests of the solver indicate that the multiple PDE solver is capable of solving systems with up to two PDEs. Consequently, in our numerical experiments, we are restricted to using matrix parameters of size 2×2 in our numerical experiment. Additionally, the PDE solver is limited to solving single PDE systems that depend solely on spatial variables, further constraining our testing capabilities.

b) Hyper-Latin Sampling: Latin Hypercube Sampling (LHS) is a sophisticated statistical method employed for generating samples from multidimensional spaces with enhanced coverage and efficiency. Unlike traditional random sampling, which may lead to clustering and uneven representation of the sample space, LHS ensures a more uniform distribution of samples across the entire range of each dimension.

The process of Latin Hypercube Sampling begins with partitioning each dimension of the variable space into n equally probable intervals. This division is essential for the subsequent construction of the Latin hypercube matrix, a core component of the LHS methodology. The matrix is $n \times d$ in size, where n represents the number of desired samples and d denotes the number of dimensions. For each dimension, a random point is selected within each interval, and these points are combined to form the final sample points. This approach guarantees that the entire range of each dimension is covered without redundancy, thus improving the efficiency of the sampling process.

The primary advantage of LHS lies in its ability to provide uniform coverage of the sample space. By ensuring that all intervals within each dimension are represented, LHS mitigates the risk of clustering and gaps that are often observed in random sampling methods. Moreover, LHS is known for its efficiency in utilizing sample sizes. It achieves a comprehensive exploration of the sample space with fewer points compared to traditional methods.

In our study, which involves high-dimensional models, LHS is particularly suitable. Additionally, given our objective of maximizing the informational content of a limited dataset within Physics-Informed Neural Networks (PINNs), LHS provides an effective means of achieving sufficient data representation with fewer samples.

2) Evaluation Metrics: We defined two norms to evaluate the model, the validation loss and the parametric error.

Validation loss is a critical metric used to assess the model's performance on a validation set, which is distinct from the training data. It is computed using the same loss function employed during the training process. This metric measures the discrepancy between the predicted outputs and the actual target values on the validation dataset. In our experiments, we designated 20% of the total data as the validation set for each test to ensure a robust evaluation.

The primary purpose of validation loss is to evaluate how well the model generalizes to unseen data. A lower validation loss indicates better generalization, whereas a higher validation loss suggests that the model may be overfitting to the training data. By monitoring validation loss, we can adjust hyperparameters and employ techniques such as early stopping to improve the model's generalizability.

Parametric error quantifies the precision of parameter estimation by comparing the estimated values against the true parameter values. It serves as a crucial metric in evaluating the accuracy of parameter identification. The parametric error is mathematically defined as (32).

$$
\begin{aligned}
\mathcal{E}_D &= \frac{1}{(n_1+\ldots+n_M)^2}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\left|\hat{d}_{ij}-d_{ij}\right| \\
\mathcal{E}_U &= \frac{1}{(n_1+\ldots+n_M)^2}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\left|\hat{u}_{ij}-u_{ij}\right| \\
\mathcal{E}_K &= \frac{1}{(n_1+\ldots+n_M)^2}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{n_m}\left|\hat{k}_{ij}-k_{ij}\right| \\
\mathcal{E}_P &= \frac{1}{(n_1+\ldots+n_M)(n_{q,1}+\ldots+n_{q,M})}\sum_{m=1}^{M}\sum_{i=1}^{n_m}\sum_{j=1}^{n_{q,m}}\left|\hat{p}_{ij}-p_{ij}\right| \\
\mathcal{E} &= \frac{1}{4}(\mathcal{E}_D+\mathcal{E}_U+\mathcal{E}_K+\mathcal{E}_P)
\end{aligned}
\tag{32}
$$

Here, $\hat{d}_{i,j}$,$\hat{u}_{i,j}$,$\hat{k}_{i,j}$, and $\hat{p}_{i,j}$ denote the estimated values of the entries of coefficient matrices obtained from the training process.

When dealing with parameters exhibiting spatial or temporal dependencies, the parametric error is extended to account for these variations. It is defined as (33).

$$
\begin{aligned}
\mathcal{E}_D &= \frac{1}{N(n_1+\ldots+n_M)^2}\sum_{i=1}^{N}\sum_{m=1}^{n_1+\ldots+n_M}\sum_{n=1}^{n_1+\ldots+n_M} \\
&\qquad \left|[F_{\Phi_D}(x_i,t_i)]_{m,n}-[D(x_i,t_i)]_{m,n}\right|, \\
\mathcal{E}_U &= \frac{1}{N(n_1+\ldots+n_M)^2}\sum_{i=1}^{N}\sum_{m=1}^{n_1+\ldots+n_M}\sum_{n=1}^{n_1+\ldots+n_M} \\
&\qquad \left|[F_{\Phi_U}(x_i,t_i)]_{m,n}-[U(x_i,t_i)]_{m,n}\right|, \\
\mathcal{E}_K &= \frac{1}{N(n_1+\ldots+n_M)^2}\sum_{i=1}^{N}\sum_{m=1}^{n_1+\ldots+n_M}\sum_{n=1}^{n_1+\ldots+n_M} \\
&\qquad \left|[F_{\Phi_K}(x_i,t_i)]_{m,n}-[K(x_i,t_i)]_{m,n}\right|, \\
\mathcal{E}_P &= \frac{1}{N(n_1+\ldots+n_M)(n_{q,1}+\ldots+n_{q,M})}\sum_{i=1}^{N}\sum_{m=1}^{n_1+\ldots+n_M} \\
&\qquad \sum_{n=1}^{n_{q,1}+\ldots+n_{q,M}}\left|[F_{\Phi_P}(x_i,t_i)]_{m,n}-[P(x_i,t_i)]_{m,n}\right|, \\
\mathcal{E} &= \frac{1}{4}(\mathcal{E}_D+\mathcal{E}_U+\mathcal{E}_K+\mathcal{E}_P).
\end{aligned}
\tag{33}
$$

F_{Φ_D},F_{Φ_U},F_{Φ_K}, and F_{Φ_P} represents an estimation function of the parameters D, U, K, and P.

The parametric error provides a quantitative measure of how closely the estimated parameters match the true parameters, thus evaluating the parameter approximation accuracy of the model. A smaller parametric error indicates that the model is accurately identifying the system by closely approximating the true parameters. Conversely, a larger parametric error suggests discrepancies between the model's estimates and the actual parameter values, indicating potential areas for model improvement.

By combining the two evaluation metrics—validation loss and parametric error—we can comprehensively assess the model's performance.

3) The Lagrange Multiplier λ: In the previous section, we mentioned that the constraint problem can use the Lagrange multiplier to solve it in (19) and (20). To solve that min-max optimization problem, an optimization strategy named Primal-dual optimization can be used. In normal optimization, we use Gradient Descent to update the network in the back-propagation. In the min-max problem, we can still update the network parameters by using Gradient Descent but use Gradient Ascent to update λ during the back-propagation.

We try this way on the Single PDE system and compare it with the result from a constant λ, we find that the constant λ and the updating λ can both obtain good results.

So in the implementation test, λ is treated as a regularization parameter in the loss function and tuned for specific test cases. We use Adam [54] as the optimizer in the following experiments.

4) Simulation and Parameter Estimation of Multi-PDEs System: To evaluate the performance of our neural network on systems with constant parameters, we define systems involving both a single partial differential equation (PDE) and multiple coupled PDEs. Due to space constraints, the results and analysis of the single PDE system are provided in the Appendix. Here, we present the system of two coupled PDEs, defined as (34).

$$\frac{\partial u_1(x,t)}{\partial t} = 3\frac{\partial^2 u_1(x,t)}{\partial x^2} - 1.1\frac{\partial^2 u_2(x,t)}{\partial x^2} + 2u_1 + 1.8u_2$$
$$\frac{\partial u_2(x,t)}{\partial t} = 2.7\frac{\partial^2 u_1(x,t)}{\partial x^2} + 1.5\frac{\partial^2 u_2(x,t)}{\partial x^2} + 1.2u_1 - 2.5u_2$$

For the system, the matrix parameters are defined as

$$A = \begin{bmatrix} 3 & -1.1 \\ 2.7 & 1.5 \end{bmatrix}, C = \begin{bmatrix} 2 & 1.8 \\ 1.2 & -2.5 \end{bmatrix}$$

The initial conditions were defined as:

$$u_1 = sin(0.5\pi t), u_2 = sin(0.5\pi t)$$

The left boundary conditions were specified as:

$$\frac{\partial u_1}{\partial x}(0,t) = 0, u_2(0,t) = 0$$

The right boundary conditions were defined as:

$$u_1(1,t) = 1, \frac{\partial u_2}{\partial x}(1,t) = 0$$

The temporal domain and spatial domain were both considered [0,1]. The dataset was generated using the MATLAB PDE solver, as described in the preceding subsection.

To analyze the network and discern the patterns underlying its performance fluctuations, we conducted a series of experiments to fine-tune the network's parameters. All the experiments are implemented by Python 3.10.4.

a) Early stopping: To avoid overfitting and improve the efficiency of the training, we use a technique called early stopping. The main goal of early stopping is to stop training at the right point where the model performs best on the validation set, rather than continuing to train until the performance on the training set reaches its maximum. So in our training process, we evaluate the network on the validation set in every iteration, and if the validation loss doesn't decrease in μ epochs, the training will stop automatically. Since we have two items updated in each iteration, the network parameters and the system parameters may cause the loss to decrease slightly slower. In our experiment, we use 30000 epochs in total for each training and choose $\mu = 3000$.

b) PINN with Different Number of Spatial and Temporal Sampling Points: Firstly, we investigate the effect of the number of data points N_d sampled from the model. The number of sampling points in the space x and time t intervals is a crucial parameter. The training was conducted with a fixed learning rate of 0.001 and a network architecture comprising three layers, each with 32 neurons. We also fixed the number of physics points at 5041.

These experiments' validation loss and parametric error are presented in Table I and Table II.

TABLE I: Validation loss with different sampling points

t \ x	10	20	40	60	80	100
10	0.9858	1.2999	0.6430	0.2992	0.6411	0.8001
20	**0.0237**	**0.0109**	0.2694	0.1400	0.5027	0.6117
40	**0.0199**	0.0555	0.3325	0.3435	0.2613	0.4138
60	0.5402	0.3577	0.1442	0.2214	0.1919	0.2964
80	1.1411	0.3631	0.0583	0.0703	0.2347	0.3197
100	**0.0375**	**0.0288**	0.3319	0.1100	0.1161	0.1562

TABLE II: Parametric error with different sampling points

t \ x	10	20	40	60	80	100
10	4.5554	4.0487	2.7572	3.7139	3.2826	2.9068
20	**0.0412**	**0.0419**	0.1057	0.1013	0.2009	0.1222
40	0.0785	0.1006	0.1450	0.1385	0.1580	0.1338
60	0.1206	0.1677	0.1510	0.1728	0.1532	0.1740
80	**0.0334**	0.1549	0.0684	**0.0535**	0.1587	0.1591
100	0.0747	**0.0559**	0.1328	0.1632	0.1669	0.1584

From Table I, it is evident that the validation loss decreases significantly with an increasing number of sampling points for both x and t, However, the minimal loss does not occur at the maximum number of sampling points. We highlight the five lowest validation losses in the table, all of which correspond to a relatively small number of sampling points for x.The lowest validation loss is observed when $x = 20$ and $t = 20$.

From Table II, it is apparent that the parametric error does not exhibit significant changes when the sampling points exceed 20 in the temporal interval. We have highlighted the five lowest parametric errors in the table, which are primarily concentrated in the left half of the table.

From both tables, we observe that the number of sampling points for x substantially impacts the network's performance. Excessive or insufficient sampling points in the spatial interval can reduce the network's accuracy. An optimal choice appears to be 20 sampling points for both the spatial and temporal intervals.

c) PINN with Different Number of Physics Points: We sample 20 x and t from the space and time interval and obtain 400 data points. The training was still conducted with a fixed learning rate of 0.001 and a network architecture comprising three layers, each with 32 neurons.

The validation loss and parametric error of different amounts of physics points are represented in Table III.

From Table II, a significant reduction in validation loss is observed as the number of physics points increases. However, when the number of physics points exceeds 1024, the downward trend stabilizes, and there is no significant change. Beyond 5041 physics points, the validation loss increases slightly, although it remains considerably lower than the values recorded at 100 and 1024 physics points. The lowest validation loss of 0.0109 is achieved at 5041 physics points.

Similarly, the parametric error shows a marked reduction as the number of physics points increases, with the most significant decline observed from 100 to 5041 physics points. As with the validation loss, the parametric error experiences an increase beyond 5041 physics points. The lowest parametric error is recorded at 5041 physics points.

Both metrics indicate that the number of physics points influences the performance of the network, but the optimal choice is not the maximum possible. These results suggest that both the validation loss and parametric error are minimized at 5041 physics points, identifying this as the optimal number for the given network architecture and training setup.

d) PINN with Different Network Structures: We investigated the influence of different network structures by sampling 20 points in space and time intervals and selecting 5041 physics points. The learning rate was maintained at 0.001, and we varied the number of network layers and the neurons in each layer.

The results of the validation loss and parametric error are presented in Table IV and V respectively.

From Table IV, it is evident that increasing the number of neurons and layers generally reduces the validation loss. However, an overly complex network does not necessarily yield better results. The lowest validation loss (0.0109) is achieved with a network structure of three layers, each containing 32 neurons.

From Table V, increasing the number of neurons generally decreases the parametric error, although there are some variations, sometimes lead to an increase. The lowest parametric error (0.0419) is achieved with a network structure of three layers, each containing 32 neurons, mirroring the results for the validation loss.

These results indicate that both the validation loss and parametric error are minimized with a network structure consisting of three layers, each containing 32 neurons. Increasing the number of layers or neurons beyond this point does not yield significant improvements and, in some cases, may lead to worse performance.

e) Simulation and Parametric Estimation results of the System : Finally, we can get the simulation and parametric estimation results using the optimal hyper-parameters and network structure: 400 data points which sample 20 points in the space and time interval respectively, 5041 physics points, the learning rate is 0.001 and the network has 3 layers with 32 neurons in each, and retaining epochs are 50000 with early stop $\mu = 3000$.

The test dataset has 2500 points which are not included in the data points. The final results are shown in Fig.4.

Fig.4(a) and Fig.4(b) provide a detailed representation of the parameter approximation process over successive epochs. Each figure delineates the predicted values for the four elements within the parameter matrices A and C. The dotted lines, which match the corresponding predicted values' colors, indicate these elements' true values. The visualized data reveal a convergence trend wherein the predicted values increasingly align with the true values as the training epochs progress. This convergence underscores the network's efficacy in parameter estimation.

Fig.4(c) and Fig.4(d) depict the predicted solutions for the partial differential equations (PDEs) under consideration, while Fig.4(e) and Fig.4(f) present the true solutions of the same PDE system. A comparative analysis reveals a close alignment between the predicted and true solutions, indicating the networks successfully simulate the system.

These results prove the efficacy of the proposed neural network model in dealing with intricate PDE systems with constant matrix parameters.

5) Simulation and Parameter Estimation of System with Spatial-Temporal Dependence Parameters:

a) PDE with Spatial Dependence Parameter: For the limited function of the open-source PDE solver, we can only obtain a dataset for the system including a single PDE with spatial dependence. So in this section, we use the system of single PDE with only spatial dependence to test our network and analyze the influence of the structure and the hyper-parameters.

We use a specific example which solves the diffusion equation with a heterogeneous diffusivity as

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial}{\partial x}[D(x,t)\frac{\partial}{\partial x}u(x,t)]$$

Here, we define the diffusion coefficient as

$$D(x) = 1.01 + tanh(x)$$

So the diffusion equation can be transformed as

$$\frac{\partial u(x,t)}{\partial t} = f(x)\frac{\partial^2 u(x,t)}{\partial x^2} + g(x)\frac{\partial u(x,t)}{\partial x}$$

TABLE III: Validation loss and Parametric error with different physics points

physics points	100	1024	2025	4096	5041	6084	8100	1000
validation loss	1.0443	0.7179	0.0184	0.0337	**0.0109**	0.0384	0.0821	0.0285
parametric error	2.7235	0.3394	0.1911	0.0947	**0.0419**	0.2147	0.2528	0.2006

TABLE IV: Validation loss with different network structure

layers \ neurons	16	32	64
1	1.8858	1.3988	0.9532
2	1.4420	1.1007	0.4007
3	1.3411	**0.0109**	0.7578
4	0.5334	0.0458	0.1516

TABLE V: Parametric error with different network structure

layers \ neurons	16	32	64
1	0.7166	0.6497	0.8709
2	0.3191	0.1202	0.2711
3	0.3525	**0.0419**	0.4245
4	0.1469	0.1485	0.0555

In this case, $f(x) = 1.01 + tanh(x)$ and $g(x) = 1.01 - tanh^2(x)$ are the parameters with spatial dependence in the PDE system.

The initial condition of the system is defined as

$$u = sin(\pi t)$$

The boundary condition is defined as

$$u(-5, t) = 0, u(5, t) = 0$$

The temporal interval is [0,1], and the spatial interval is [-5,5]. The dataset is also made by the PDE solver introduced in the previous subsection.

All the experiments are implemented by Python 3.10.4.

b) PINN with Different Number of Training Data: To investigate the influence of the number of training data, we also do several experiments varies number of data points or physics points to see the changes of the validation loss and parametric error.

Firstly, we observe the impact of the data points, so we fix the physics points at 5041, the learning rate at 0.001, and use a 3-layer network with 32 neurons on each. We plan to test the network on different sampling points on space and time intervals, but since the training time is very long for this network, due to the limitation of time, we only have the results on the sampling points $t = 100$ and $x = 5, 10, 20, 30, 40, 50, 80$. The results are represented in Fig.7 (a) (c) (d).

Then we choose 4000 data points, change the physics points to 500,1000,2000,3000,4000,5000 and 8000, and use the same hyper-parameters, the results are represented in Fig.7 (b) (d) (f).

Fig.5 (a) and (b) illustrate the convergence behavior of the validation loss over the initial 2000 epochs for various quantities of data points and physics points, respectively. In Fig.5 (a), it is observed that the convergence rate of the validation loss is notably influenced by the number of data points. Both very large and very small data volumes result in slower convergence, indicating that there is an optimal range for data quantity that facilitates more efficient training. Conversely, Fig.5 (b) demonstrates that the number of physics points has minimal impact on the convergence speed, suggesting that the primary driver of convergence efficiency is the quantity of data points rather than physics points.

Fig.5 (c) (d) depicts the validation loss after 80,000 epochs for varying numbers of data points and physics points. In Fig.5 (c), it is evident that an increase in the number of data points does not consistently reduce the validation loss. Instead, both excessive and insufficient data volumes are associated with higher validation loss. This suggests that there is a threshold beyond which additional data does not contribute to improved model performance and may even degrade it due to potential noise or overfitting. Similar trends are observed in Fig.5 (d) for different physics points, indicating that validation loss is relatively insensitive to changes in the number of physics points within the tested range.

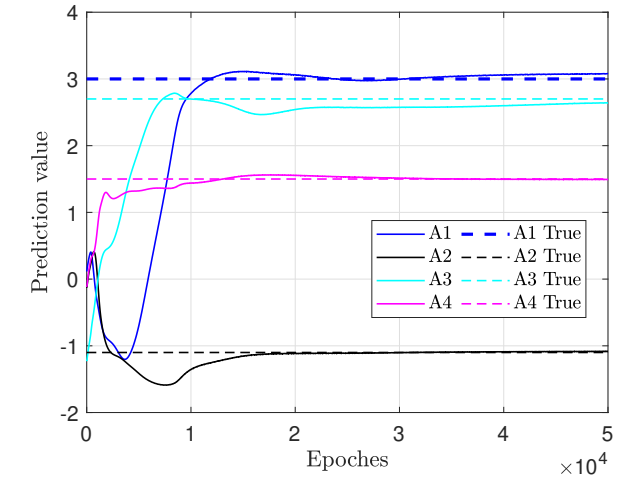
Fig.5 (e) provides insights into the parametric error after 50,000 epochs, focusing on varying data points. The results show that the parametric error decreases with an increase in the number of data points up to a certain level (4000 data points), beyond which it begins to rise. This behavior suggests that while more data generally improves parameter estimation accuracy, there is an optimal data volume beyond which the benefits diminish and potential complexities or noise from the excess data may lead to increased error. The influence of physics points on parametric error appears minimal, with only slight variations observed when physics points exceed 1000.

Integrating the observations from validation loss and parametric error analyses, it is concluded that an optimal balance is achieved with 4000 data points and 4000 physics points.

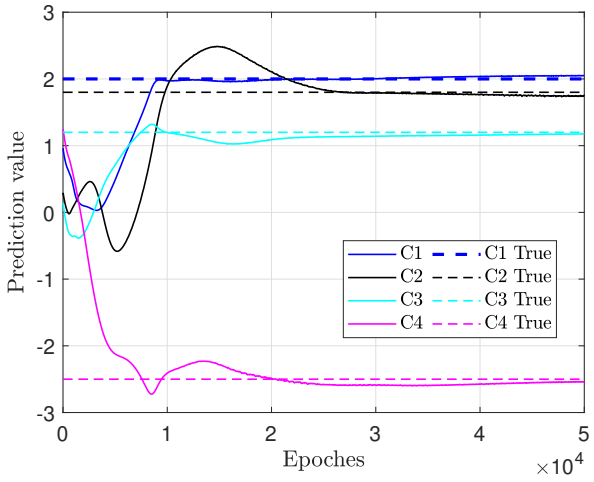
c) PINN with Different Learning Rate: To observe the influence of the learning rate and find the optimal, we select 4000 data points and physics points respectively, fix the network structure as 3 layers 32 in each, and decrease the learning rate from 0.1 to 0.0001, the estimation of the parameters and the validation loss is represented in Fig.6.

Fig.6 (a) illustrates that a larger learning rate results in faster convergence during the training process. However, Fig.6 (b) smaller learning rates result in lower loss values, indicating more precise parameter estimation and better overall model performance. This suggests a trade-off between convergence speed and estimation accuracy, necessitating a balanced approach to learning rate selection.

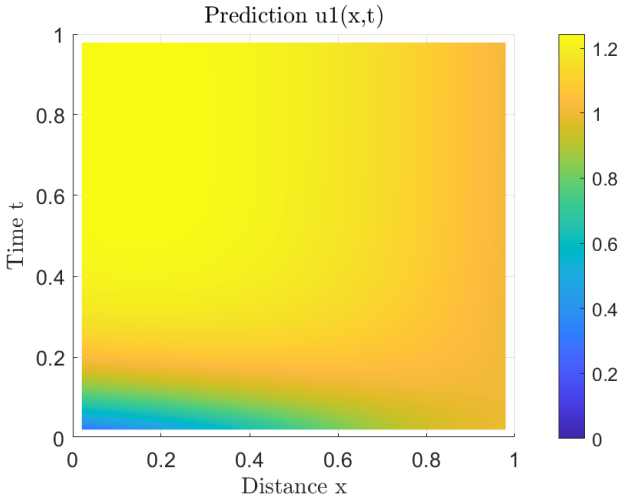
d) Fine-tuning the Learning Rate: Given that the model's training cost is substantial and the parametric estimation



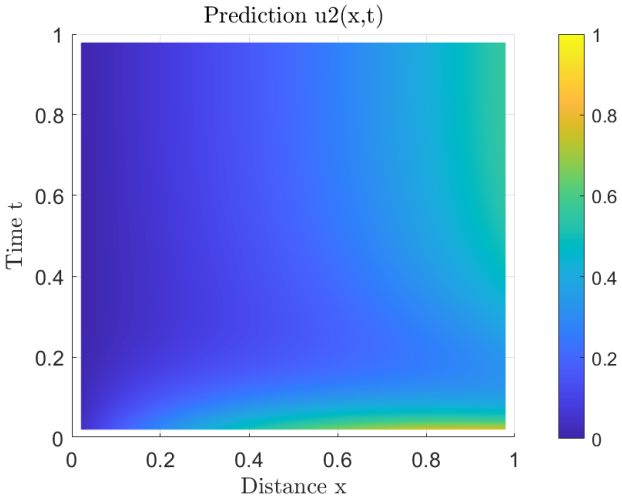
(a)



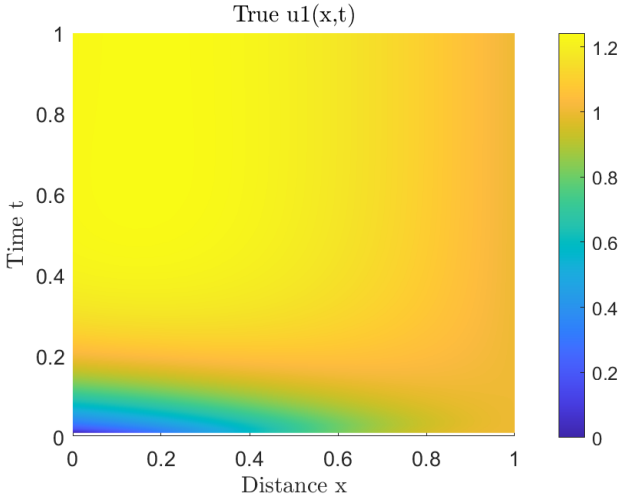
(b)



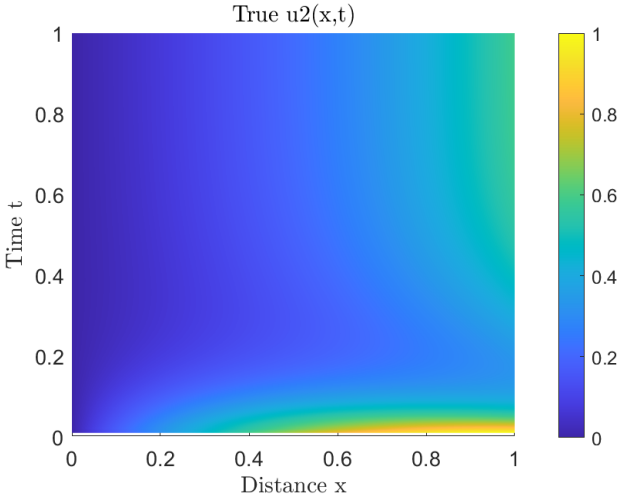
(c)



(d)



(e)



(f)

Fig. 4: (a) (b) are the estimation of elements value in matrix A and C in the first 50,000 epochs, (c) and (d) is the prediction of the PDE solution u_1 and u_2, (e) (f) are the true value of the solution u_1 and u_2 from the dataset

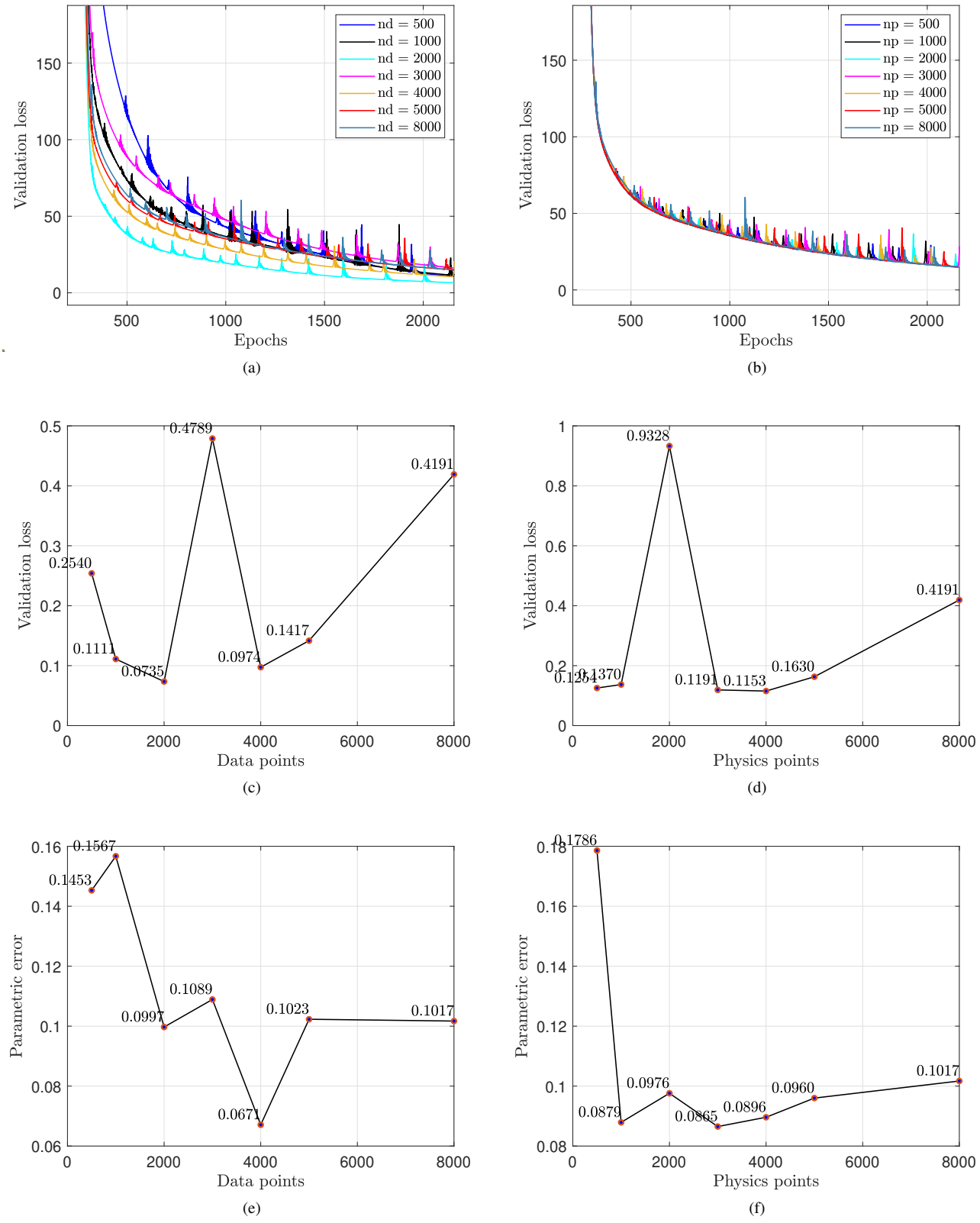


Fig. 5: (a) represents the validation loss in different data points in the first 2000 epochs. (b) represents the validation loss in different physics points in the first 2000 epochs. (c) represents the validation loss after 50,000 epochs in different data points. (d) represents the validation loss after 50,000 epochs in different physics points. (e) represents the parametric error after 50,000 epochs in different data points. (f) represents the parametric error after 50,000 epochs in different physics points.

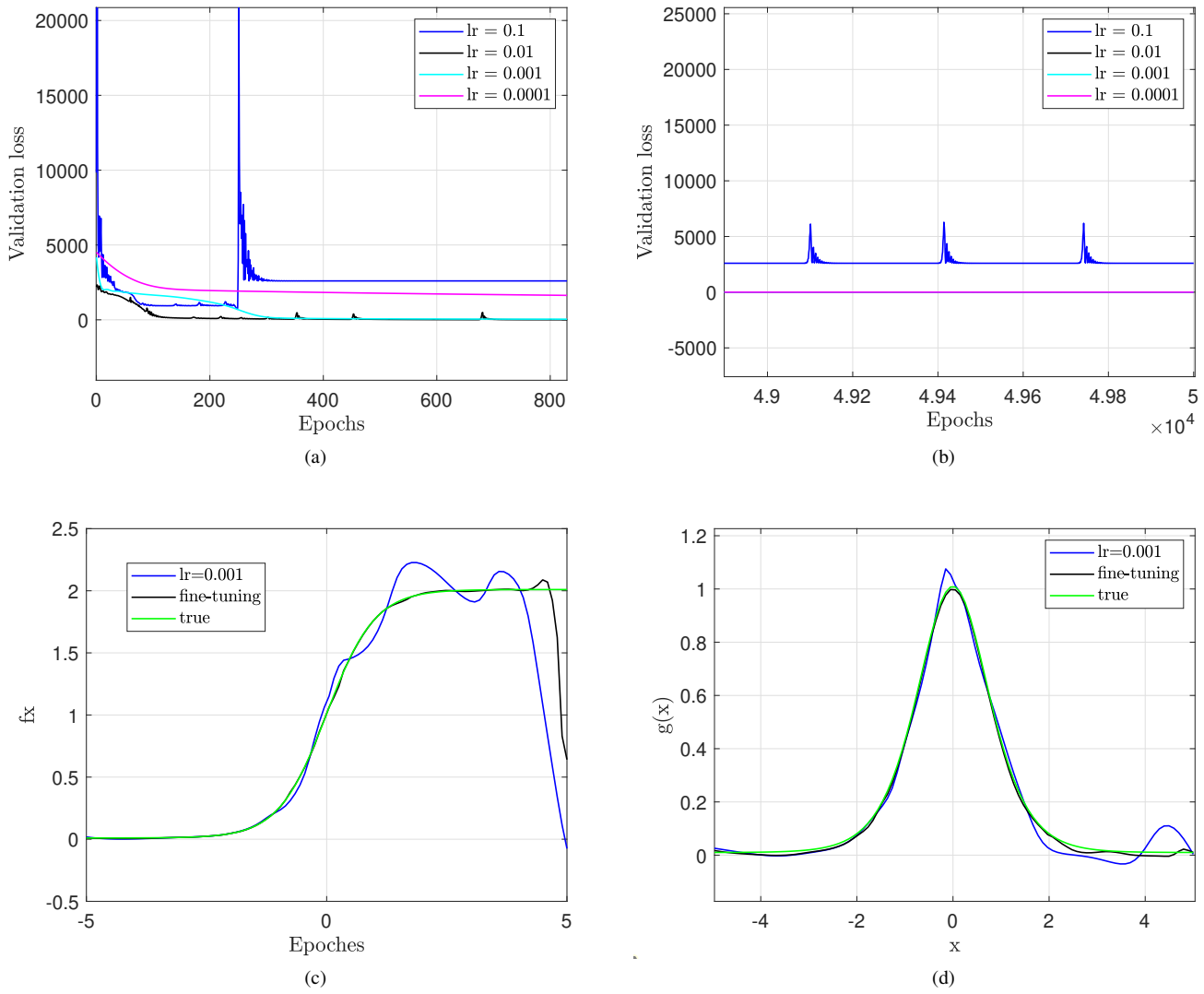


Fig. 6: (a) (b)represents part of the validation loss of the network with different learning rates. (c) represents the comparison the estimation of $f(x)$ between the constant learning rates and fine-tuning learning rate. (d) represents the comparison the estimation of $g(x)$ between the constant learning rates and fine-tuning learning rate

results remain suboptimal, an effective strategy is to employ a method known as fine-tuning. This technique involves initially using a larger learning rate to accelerate convergence. When the loss reaches a plateau and ceases to improve over a specified number of epochs, the learning rate is reduced. This approach facilitates continued training with a finer adjustment to the learning rate, allowing for more precise optimization and improved parameter estimation.

The comparison of the estimation results after 50,000 epochs of two parameters between the standard training process with a fixed learning rate of 0.001 and the fine-tuning approach is depicted in Fig.6 (c) (d). In the fine-tuning process, the learning rate was initially set to 0.01 and subsequently reduced to 0.001 after several epochs of no improvement in the loss function. This gradual reduction continued, with the learning rate decreasing by an order of magnitude whenever the loss stagnated for a specified number of epochs. The results clearly indicate that the fine-tuning approach leads to significant improvements in parameter estimation accuracy compared to the standard fixed learning rate training.

e) Simulation and Parametric Estimation of the System with Spatial Dependence Parameters: In the final stage of our experiments, we utilized 4000 physics points and 4000 data points, implementing a fine-tuning strategy for the learning rate. After training the network for 200,000 epochs, the results are depicted in Fig.7. Fig.7 (a) and (b) illustrate the estimated parameter functions $f(x)$, respectively. The results demonstrate that the network effectively fits the parameter functions, though minor discrepancies are observed near the boundary conditions. We try to train more epochs but the results in the boundary have no significant improvement, these boundary errors indicate areas for potential improvement in future work.

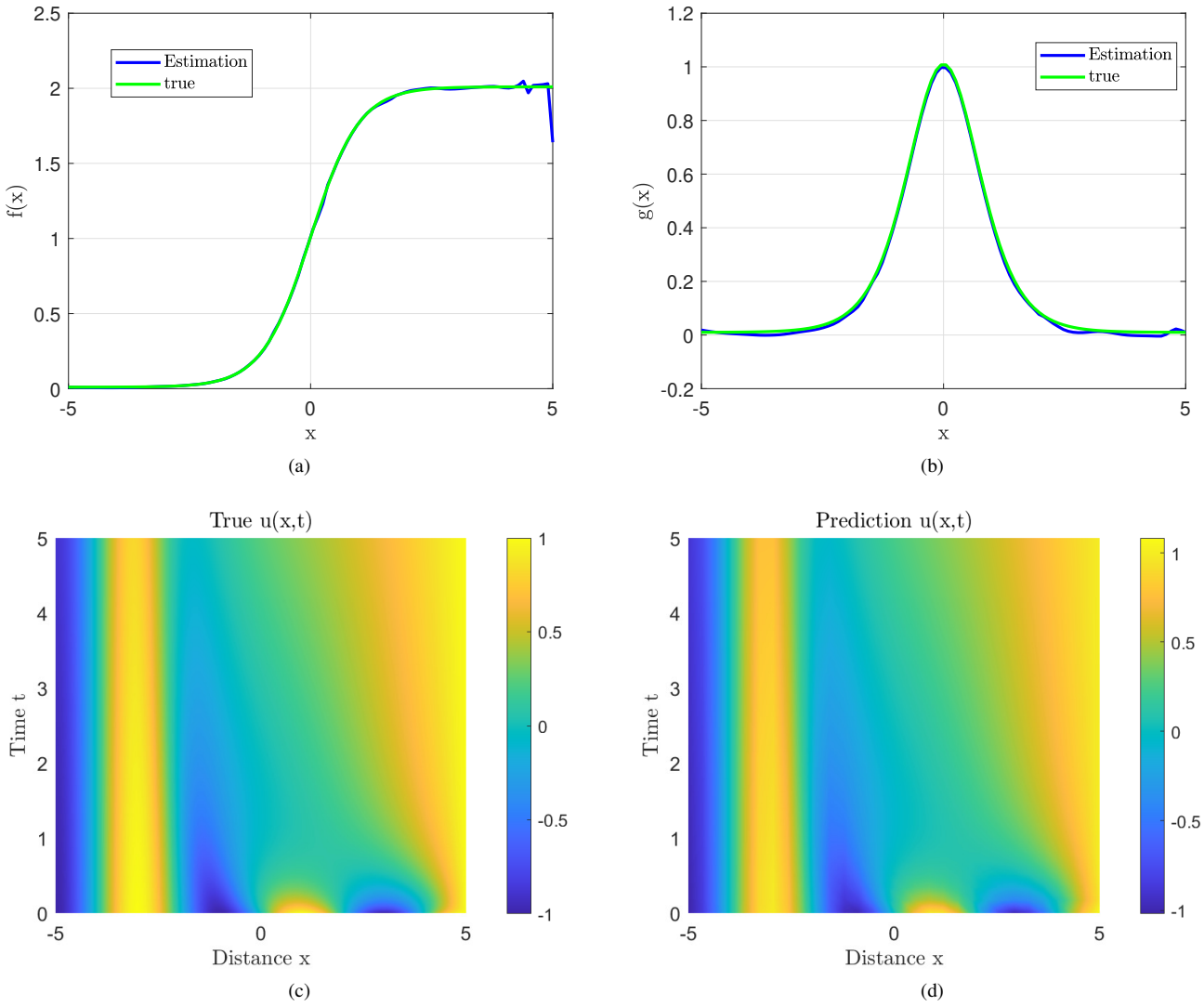


Fig. 7: (a) (b)represents the estimation of the parameter $f(x)$ and $g(x)$ (c) represents the true solution of PDE system. (d) represents predict solution from the network.

Fig.7 (c) and (d)present a comparison between the PDE solutions generated by the network and the corresponding data. The network's solutions exhibit a high degree of agreement with the data, indicating robust performance in modeling the PDE.

B. Application: the Fixation Model

1) Model Introduction: The model in the practical case is a PDE with the diffusion coefficient. which is described as (35).

$$\frac{\partial \mathbf{T_i}\left(x_i,t\right)}{\partial t} = D_i\left(x_i\right)\frac{\partial^2 \mathbf{T_i}\left(x_i,t\right)}{\partial x_i^2} \tag{35}$$

In the equation, $T_i(s_i,t)$ means the temperature in the i_{th} layer of the composite material, s and t represent the space and time. The practical model is also built in another physics way, so there's the true value of comparing the results from our network with the real one. The model of normal paper is built of 5 layers, which means $i=5$, so the model is a system consisting of 5 PDEs as

$$\begin{bmatrix} \frac{\partial T_1}{\partial t} \\ \frac{\partial T_2}{\partial t} \\ \frac{\partial T_3}{\partial t} \\ \frac{\partial T_4}{\partial t} \\ \frac{\partial T_5}{\partial t} \end{bmatrix} = \begin{bmatrix} 2500 & 0 & 0 & 0 & 0 \\ 0 & 0.8877 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 1.7501 & 0 \\ 0 & 0 & 0 & 0 & 250 \end{bmatrix} \begin{bmatrix} \frac{\partial^2 T_1}{\partial s^2} \\ \frac{\partial^2 T_2}{\partial s^2} \\ \frac{\partial^2 T_3}{\partial s^2} \\ \frac{\partial^2 T_4}{\partial s^2} \\ \frac{\partial^2 T_5}{\partial s^2} \end{bmatrix} \tag{36}$$

Here, the diffusion coefficient D is defined as a 5×5 matrix.

The initial condition is defined as

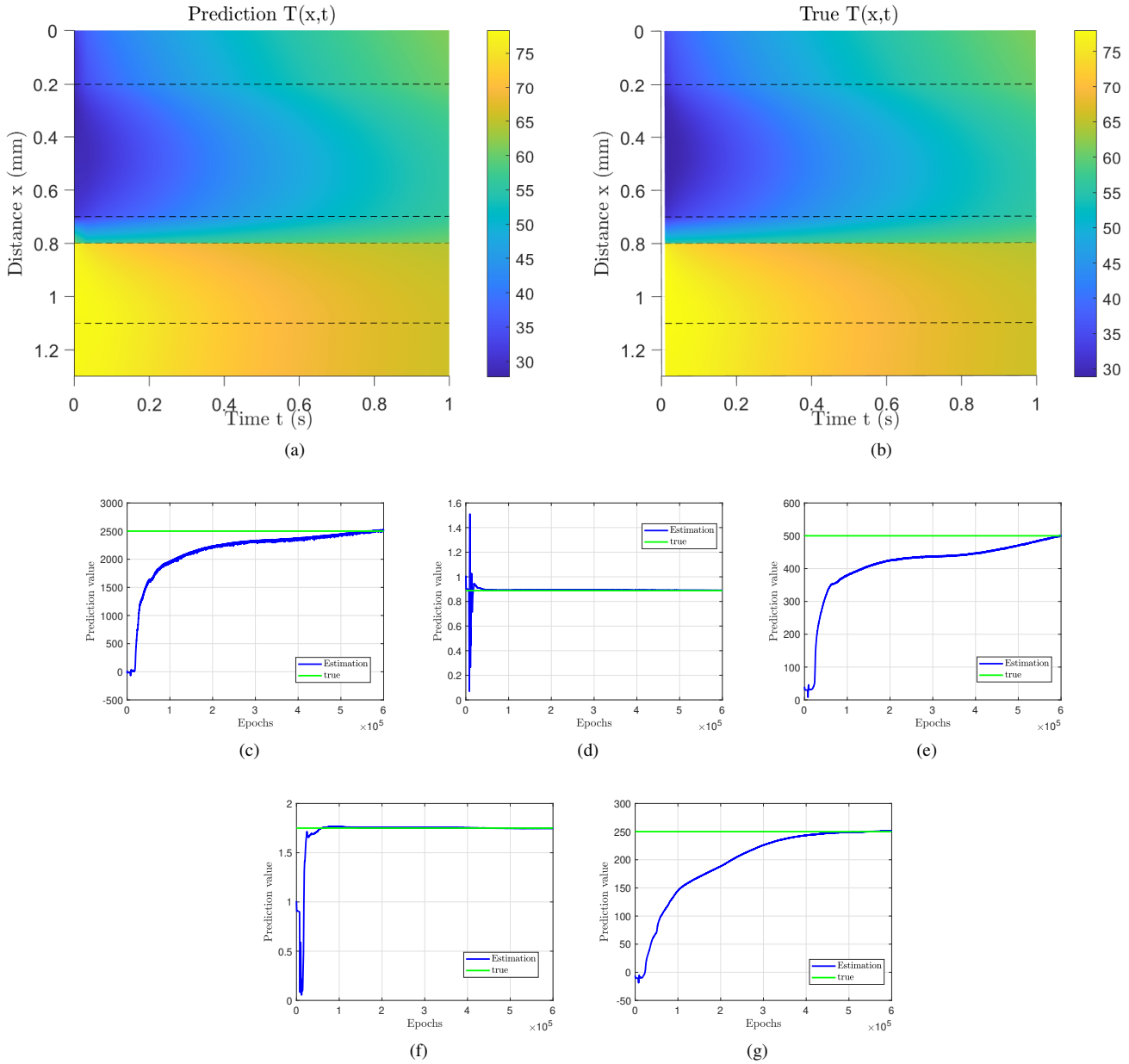


Fig. 8: (a) represents the prediction of the temperature in 5 layers. (b) represents the true temperature of 5 layers. (c) (d) (e) (f) (g) represents the convergence of the parameters in the diffusion matrix

The boundary condition is defined as

$$
\begin{aligned}
340T_1(0,t) - 5000\frac{\partial T_1}{\partial x}(0,t) &= 0 \\
490\frac{\partial T_2}{\partial x}(0,t) - 5000\frac{\partial T_1}{\partial x}(1,t) &= 0 \\
500\frac{\partial T_3}{\partial x}(0,t) - 490\frac{\partial T_2}{\partial x}(1,t) &= 0 \\
-600T_4(0,t) + 600T_3(1,t) + 833.3333\frac{\partial T_4}{\partial x}(0,t) &= 0 \\
-600T_4(0,t) + 600T_3(1,t) + 500\frac{\partial T_3}{\partial x}(1,t) &= 0 \\
500\frac{\partial T_5}{\partial x}(0,t) - 833.3333\frac{\partial T_4}{\partial x}(1,t) &= 0 \\
-T_2(0,t) + T_1(1,t) &= 0 \\
-T_3(0,t) + T_2(1,t) &= 0 \\
-T_5(0,t) + T_4(1,t) &= 0 \\
20T_5(1,t) + 500\frac{\partial T_5}{\partial x}(1,t) &= 0
\end{aligned}
$$

2) Implementation: This study involves approximating the diffusion coefficient D in the fixation system using our neural network. The implementation details are discussed in the third section. For training the model, we utilized 5000 data points and 5000 physics points. We select 10% of the training data as a validation set. And generate the test set includes 900 data points that are not in the training set. The network architecture comprised three layers with 32 neurons each.

Given the presence of several large parameters that update slowly during training, we employed different optimizers and set varying learning rates to expedite the training process. Specifically, we used a learning rate of 0.001 for the general

PINN training, 0.1 for the first parameter, and 0.01 for the third and fifth parameters. The results of this approach are depicted in Fig.8.

3) Results Representation: Fig.8 illustrates the convergence of the parameters in subfigures (c) (d) (e) (f) (g), the green line is the true value and the blue curve is the value of the parameter. Since some of the parameters are very large, the convergence is slow.

In Fig.8 (a) and (b), we compare the model simulations generated by the network with the ground truth in 5 layers. Generally, our network accurately simulates the model.

However, slight differences remain between the predicted values and the ground truth, particularly in subfigures (a) and (d) within the interval $t \in [0, 0.2]$. This indicates an area where accuracy can be improved in future work. Additionally, the training process remains lengthy, suggesting a potential area for optimization.

Overall, while our approach effectively identifies the diffusion coefficient and provides accurate PDE solutions, further work is required to enhance training efficiency, improve boundary accuracy, and reduce computational time.

VII. Conclusion

In this thesis, we have conducted a comprehensive study on the application of Physics-Informed Neural Networks (PINNs) for system identification, focusing on the fixation model. Our research demonstrates the effectiveness of integrating physics-based models with data-driven approaches to enhance the accuracy and reliability of system identification beyond traditional methods.

We developed a robust model for the spatial-temporal thermal-fluidic fixation process, effectively managing initial and boundary conditions. This approach accurately identifies unknown PDE parameters using available data, allowing for higher-dimensional parameter extension and the solution of complex systems involving multiple PDEs. Our methodology addresses PDE systems with parameters varying independently across spatial and temporal dimensions, which is crucial for modeling complex physical systems. The applicability of our network to practical systems highlights its significant utility in engineering. Numerical tests validated our method, optimizing hyperparameters and configurations to ensure robustness and accuracy. Applying the model to the fixation dataset demonstrated its practical utility in understanding systems with complex spatial-temporal dependencies.

Future research should focus on further optimizing the model to accelerate convergence and enhance accuracy, as well as improving the robustness of the model by introducing noise analysis. Additionally, testing the model on systems with matrix-form datasets containing time-space variation function parameters is crucial, as this has not been explored due to data availability constraints. These directions will help in refining the model and expanding its applicability to more complex and varied physical systems.

REFERENCES

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*. Cambridge, MA, USA: MIT Press, 1988, p. 696–699.

[3] A. Das, "A digital twin for controlling thermo-fluidic processes," Phd Thesis 1 (Research TU/e / Graduation TU/e), Electrical Engineering, Nov. 2020, proefschrift.

[4] K. J. Åström and P. Eykhoff, "System identification—a survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[5] K. Soal, Y. Govers, J. Bienert, and A. Bekker, "System identification and tracking using a statistical model and a kalman filter," *Mechanical Systems and Signal Processing*, vol. 133, p. 106127, 2019.

[6] T. G. Dietterich, "Machine learning," *Annual review of computer science*, vol. 4, no. 1, pp. 255–306, 1990.

[7] A. Ganapathi, "Predicting and optimizing system utilization and performance via statistical machine learning," Ph.D. dissertation, UC Berkeley, 2009.

[8] S. Alhusain, S. Coupland, R. John, and M. Kavanagh, "Towards machine learning based design pattern recognition," in *2013 13th UK Workshop on Computational Intelligence (UKCI)*. IEEE, 2013, pp. 244–251.

[9] J. Porciello, M. Ivanina, M. Islam, S. Einarson, and H. Hirsh, "Accelerating evidence-informed decision-making for the sustainable development goals using machine learning," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 559–565, 2020.

[10] K. S. Narendra and K. Parthasarathy, "Neural networks in dynamical systems," in *Intelligent Control and Adaptive Systems*, vol. 1196. SPIE, 1990, pp. 230–241.

[11] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *International journal of control*, vol. 56, no. 2, pp. 319–346, 1992.

[12] H. Yi, S. Shiyu, D. Xiusheng, and C. Zhigang, "A study on deep neural networks framework," in *2016 IEEE advanced information management, communicates, electronic and automation control conference (IMCEC)*. IEEE, 2016, pp. 1519–1522.

[13] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," *arXiv preprint arXiv:1610.01439*, 2016.

[14] J.-S. Wang and Y.-P. Chen, "A fully automated recurrent neural network for unknown dynamic system identification and control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 6, pp. 1363–1372, 2006.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] R. Zazo, A. Lozano-Diez, J. Gonzalez-Dominguez, D. T. Toledano, and J. Gonzalez-Rodriguez, "Language identification in short utterances using long short-term memory (lstm) recurrent neural networks," *PloS one*, vol. 11, no. 1, p. e0146917, 2016.

[17] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2013, pp. 207–215.

[20] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, "Dynamic systems identification with gaussian processes," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005.

[21] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[22] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[23] F. Flandoli, *Random Perturbation of PDEs and Fluid Dynamic Models: École d'été de Probabilités de Saint-Flour XL–2010*. Springer Science & Business Media, 2011, vol. 2015.

[24] A. F. Mills, *Heat transfer*. CRC Press, 1992.

[25] A. Messiah, *Quantum mechanics*. Courier Corporation, 2014.

[26] J. C. Strikwerda, *Finite difference schemes and partial differential equations*. SIAM, 2004.

[27] R. Gallagher, J. Oden, C. Taylor, and O. Zienkiewicz, "Finite element," *Analysis: Fundamentals. Prentice Hall, Englewood Cliffs*, 1975.

[28] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," *Handbook of numerical analysis*, vol. 7, pp. 713–1018, 2000.

[29] H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," *Journal of Computational Physics*, vol. 91, no. 1, pp. 110–131, 1990.

[30] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[31] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[32] L. Zjavka, "Construction and adjustment of differential polynomial neural network," *Journal of Engineering and Computer Innovations*, vol. 2, no. 3, pp. 40–50, 2011.

[33] ——, "Recognition of generalized patterns by a differential polynomial neural network," *Engineering, Technology & Applied Science Research*, vol. 2, no. 1, pp. 167–172, 2012.

[34] L. Zjavka and V. Snáel, "Composing and solving general differential equations using extended polynomial networks," in *2015 International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 2015, pp. 110–115.

[35] M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," *Journal of Computational Physics*, vol. 357, pp. 125–141, 2018.

[36] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[37] ——, "Multistep neural networks for data-driven discovery of nonlinear dynamical systems," *arXiv preprint arXiv:1801.01236*, 2018.

[38] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," *Journal of Machine Learning Research*, vol. 19, no. 25, pp. 1–24, 2018.

[39] 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," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[40] Y. Cheng, Y. Huang, B. Pang, and W. Zhang, "Thermalnet: A deep reinforcement learning-based combustion optimization system for coal-fired boiler," *Engineering Applications of Artificial Intelligence*, vol. 74, pp. 303–311, 2018.

[41] J. Li and L. Zheng, "Deepwave: Deep learning based real-time water wave simulation," 2018.

[42] Z. Fang, "A high-efficient hybrid physics-informed neural networks based on convolutional neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5514–5526, 2021.

[43] R. G. Nascimento and F. A. Viana, "Fleet prognosis with physics-informed recurrent neural networks," *arXiv preprint arXiv:1901.05512*, 2019.

[44] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang, "Physics-informed machine learning for reduced-order modeling of nonlinear problems," *Journal of computational physics*, vol. 446, p. 110666, 2021.

[45] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.

[46] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, 2020.

[47] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[48] J. Pu, J. Li, and Y. Chen, "Solving localized wave solutions of the derivative nonlinear schrödinger equation using an improved pinn method," *Nonlinear Dynamics*, vol. 105, pp. 1723–1739, 2021.

[49] C. L. Wight and J. Zhao, "Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks," *arXiv preprint arXiv:2007.04542*, 2020.

[50] D. Klyuchinskiy, N. Novikov, and M. Shishlenin, "A modification of gradient descent method for solving coefficient inverse problem for acoustics equations," *Computation*, vol. 8, no. 3, p. 73, 2020.
[51] J. Li and A. M. Tartakovsky, "Physics-informed karhunen-loéve and neural network approximations for solving inverse differential equation problems," *Journal of Computational Physics*, vol. 462, p. 111230, 2022.
[52] A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, "Physics-informed neural networks for inverse problems in supersonic flows," *Journal of Computational Physics*, vol. 466, p. 111402, 2022.
[53] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes," *IMA Journal of Numerical Analysis*, vol. 42, no. 2, pp. 981–1022, 2022.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

The Appendix includes content that could not be explained in detail within the main text of the thesis due to space limitations.

APPENDIX A
DETAILED RESULTS IN NUMERICAL EXPERIMENT

In this section, we present more numerical experiments and some detailed results that could not represented within the main text of the thesis due to space limitations.

A. Simulation and Parametric Estimation of the Single PDE system

1) Single PDE System: To observe the performance of the network changes through the hyper-parameters and network structure, we define a Single PDE system with parameters in constant numbers as

$$\frac{\partial u}{\partial t} = 3.5\frac{\partial^2 u}{\partial x^2} + 6.7\frac{\partial u}{\partial x} + 8.5u$$

The initial condition is defined as

$$u = sin(\pi t)$$

The boundary condition is defined as

$$u(0,t) = 0, u(2,t) = 0$$

The temporal interval is [0,1], and the spatial interval is [0,2]. The dataset is made by the PDE solver introduced in the previous subsection.

To analyze the influence of network structure, hyperparameters such as learning rate, and the number of training data, we use the control variates method and conduct a series of tests. The results are presented as follows.

2) PINN with Different Number of Training Data: Firstly, the amount of training data plays a critical role in determining the performance of machine learning models. To investigate the impact of training data size on our neural network's performance, we fixed the learning rate at 0.001 and employed a network architecture comprising 3 layers, each containing 32 neurons. To disentangle the effects of data points N_d from those of physics points N_p, we maintained a constant number of physics points at $N_p = 5000$ and varied the number of data points N_d. The objective was to observe how variations in N_d influence the model's predictive accuracy and convergence behavior. The validation loss and parametric error are shown in Fig.9(a) and (b) after 30000 epochs.

In our experiments, we varied the number of data points N_d to train our model, selecting values of 500, 1000, 2000, 2500, 3000, 3500, 4000, 4500, and 5000. The purpose of this variation was to determine the optimal amount of data needed to achieve accurate predictions and parameter estimations while maintaining computational efficiency.

From Fig.9 (a), we observed that increasing the number of data points does not consistently lead to a lower validation loss. This indicates that merely increasing the data volume is not always beneficial for model performance. Instead, it is crucial to identify a suitable number of data points that optimize the validation loss. The validation loss tends to stabilize beyond a certain data point threshold, suggesting that an optimal amount of data exists beyond which additional data points provide diminishing returns in performance.

Fig.9 (b) highlights the relationship between the number of data points and parametric error. Insufficient data points result in a higher parametric error, meaning that the model struggles to accurately identify the PDE parameters with limited data. Notably, when the number of data points exceeds 2000, the parametric error stabilizes and changes only slightly. This suggests that 2000 data points represent a threshold for sufficient data, beyond which additional data points contribute minimally to improving parameter estimation accuracy. Thus, having at least 2000 data points is important for accurately identifying the PDE model.

Next, we maintained the number of data points at $N_d = 4500$ and varied the number of physics points to 500, 1000, 2000, 2500, 3000, 3500, 4000, 4500, and 5000. The validation loss and parametric error after 30,000 epochs are illustrated in Fig.9 (c) and (d).

From Fig.9 (c), it is evident that when the number of physics points exceeds 1000, the validation loss remains largely unchanged. This suggests that the influence of physics points on validation loss becomes negligible beyond this threshold.

From Fig.9 (d) shows that while the physics points have a slight influence on the parametric error, the impact is more pronounced than on the validation loss. The parametric error does not reach its minimum when the number of physics points is maximized, indicating that a suitable number of physics points is necessary for optimal performance.

Balancing the quantity of training data with computational efficiency is essential. In our study, we determined that using $N_d = 4500$ and $N_p = 3500$, strikes an optimal balance, providing accurate performance while maintaining training costs. This selection ensures that the model avoids underfitting due to insufficient data and overfitting, and also avoids the excessive computational requirements without significant performance improvement. The detailed convergence process is represented in Appendix.

3) PINN with Different Learning Rate: The learning rate is an important hyperparameter in the training of neural networks, influencing the speed and quality of convergence during the network learning process. To investigate its effect on the performance of our neural network model, we conducted a series of experiments. We fixed the number of data points at 4500 and the physics points at 3500, employed three neural networks, each comprising three layers with 32 neurons per layer, and varied the learning rate to observe its impact on parameter convergence, validation loss, and parametric error. The results of these experiments are illustrated in Fig.10.

From the experimental results, it is evident that the performance of the network varies significantly with different learning rates. The learning rates tested were 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, and 0.0001. The figure demonstrates distinct differences in convergence behavior and evaluation metrics across these learning rates.

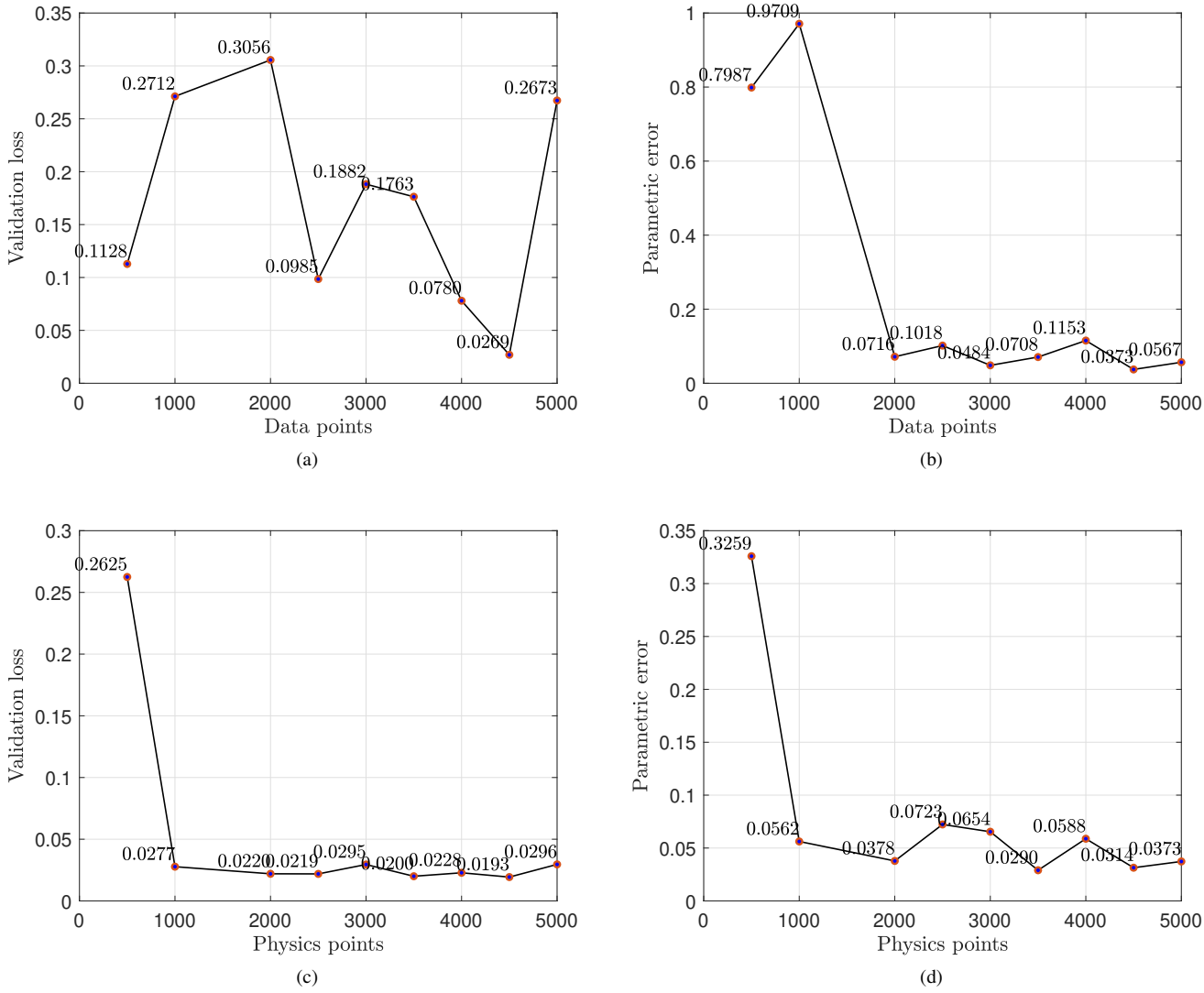


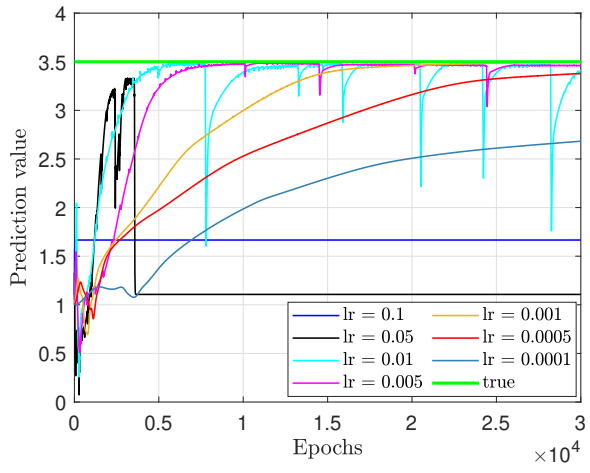
Fig. 9: (a) represents the validation loss with different data points. (b) represents the parametric error with different data points.(c) represents the validation loss with different physics points. (d) represents the parametric error with different physics points.

Fig.10 (a) (b) and (c) illustrate the convergence of the trainable parameters A, B, and C over 30,000 epochs.When the learning rate exceeds 0.05, the parameters fail to converge to their true values and instead remain constant during training, possibly due to being trapped in local optimal solutions. In the range of learning rates from 0.01 to 0.001, although the parameters converge towards the true values, they exhibit significant fluctuations that decrease as the learning rate decreases. Learning rates below 0.001 result in smoother convergence curves albeit at a slower pace, suggesting that a learning rate of 0.001 strikes an optimal balance between achieving accurate parameter estimation and maintaining stable training dynamics.

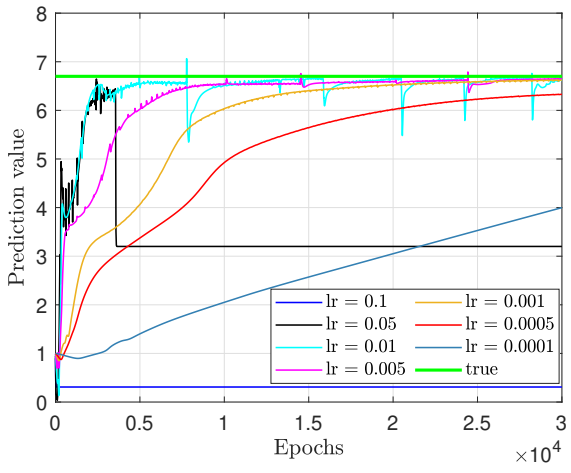
The Fig.10 (d) displays the validation loss across different learning rates. Learning rates above 0.001 show increased validation loss fluctuations, which become more pronounced with higher rates. This observation indicates that excessively high learning rates destabilize the training process, causing parameter values to fluctuate significantly between iterations rather than steadily approaching optimal values.

Fig.10 (e) represents the validation loss after 30,000 epochs are visualized. Notably, learning rates of 0.1 and 0.05 yield validation losses of 100.88 and 136.92, respectively, values too large to display on the graph. The minimum validation loss of 0.02 occurs at a learning rate of 0.001, indicating optimal model performance at this rate. And Fig.10 (f) illustrates the parametric error for different learning rates after 30,000 epochs. Consistently, the parametric error is minimized at a learning rate of 0.001.

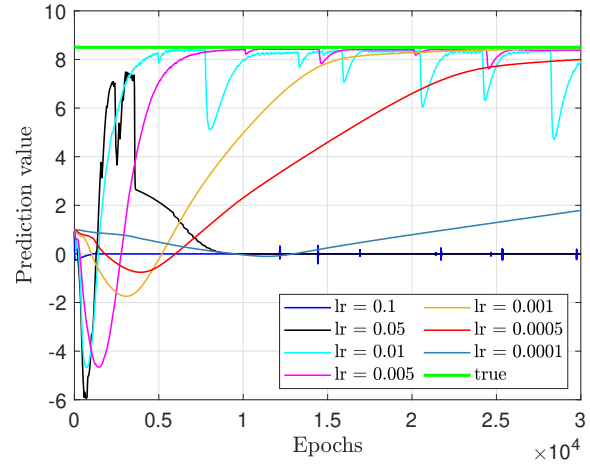
From the figure, it is evident that the learning rate significantly impacts the network's performance. When the learning rate is excessively large, such as 0.1, the network fails



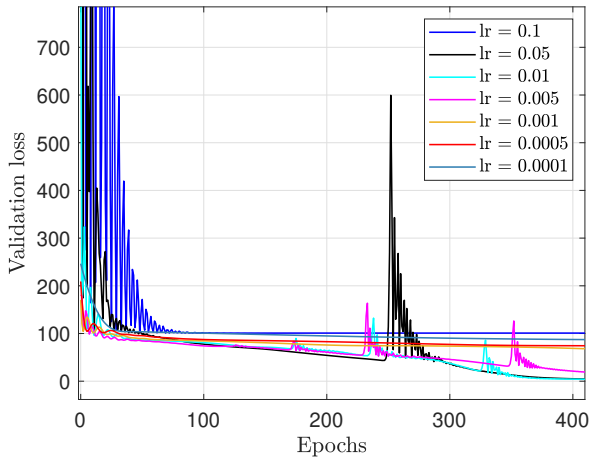
(a)



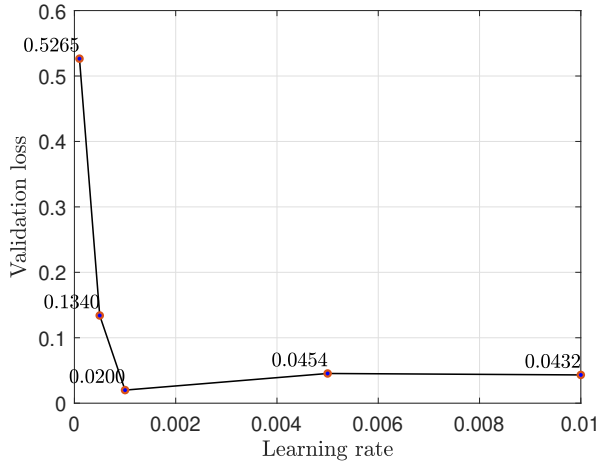
(b)



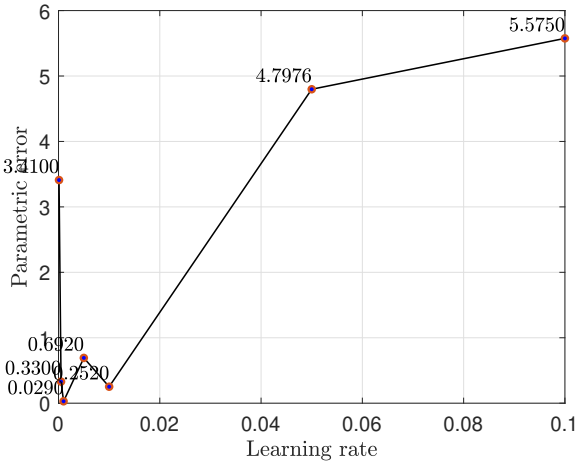
(c)



(d)



(e)



(f)

Fig. 10: (a) (b) and (c) represents the convergence of the prediction value of A, B, and C with different learning rates. (d) represents the validation loss of the network with different data points from the first 400 epochs. (e) represents the validation loss after 30000 epochs. (f) represents the parametric error after 30000 epochs

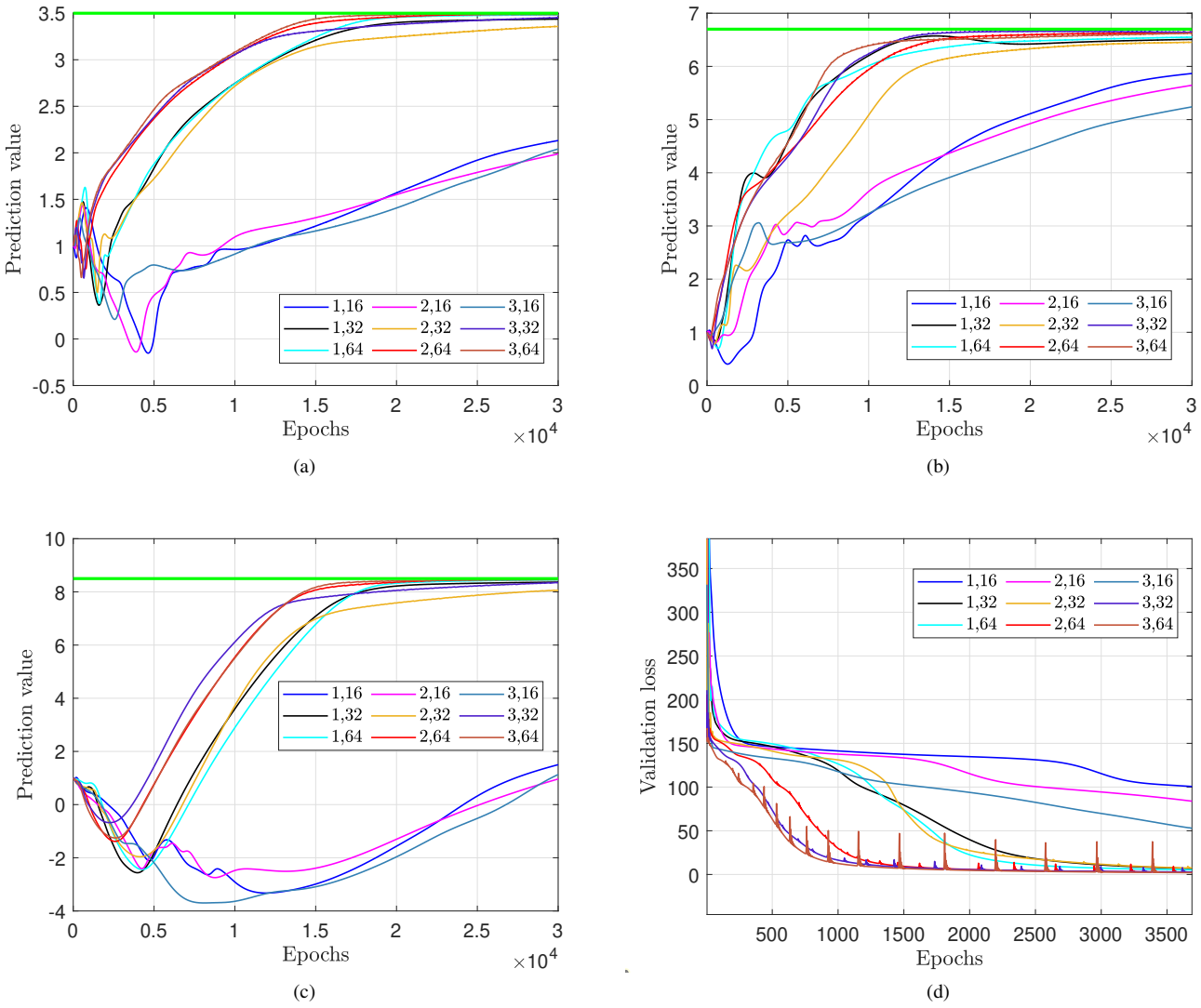


Fig. 11: (a) (b) and (c) represents the prediction value of A, B, and C changes through epochs with different network layers and neurons. (d) represents the validation loss of the network with different network structures

to accurately estimate parameter values, resulting in non-convergent training parameters for A B, and C. Additionally, the validation loss exhibits substantial fluctuations, likely due to the large learning rate causing the network to overshoot the optimal solution in each epoch.

In summary, our experiments underscore the critical role of the learning rate in neural network training. A learning rate of 0.001 was identified as optimal, minimizing both validation loss fluctuations and parametric error.

Based on these observations, it is clear that a learning rate of 0.001 strikes an appropriate balance. It enables efficient convergence without the instability associated with larger learning rates or the inefficiency of smaller ones. Thus, we identify 0.001 as the optimal learning rate for our neural network model under the given experimental conditions.

4) PINN with Different Network Structure: The complexity of a neural network is a critical factor influencing its overall performance. To investigate the impact of network complexity on structural efficacy, we constructed and trained networks with varying configurations of layers and neurons per layer. For consistency, the learning rate was fixed at 0.001, and the training process involved 3000 data points alongside 3000 physics-informed points. This experimental setup aimed to elucidate the effects of network architecture on parameter estimation accuracy and validation loss. The results of these experiments are illustrated in Fig.10.

In Fig.11, we present the estimation values of key parameters and the corresponding validation losses for networks configured with 1 to 3 layers, each layer containing 16, 32, or 64 neurons. The data indicates a pronounced influence of neuron quantity per layer on network performance. Specifically, networks with 1, 2, or 3 layers, each consisting of 16 neurons, demonstrated suboptimal results in terms of accuracy and convergence rates. Conversely, increasing the number of layers

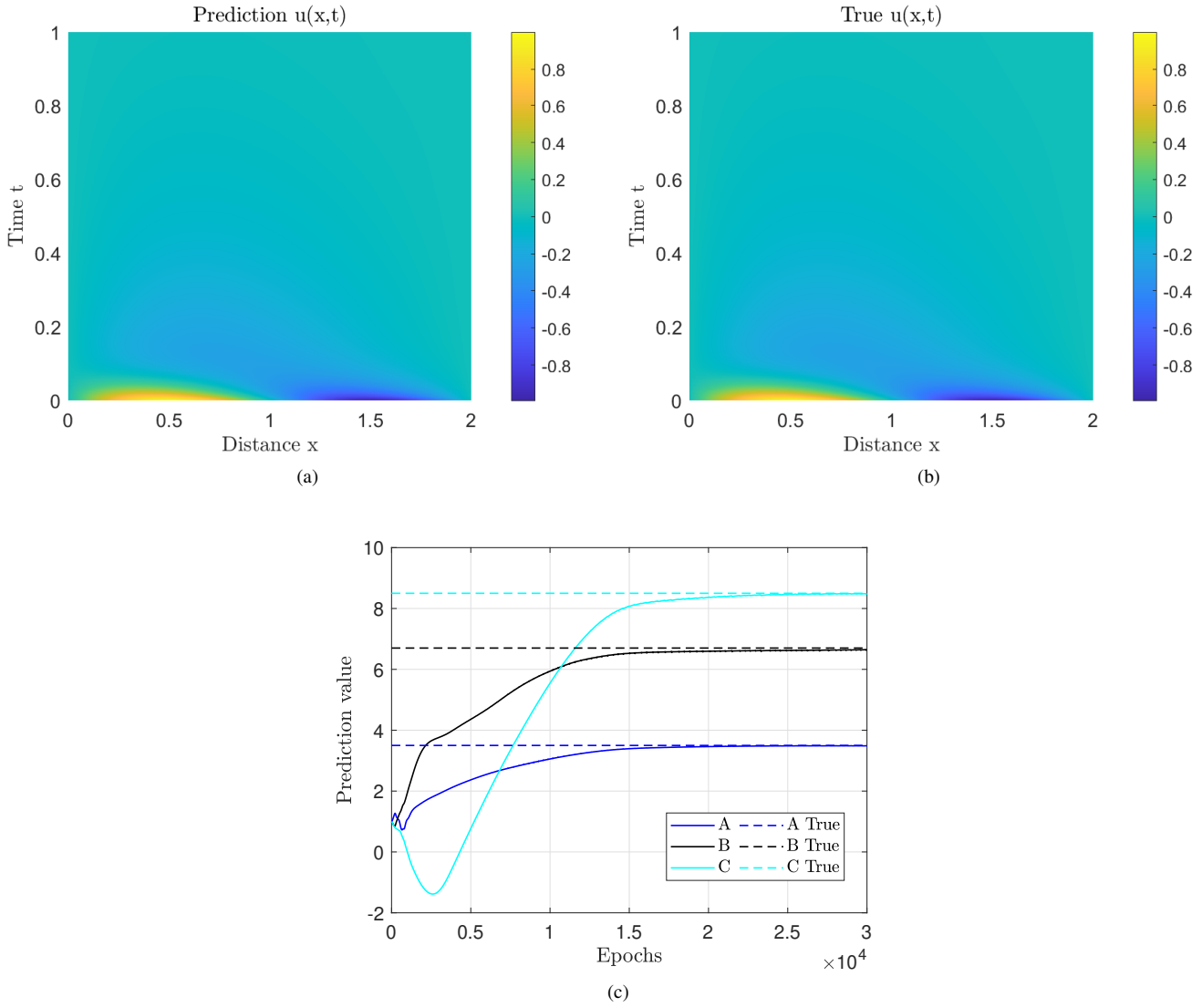


Fig. 12: (a) represents the prediction of the solution of the PDE system. (b) represents the true solution of the PDE system from the dataset.(c) represents the estimation of the parameters.

generally facilitated faster convergence, while a higher neuron count per layer substantially improved accuracy. Networks with minimal neuron counts (i.e., 16 neurons per layer) exhibited considerable variance and instability in parameter predictions, as shown in Fig.11(a) to (c). In contrast, networks with 64 neurons per layer showed markedly better performance, achieving higher accuracy and more stable convergence.

Furthermore, Fig.11(d) elucidates the relationship between network architecture and validation loss. Networks with deeper architectures (i.e., more layers) generally reached lower validation losses more swiftly than their shallower counterparts, indicating faster learning rates and better generalization capabilities. However, it is crucial to note that simply increasing the number of layers is not a panacea; the balance between depth and neuron count per layer is pivotal. Excessive complexity can lead to overfitting, whereas insufficient complexity may result in underfitting.

These findings show the importance of carefully tuning both the depth and width of neural networks to optimize performance. The interplay between the number of layers and neurons per layer significantly affects both the convergence behavior and the ultimate accuracy of the network. Thus, selecting an appropriate network architecture is paramount for achieving optimal results in parameter estimation and minimizing validation loss. Here, considering the training cost, accuracy, and convergence, we choose 2 layers with 64 neurons in each as an optimal structure.

5) Simulation and Parametric Estimation Results: Finally, we obtain the predicted output $u_{prediction}$ and the estimated parameters of the PDE system. These results are derived using a learning rate of 0.001, with 4500 data points and 3500 physics points, and a neural network architecture consisting of 2 layers with 64 neurons per layer. The estimation of the parameters and comparison between the predicted solution and

the true solution of the PDE system is depicted in Fig.12.

From the figure, the prediction results and the true solution are almost the same, which means the good performance of our network in the single PDE system.

In this context, we applied the optimal network architecture and hyperparameters determined from prior experiments to obtain the results. Specifically, we employed a learning rate of 0.001, and a neural network comprising three layers, each with 32 neurons. For training the model, we selected 3000 data points and 3000 physics points. The Adam optimizer was utilized to facilitate the training process.

The results, which include the estimated parameters and the predicted solution of the PDE system, are illustrated in Fig.4.