Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 32(7): 4291–4320.
DOI: 10.3934/era.2024193
Received: 28 May 2024
Revised: 26 June 2024
Accepted: 01 July 2024
Published: 08 July 2024

Research article

Deep multi-input and multi-output operator networks method for optimal control of PDEs

Jinjun Yong[1,2], **Xianbing Luo**[1,*] **and Shuyu Sun**[3]

[1] School of Mathematics and Statistics, Guizhou University, Guiyang 550025, China

[2] School of Mathematics And Big Data, Guizhou Education University, Guiyang 550018, China

[3] Computational Transport Phenomena Laboratory, Division of Physical Science and Engineering, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

* **Correspondence:** Email: xbluo1@gzu.edu.cn.

Abstract: Deep operator networks is a popular machine learning approach. Some problems require multiple inputs and outputs. In this work, a multi-input and multi-output operator neural network (MIMOONet) for solving optimal control problems was proposed. To improve the accuracy of the numerical solution, a physics-informed MIMOONet was also proposed. To test the performance of the MIMOONet and the physics-informed MIMOONet, three examples, including elliptic (linear and semi-linear) and parabolic problems, were presented. The numerical results show that both methods are effective in solving these types of problems, and the physics-informed MIMOONet achieves higher accuracy due to its incorporation of physical laws.

Keywords: operator neural networks; multi-input; multi-output; physics-informed; PDE optimal control

1. Introduction

The optimal control problem has been successfully applied in various fields, such as heat transfer phenomena [1], finance [2], image processing [3], shape optimization [4,5], aerodynamics [6,7], crystal growth [8], and drug delivery [9]. To solve partial differential equation constrained (PDE-constrained) optimal control problems, numerous numerical methods have been developed, including finite element methods, finite differences methods, finite volume methods, spectral methods, and mesh-less methods (see, e.g., [10–13]). Despite their effectiveness, the optimal control problem remains challenging to solve, particularly when the problem is nonlinear. Recently, deep learning has emerged as a popular method for solving partial differential equations, especially nonlinear ones.

Deep learning methods for solving PDEs have received significant attention, including physics-informed neural networks (PINN) [14, 15], deep Galerkin method [16], deep Ritz method [17], deep Nitsche method [18], and deep operator networks (DeepONets) [19, 20]. PINNs can be used to solve specific PDEs with boundary conditions and loading terms, but they require expensive optimization computation cost during inference. Therefore, the PDEs with operating conditions and real-time inference cannot be solved by PINNs. The neural operator full-fills well. Various versions of operator networks have been published, e.g., graph neural operator networks [21], Fourier neural operator (FNO) [22], physics-informed neural operators (PINO) [23], and deep multiple input operator network (DeepMIONet) [24].

Neural networks have several advantages over traditional numerical solvers, including being mesh-free and easier to deal with complex geometric regions. Simulating control problems that involve these complex geometric regions using traditional numerical methods often requires high-quality grids and extensive preprocessing before simulation. By contrast, researchers are interested in using neural networks to replace traditional numerical methods. To solve optimal control problems with PINNs, methodologies and guidelines have been proposed in previous works [25, 26]. Barry-Straume et al. used a two-stage framework to solve PDE-constrained optimization problems [27]. Wang et al. used physics-informed deep operator networks (DeepONets) to learn the solution operator of parametric PDEs, building a surrogate for solving PDE-constrained optimization problems [28]. In summary, deep learning approaches, such as PINNs and DeepONets, have shown promise for solving PDE-constrained optimal control problems, providing an efficient solution without requiring extensive preprocessing or expensive optimization during inference. Future work could explore further improvements to deep learning methods and investigate their application in new fields.

In this paper, to solve PDE-constrained optimal control problems with available data, we introduce a MIMOONet. In the context of the PDE optimal problem, the governing PDE is fully known, and the objective is to determine a control variable that minimizes the cost function. Initially, the PDE-constrained optimal control problem is reformulated into a PDE system using the adjoint method. Subsequently, the PDE system is tackled by using MIMOONets. Additionally, we examine a physical system described by PDEs and propose physics-informed MIMOONets for addressing the PDE-constrained optimal control problem. Overall, this method (MIMOONet) has the following advantages:

- MIMOONet can solve PDE-constraint problems governed by different types of PDE.
- MIMOONet can easily approximate nonlinear optimal control problems.
- Compared with traditional numerical methods, its prediction speed is faster.

The remainder of this paper is organized as follows. In Section 2, we introduce the adjoint state method of solving PDE-constrained optimization problem and optimality system, and provide the framework of MIMOONets, physics-informed MIMOONets, and the detailed method of our main technical contribution. In Section 3, we give the deep learning framework of elliptic and parabolic constrained optimal control problem, and present the numerical results to assess the performance of the proposed MIMOONets and physics-informed MIMOONets. Finally, Section 4 summarizes the results, potential pitfalls, and shortcomings, and details the groundwork for future directions.

2. Methods

Let U, S, V be Banach spaces. We consider the following PDE-constrained optimization problem:

$$\begin{cases} u^*, v^* = \arg\min\limits_{u\in S, v\in U} J(u, v), \\ \text{subject to} \quad \mathcal{F}(u, v) = 0, \end{cases} \tag{2.1}$$

where $J : S \times U \to \mathbb{R}$ is a cost function, $\mathcal{F} : S \times U \to V$ is a system of PDEs subject to initial and boundary conditions, u and v are the state variable and control variable, respectively. Assume that the problem (2.1) has unique solutions u and v. In the subsequent sections, the MIMOONets method is presented under this assumption.

2.1. Optimality system

In this subsection, we transform the PDE-constrained optimization problem into an optimality system.

The PDE-constrained optimization problem, which involves optimizing an objective function subject to a set of partial differential equations (PDEs), can be transformed into an optimality system. The resulting optimality system consists of two sets of equations: the state equations and the adjoint equations. These equations are coupled and must be solved simultaneously to obtain the optimal solution to the original PDE-constrained optimization problem.

Consider the following problem [29],

$$\min_{u\in S, v\in U} J(u, v), \tag{2.2}$$

$$\text{subject to} \quad \begin{cases} \mathcal{F}[u(x,t); v(x,t)] = 0, & x \in \Omega, t \in [0, T], \\ \mathcal{B}[u(x,t)] = 0, & x \in \partial\Omega, t \in [0, T], \\ \mathcal{I}[u(x,0)] = 0, & x \in \Omega, \end{cases} \tag{2.3}$$

where x and t denote space and time variables, respectively, the domain $\Omega \subseteq \mathbb{R}^d$, $\partial\Omega$ is the boundary of the domain Ω, and $\mathcal{B}$ and $\mathcal{I}$ are boundary conditions and initial condition, respectively. We construct the Lagrangian function for the problems (2.2) and (2.3) as follows:

$$\mathcal{L}(u, v, p_1, p_2, p_3) = J(u, v) - \int_0^T \int_\Omega p_1 \mathcal{F}(u, v) dx dt - \int_0^T \int_{\partial\Omega} p_2 \mathcal{B}(u) ds dt - \int_\Omega p_3 \mathcal{I}(u) dx. \tag{2.4}$$

Here, p_1, p_2, and p_3 are Lagrange multiplier functions defined on $\Omega \times [0, T]$, $\partial\Omega \times [0, T]$ and $\partial\Omega \times 0$, respectively. According to the Lagrange principle, we seek the pair (u^*, v^*) and the Lagrange multipliers or adjoint field $\mathbf{p} = (p_1, p_2, p_3)$ to satisfy the optimality conditions. Therefore, the problems (2.2) and (2.3) are equivalent to the following unconstrained problem

$$(u^*, v^*, \mathbf{p}^*) = \arg\min_{u\in S, v\in U, \mathbf{p}} \mathcal{L}(u, v, \mathbf{p}). \tag{2.5}$$

Then, the directional derivative of $\mathcal{L}$ with respect to u disappears at the optimal point, that is

$$D_u \mathcal{L}(u^*, v^*, \mathbf{p}^*)\delta u = \lim_{\varepsilon\to 0} \frac{\mathcal{L}(u^* + \varepsilon\delta u, v^*, \mathbf{p}^*) - \mathcal{L}(u^*, v^*, \mathbf{p}^*)}{\varepsilon} = 0, \quad \forall \delta u \in S. \tag{2.6}$$

For the control variable v and Lagrange multipliers $\mathbf{p}$, we have

$$D_v\mathcal{L}(u^*, v^*, \mathbf{p}^*)\delta v = 0, \quad \forall \delta v \in U, \tag{2.7}$$

and

$$D_{\mathbf{p}}\mathcal{L}(u^*, v^*, \mathbf{p}^*)\delta \mathbf{p} = 0, \quad \forall \delta \mathbf{p}. \tag{2.8}$$

Therefore, the problems (2.2) and (2.3) can be written in the following way:

$$\begin{cases} D_u\mathcal{L}(u, v, \mathbf{p})\delta u = 0, \\ D_v\mathcal{L}(u, v, \mathbf{p})\delta v = 0, \\ D_{\mathbf{p}}\mathcal{L}(u, v, \mathbf{p})\delta \mathbf{p} = 0. \end{cases} \tag{2.9}$$

Once the optimality system has been derived, it can be solved by using neural networks. Next, we introduce the MIMOONet and the physics-informed MIMOONet methods for solving the optimality system.

2.2. Multiple-input operators networks

DeepONets is a learning framework proposed by Lu et al. [19], which enables the learning of abstract nonlinear operators in infinite-dimensional function spaces. It is inspired by the universal approximation theorem of operators [30]. The DeepONets network comprises two main components: the trunk network and the branch network.

The trunk network provides the basis functions for the output function by encoding information related to the space-time coordinates. It takes as input the space-time coordinates and any other relevant physical parameters and produces a set of basis functions for the function space. These basis functions serve as a representation of the output function and are used in the computation of the final output.

The branch network encodes the input function to provide the coefficients at fixed sensor points. Given an input function, which may be a solution to a PDE or a data-driven function, the branch network maps it to a set of coefficients that correspond to specific sensor points in the domain. These coefficients are then combined with the basis functions from the trunk network to produce the final output function.

Theorem 2.1. *Suppose that X is a Banach space, $K_1 \subset X$, $K_2 \subset R^d$ are two compact sets in X and R^d, respectively. Let V be a compact set in $C(K_1)$, $G : V \to C(K_2)$ a nonlinear continuous operator, σ a continuous non-polynomial function. Then, for $\forall \varepsilon > 0$, there exist n, p, $m \in \mathbb{N}$, constants c_i^k, a_{ij}^k, θ_i^k, $\zeta_k \in R$, $w_k \subset R^d$, $x_j \in K_1$, $i = 1, 2, \cdots, n$, $k = 1, \cdots, p$, $j = 1, \cdots, m$, such that*

$$| G(u)(y) - \sum_{k=1}^{p} \underbrace{\sum_{i=1}^{n} c_i^k \sigma(\sum_{j=1}^{m} a_{ij}^k u(x_j) + \theta_i^k)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk} | < \varepsilon, \tag{2.10}$$

for any $u \in V$ and $y \in K_2$.

DeepONets are designed to learn abstract nonlinear operators in infinite-dimensional function spaces from a single input function defined on a Banach space, while real-world problems often involve multiple input functions. To address this issue, the DeepMIONet was proposed in [24], which is defined through the tensor product of Banach spaces.

In DeepMIONets, only the case of two input functions u and v are considered, which are represented by their respective branch networks. The trunk network provides basis functions that span the function space, and the outputs of the branch networks are combined with the basis functions by using tensor products to produce the final output function. Specifically, the output of DeepMIONet $G_\theta(u, v)$ can be written as:

$$G_\theta(u, v) = \sum_{i=1}^{p} b_i^u b_i^v tr_i, \tag{2.11}$$

where b_i^u and b_i^v denote the i-th output of the branch networks corresponding to the input functions represented by u and v, respectively. And tr_i is the i-th output of the trunk network.

The architecture of DeepMIONets includes two separate branch networks and a shared trunk network (as shown in Figure 1), which provide basis functions spanning the function space. The branch networks encode the input functions and provide coefficients at fixed sensor points, while the trunk network provides basis functions spanning the function space. The outputs of the branch networks and the trunk network are combined using tensor products to produce the final output function. Specifically, the outputs of the branch networks are the coefficients of the input functions, while the trunk network provides a set of basis functions that are combined with the outputs of the branch networks using tensor products to obtain the final output function.

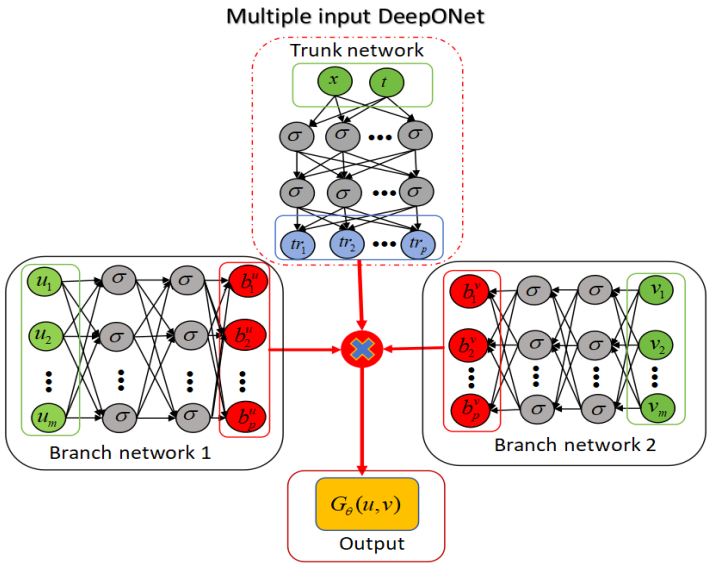


Figure 1. Architectures of MIONet for $G_\theta(u, v)(x, t)$: the branch network 1 takes u as input functional [employs a fully connected neural network (FNN) to take as input the values at m sensor], the branch network 2 takes v as input functional (employs an FNN to take as input the values at n sensor), and computes coefficients of the solution for the coordinates, which are inputs of the trunk network (employs a FNN).

2.3. Multi-input and multi-output operators networks

Here, we focus on using neural operator networks to solve systems of PDEs.

Although DeepONets and DeepMIONet can be used to solve a single PDE, they cannot do it for a system of PDEs, in which at least two solution operators must be generated and two output operators are needed for a network. To solve this problem, we propose MIMOONets, which also can be used to solve systems of PDEs. The MIMOONets framework is composed of a trunk network and multiple branch networks. The trunk network provides the basis functions of the solution operators, while the branch networks provide additional groups of coefficients of the solution operators at fixed sensor points.

We consider the following problem that is a system of PDEs in domain $D \subseteq \mathbb{R}^d$ (d is the dimension of space),

$$\begin{cases} \mathcal{L}_i[u_1(x), u_2(x), \cdots, u_n(x)] = f_i(x), & x \in D, i = 1, 2, \cdots, n, \\ \mathcal{B}_i[u_i(x)] = \varphi_i(x), & x \in \partial D, i = 1, 2, \cdots, n, \end{cases}$$

where u_i and f_i are functions, $\mathcal{L}_i$ is the differential operator, φ_i is the boundary condition of u_i. Let G^i be the solution of the operator with input functions f_i and φ_i s.t.

$$\mathcal{L}_i[G^1, G^2, \cdots, G^i] = f_i, i = 1, 2, \cdots, n,$$

and

$$\mathcal{B}_i \circ G^i = \varphi_i, on \ \ \partial D, i = 1, 2, \cdots, n.$$

This means that $G^i(f_1, f_2, \cdots, f_n, \varphi_1, \varphi_2, \cdots, \varphi_n)(y)$ is the corresponding output function. According to the results in [24] and [30], the solution u_i can be expressed as

$$u_i = G^i(f_1, f_2, \cdots, f_n, \varphi_1, \varphi_2, \cdots, \varphi_n)(y). \tag{2.12}$$

Then, the solution of problem (2.12) can be learned by using MIMOONets, which is defined through the tensor product of Banach spaces.

Here, we only give the framework of two-input and two-output operators networks (see Figure 2). The solution operator networks can be described by

$$\begin{cases} G^1_\theta(f, g) = \sum\limits_{k=1}^{p} b1^f_k b1^g_k tr_k, \\ G^2_\theta(f, g) = \sum\limits_{k=1}^{p} b2^f_k b2^g_k tr_k, \end{cases} \tag{2.13}$$

where the definitions of $b1^f_k$, $b2^f_k$, $b1^g_k$, $b2^g_k$ and tr_k are similar to (2.11). To reduce the generalized error, we may add a bias $b^1_0, b^2_0 \in \mathbb{R}$ in the last stage:

$$\begin{cases} G^1_\theta(f, g) = \sum\limits_{k=1}^{p} b1^f_k b1^g_k tr_k + b^1_0, \\ G^2_\theta(f, g) = \sum\limits_{k=1}^{p} b2^f_k b2^g_k tr_k + b^2_0. \end{cases} \tag{2.14}$$

Let $G^i : C(D) \longrightarrow L^2(D)$ be a Borel measurable mapping with $G^i \in L^2(\mu)$. Then, for any $\varepsilon > 0$, there exists an operator network $G^i_\theta : C(D) \longrightarrow L^2(D)$, such that

$$\|G^i - G^i_\theta\|_{L^2(\mu)} = \left(\int_{C(D)} \|G^i - G^i_\theta\|_{L^2}(\mu) d\mu(f, g) \right)^{\frac{1}{2}} < \varepsilon,$$

here, μ is a probability measure on $C(D)$.

When we have some dataset of the pair solution $\{u(x_k), v(x_k)\}_1^N$, the solution can be learned by MIMOONets. The corresponding loss function can be formulated as follows

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\left(|\, u(x_k) - G_\theta^1(f,g)(x_k)\,|^2 + |\, v(x_k) - G_\theta^2(f,g)(x_k)\,|^2\right). \tag{2.15}$$

Now we can learn parameter θ by minimizing the loss function (2.15) and using the stochastic gradient descent method.

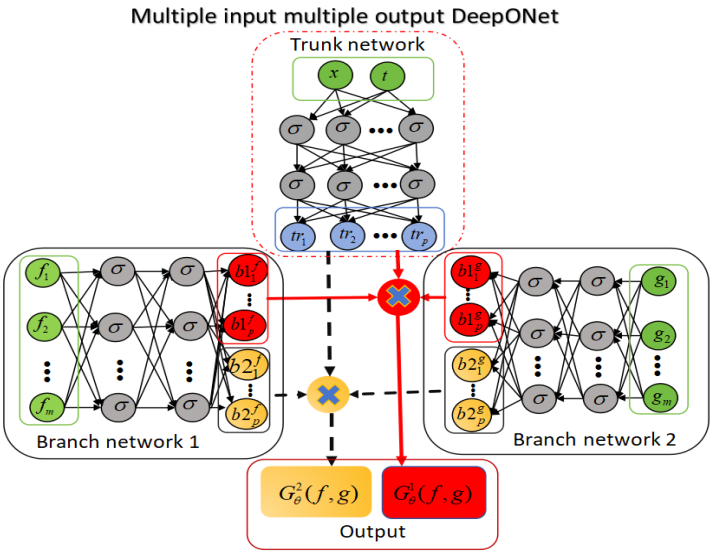


Figure 2. Architectures of MIMOONets for $G_\theta(f,g)(x)$: the branch network 1 takes f as input functional (employs a FNN to take as input the values at m sensor), the branch network 2 takes g as input functional (employs an FNN to take as input the values at n sensor), and computes two groups coefficients of the solution for the coordinates, which are inputs of the trunk network (employs a FNN).

2.4. Physics-informed MIMOONets

A large amount of paired datasets is required to solve PDE systems by using MIMOONets. However, data acquisition is expensive in many engineering applications and physical systems. Under the condition of sparse data, it becomes important to introduce physics-informed neural networks to train MIMOONets by integrating known differential equations with label data in the loss function. We use automatic differentiation of the outputs of MIMOONets with respect to their input coordinates and adopt an appropriate regularization mechanism to ensure that the target output functions satisfy the PDE constraints.

For simplicity, we consider the following problem (without causing confusion, we still use the

preceding symbols):

$$\begin{cases} \mathcal{L}_1[u(x); v(x)] = f(x), & x \in D, \\ \mathcal{L}_2[u(x); v(x)] = g(x), & x \in D, \\ \mathcal{B}_1[u(x)] = \varphi(x), & x \in \partial D, \\ \mathcal{B}_2[v(x)] = \psi(x), & x \in \partial D. \end{cases} \tag{2.16}$$

The solutions u and v can be expressed as

$$\begin{cases} u = G^1(f, g, \varphi, \psi)(y), \\ v = G^2(f, g, \varphi, \psi)(y). \end{cases} \tag{2.17}$$

For the problem (2.16), the loss function of a physics-informed MIMOONets is defined as follows

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \mathcal{L}_{physics}(\theta), \tag{2.18}$$

where

$$\begin{cases} \mathcal{L}_{data}(\theta) = \frac{1}{N} \sum\limits_{k=1}^{N} \left(\mid u(x_k) - G^1_\theta(f, g, \varphi, \psi)(x_k) \mid^2 + \mid v(x_k) - G^2_\theta(f, g, \varphi, \psi)(x_k) \mid^2 \right), \\ \mathcal{L}_{physics}(\theta) = \mathcal{L}_{pde1}(\theta) + \mathcal{L}_{pde2}(\theta) + \mathcal{L}_{BC1}(\theta) + \mathcal{L}_{BC2}(\theta), \end{cases} \tag{2.19}$$

and

$$\begin{cases} \mathcal{L}_{pde1}(\theta) = \frac{1}{N_f} \sum\limits_{k=1}^{N_f} \mid \mathcal{L}_1[G^1_\theta(f, g, \varphi, \psi)(x_k); G^2_\theta(f, g, \varphi, \psi)(x_k)] - f(x_k) \mid^2, \\ \mathcal{L}_{pde2}(\theta) = \frac{1}{N_g} \sum\limits_{k=1}^{N_g} \mid \mathcal{L}_2[G^1_\theta(f, g, \varphi, \psi)(x_k); G^2_\theta(f, g, \varphi, \psi)(x_k)] - g(x_k) \mid^2, \\ \mathcal{L}_{BC1}(\theta) = \frac{1}{N_\varphi} \sum\limits_{k=1}^{N_\varphi} \mid \mathcal{B}_1[G^1_\theta(f, g, \varphi, \psi)(x_k)] - \varphi(x_k) \mid^2, \\ \mathcal{L}_{BC2}(\theta) = \frac{1}{N_\psi} \sum\limits_{k=1}^{N_\psi} \mid \mathcal{B}_2[G^2_\theta(f, g, \varphi, \psi)(x_k)] - \psi(x_k) \mid^2 . \end{cases} \tag{2.20}$$

Here, N is the number of initial data points. N_f and N_g are the number of sample from the computational domain Ω for the PDEs. N_φ and N_ψ are the number of the boundary points for u and v.

The loss function (2.18) is minimized by learning the parameters θ of the deep neural network. Sometimes, in order to improve the accuracy of the numerical solution or increase the convergence rate, we can apply penalty parameters.

2.5. PDE-constrained optimization with physics-informed MIMOONets

For a given PDE-constrained optimal control problem such as (2.2), we use physics-informed MIMOONets to solve the optimization problems. The corresponding steps are given as follows. First, we turn the PDE-constrained optimization problem (2.2) to an optimality system (2.9), which consists of adjoint equation, state equation, and optimality condition; second, we solve the optimality system using physics-informed MIMOONets. The detailed computing framework can be seen in Algorithm 1.

Remark 2.1. *For MIMOONets (no physics-informed), the algorithm is only need to change "***Input***: $\cdots$" to "***Input***: N (the number of initial data of* $\{(x_i, u(x_i), v(x_i))\}$*)".*

Algorithm 1 The steps of physics-informed MIMOONets to approximate optimal control problems

Input: N (the number of initial data of $\{(x_i, u(x_i), v(x_i))\}$), $N_f = N_g$ (the number of internal points), $N_\varphi = N_\psi$ (the number of boundary points), M (maximum number of iterations), λ_i, f, g, φ, ψ, Ω.
Output: G_θ^f (state variable function), G_θ^9 (adjoint state variable function).
1. Take N_f sample points $\{\boldsymbol{x}_i\}$ in Ω, $N_\varphi = N_\psi$ sample points $\{\boldsymbol{x}_j\}$ on $\partial\Omega$.
2. Generate G_θ^f, G_θ^g using Multiple input multiple output Deep ONet.
3. For $k = 1$ to M
4. Calculate $L(\theta)$ according to (2.18),
5. Update neural network parameter θ,
6. Endfor
7. Output G_θ^f, G_θ^g.

3. Main results

In the following demonstrations, to showcase the effectiveness of MIMOONs, some numerical examples of elliptic, semi-linear elliptic, parabolic, and second-order hyperbolic optimal control problems are provided. Data-driven MIMOONs or physics-informed MIMOONs are employed with uniform distribution random sampling on the solution domain during the training process. In the subsequent examples, the relative error of the numerical solution $\boldsymbol{u}$ is calculated by: $\dfrac{\|\boldsymbol{u} - \boldsymbol{u}^*\|}{\|\boldsymbol{u}^*\|}$, where the reference solution $\boldsymbol{u}^*$ is an analytical solution, or finite element approximated solution with 100×100 spatial grid.

3.1. Linear elliptic optimal control problem

We start with an example involving finding an optimal heat source under homogeneous Dirichlet boundary conditions. The model can be represented as follows,

$$\min J(u, v) = \frac{1}{2}\int_\Omega (u - u_d)^2 dx + \frac{\alpha}{2}\int_\Omega v^2 dx, \tag{3.1}$$

$$\text{subject to} \quad \begin{cases} -\Delta u - v = f, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega, \end{cases} \tag{3.2}$$

where Ω is a bounded domain, $u : \Omega \to \mathbb{R}$ is the unknown temperature satisfying (3.2), $u_d : \Omega \to \mathbb{R}$ is the given desired temperature, v is the unknown control function, f is the source term in Ω, $\alpha \geq 0$ is a regularization parameter. Here, we set $\Omega = (0, 1) \times (0, 1)$, $u_d(x, y) = (1 - 10\pi)\sin(\pi x)\sin(\pi y)$, $\lambda = 1$, $f(x, y) = (5 + 2\pi^2)\sin(\pi x)\sin(\pi y)$. When, $u(x, y) = \sin(\pi x)\sin(\pi y)$, $v(x, y) = -5\sin(\pi x)\sin(\pi y)$, the $J(u, v)$ gets the global minimum. We know that the optimal control problems (3.1) and (3.2) can be transformed into the optimality system as follows

$$\begin{cases} -\Delta u - v = f, & \text{in } \Omega, \\ \Delta p + u = u_d, & \text{in } \Omega, \\ \alpha v + p = 0, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega, \\ p = 0, & \text{on } \partial\Omega. \end{cases} \tag{3.3}$$

We use the MIMOONets to solve the PDEs system (3.3). The solution operator G^1 and G^2 of f and u_d can be represented as follows,

$$\begin{cases} G^1_\theta(f, u_d) = \sum\limits_{k=1}^{p} b1^f_k b1^{u_d}_k tr_k, \\ G^2_\theta(f, u_d) = \sum\limits_{k=1}^{p} b2^f_k b2^{u_d}_k tr_k, \end{cases} \tag{3.4}$$

where the branch network 1 and the branch network 2 are two separate 5-layer fully connected neural networks (FNN). Every hidden layer and output layer has 300 neurons. The input layer per network has 100 neurons. The trunk network is 5-layer FNN with 300 neurons per hidden layer and 150 neurons output layer. Relu or Tanh is used as the activation function. The loss function of the deep MIMOONets is denoted by

$$\mathcal{L}_{data}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\left(\mid u(x_k, y_k) - G^1_\theta(f, u_d)(x_k, y_k) \mid^2 + \mid p(x_k, y_k) - G^2_\theta(f, u_d)(x_k, y_k) \mid^2\right). \tag{3.5}$$

We use the same network structure as MIMOONets to consider the physics-informed MIMOONets. The activation function is Tanh. The corresponding loss function can be expressed as follows

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \mathcal{L}_{physics}(\theta), \tag{3.6}$$

where

$$\mathcal{L}_{physics}(\theta) = \mathcal{L}_{pde1}(\theta) + \mathcal{L}_{pde2}(\theta) + \mathcal{L}_{BC1}(\theta) + \mathcal{L}_{BC2}(\theta), \tag{3.7}$$

and

$$\begin{cases} \mathcal{L}_{pde1}(\theta) = \frac{1}{N_f}\sum\limits_{k=1}^{N_f} \mid -\frac{\partial^2 G^1_\theta(f,u_d)(x_k,y_k)}{\partial x^2} - \frac{\partial^2 G^1_\theta(f,u_d)(x_k,y_k)}{\partial y^2} + G^2_\theta(f, u_d)(x_k, y_k) - f(x_k, y_k) \mid^2, \\ \mathcal{L}_{pde2}(\theta) = \frac{1}{N_{u_d}}\sum\limits_{k=1}^{N_{u_d}} \mid \frac{\partial^2 G^2_\theta(f,u_d)(x_k,y_k)}{\partial x^2} + \frac{\partial^2 G^2_\theta(f,u_d)(x_k,y_k)}{\partial y^2} + G^1_\theta(f, u_d)(x_k, y_k) - u_d(x_k, y_k) \mid^2, \\ \mathcal{L}_{BC1}(\theta) = \frac{1}{N_{BC}}\sum\limits_{i=1}^{N_{BC}} \mid G^1_\theta(f, u_d)(x_i, y_i) \mid^2, \\ \mathcal{L}_{BC2}(\theta) = \frac{1}{N_{BC}}\sum\limits_{i=1}^{N_{BC}} \mid G^2_\theta(f, u_d)(x_i, y_i) \mid^2, \end{cases} \tag{3.8}$$

where N is the number of initial data points. N_f and N_{u_d} denotes the number of integration points of f and u_d in the computational domain Ω, respectively. N_{BC} is the number of the boundary points on the $\partial\Omega$.

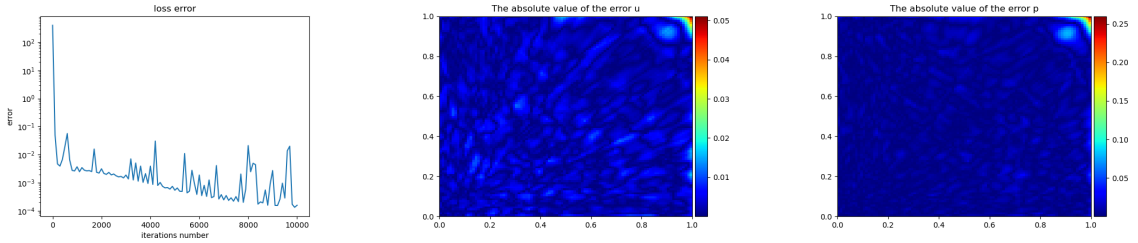
To evaluate the loss, we randomly sample $N = 10,000$ training points $(x_k, y_k) \in \Omega$, $N_f = N_{u_d} = 10,000$ residual training points $(x_k, y_k) \in \Omega$. We select $N_{BC} = 400$ equidistant boundary training points $(x_i, y_i) \in \partial\Omega$.

Table 1. MIMOONets for an optimal control of elliptic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Relu	$(0.62 \pm 0.08)\%$	$(0.39 \pm 0.05)\%$
10,000	Tanh	$(1.00 \pm 0.15)\%$	$(0.98 \pm 0.12)\%$
40,000	Relu	$(0.14 \pm 0.03)\%$	$(0.11 \pm 0.03)\%$
40,000	Tanh	$(0.45 \pm 0.05)\%$	$(0.45 \pm 0.06)\%$

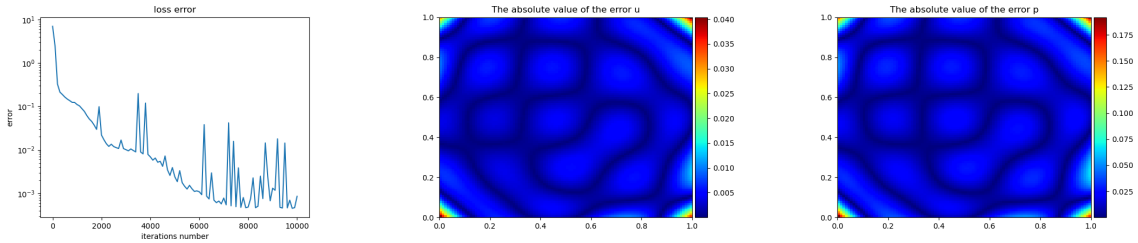
Table 2. Physics-informed MIMOONets for an optimal control of elliptic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Tanh	(0.26 ± 0.04)%	(0.28 ± 0.05)%
40,000	Tanh	(0.13 ± 0.04)%	(0.14 ± 0.05)%



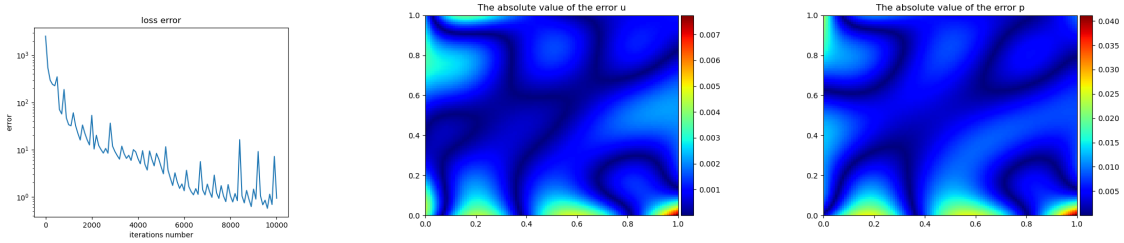
(a) (b) (c)

Figure 3. MIMONet iterations 10,000 times with Rule. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 4. MIMONet iterations 10,000 times with tanh. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 5. Physics-informed MIMONet iterations 10,000 times, $\lambda_0 = \lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 100$. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.

We utilize the Adam optimizer from PyTorch to train the networks, and take the corresponding learning rate 0.001. At the same time, we can choose different parameters to improve the accuracy of deep physics-informed MIMOONets method for the following format

$$\mathcal{L}(\theta) = \lambda_0\mathcal{L}_{data}(\theta) + \lambda_1\mathcal{L}_{pde1}(\theta) + \lambda_2\mathcal{L}_{pde2}(\theta) + \lambda_3\mathcal{L}_{BC1}(\theta) + \lambda_4\mathcal{L}_{BC2}(\theta).$$

We take $\lambda_0 = \lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 100$. The experimental results are shown in Table 2 and Figure 5.

Figures 3–5 show that both the deep MIMOONets method and the deep physics-informed MIMOONets method are effective for optimal control problems with elliptic constraints; also, the activation function Relu converges faster than Tanh with the MIMOONets method. However, the precision of the deep MIMOONets method can be achieved with only a small amount of data by means of the deep physics-informed MIMOONets method.

3.2. Semi-linear elliptic optimal control problem

A semi-linear elliptic constrained optimal control problem is considered here. The model is described as follows

$$\min J(u, v) = \frac{1}{2}\int_\Omega (u - u_d)^2 dx + \frac{\alpha}{2}\int_\Omega v^2 dx, \tag{3.9}$$

$$\text{subject to} \quad \begin{cases} -\Delta u + u^3 = v, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega. \end{cases} \tag{3.10}$$

The problems (3.9) and (3.10) lead to the following optimality system

$$\begin{cases} -\Delta u + u^3 - v = 0, & \text{in } \Omega, \\ \Delta p - 3u^2 p + u = u_d, & \text{in } \Omega, \\ \alpha v + p = 0, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega, \\ p = 0, & \text{on } \partial\Omega. \end{cases} \tag{3.11}$$

Table 3. MOONet for an optimal control of semi-linear elliptic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Relu	(0.62 ± 0.08)%	(0.076 ± 0.011)%
10,000	Tanh	(6.35 ± 0.35)%	(5.58 ± 0.22)%

Table 4. Physics-informed MOONet for an optimal control of semi-linear elliptic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Tanh	(2.40 ± 0.15)%	(1.60 ± 0.12)%

We use a multi-output operators network (MOONet) to solve the PDEs system (3.11). The corresponding solution operators G_1 and G_2 are shown below

$$\begin{cases} G_\theta^1(u_d) = \sum\limits_{k=1}^{p} b1_k^{u_d} tr_k, \\ G_\theta^2(u_d) = \sum\limits_{k=1}^{p} b2_k^{u_d} tr_k, \end{cases} \tag{3.12}$$

where the branch network is a 5-layer FNN with 300 neurons per hidden layer and output layer. The input layer contains 100 neurons. The trunk network is composed of five hidden layers with 300 neurons in each layer and 150 neurons in the output layer. The loss function for known data is similar to (3.5). Takeing into account the physics-informed MOONet and $\alpha = 1$, the corresponding loss function is expressed as follows

$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{data}(\theta) + \lambda_1 \mathcal{L}_{physics}(\theta), \tag{3.13}$$

where

$$\mathcal{L}_{physics}(\theta) = \lambda_2 \mathcal{L}_{pde1}(\theta) + \lambda_3 \mathcal{L}_{pde2}(\theta) + \lambda_4 \mathcal{L}_{BC1}(\theta) + \lambda_5 \mathcal{L}_{BC2}(\theta), \tag{3.14}$$

and

$$\begin{cases} \mathcal{L}_{pde1}(\theta) = \frac{1}{N_1} \sum\limits_{k=1}^{N_1} \mid -\frac{\partial^2 G_\theta^1(u_d)(x_k,y_k)}{\partial x^2} - \frac{\partial^2 G_\theta^1(u_d)(x_k,y_k)}{\partial y^2} + (G_\theta^1(u_d)(x_k,y_k))^3 - G_\theta^2(u_d)(x_k,y_k) \mid^2, \\ \mathcal{L}_{pde2}(\theta) = \frac{1}{N_2} \sum\limits_{k=1}^{N_2} \mid \frac{\partial^2 G_\theta^2(f,u_d)(x_k,y_k)}{\partial x^2} + \frac{\partial^2 G_\theta^2(f,u_d)(x_k,y_k)}{\partial y^2} - 3(G_\theta^1(u_d)(x_k,y_k))^2 G_\theta^2(u_d)(x_k,y_k) \\ \qquad\qquad + G_\theta^1(u_d)(x_k,y_k) - u_d(x_k,y_k) \mid^2, \\ \mathcal{L}_{BC1}(\theta) = \frac{1}{N_{BC}} \sum\limits_{i=1}^{N_{BC}} \mid G_\theta^1(f,u_d)(x_i,y_i) \mid^2, \\ \mathcal{L}_{BC2}(\theta) = \frac{1}{N_{BC}} \sum\limits_{i=1}^{N_{BC}} \mid G_\theta^2(f,u_d)(x_i,y_i) \mid^2 . \end{cases} \tag{3.15}$$

When we use physics-informed MOONet, we choose $N = N_1 = N_2 = 10,000$, $N_{BC} = 400$, $\lambda_0 = \lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 300$. Not using physics-informed MOONet, we take $N = 10,000$, $\lambda_0 = 1$, the others are 0. The sample data are collected by finite difference and sequential quadratic programming. The experimental results are shown in Tables 7 and 8.

We find that the Relu of the activation function converges faster than Tanh using the MOONETS method, which is similar to the linear elliptic optimal control problem. The deep physical information MOONets method requires very little data to achieve the same precision of the deep MOONets method.

3.3. Parabolic optimal control problem

We consider the following parabolic optimal control problem,

$$\min J(u,v) = \frac{1}{2} \int_0^T \int_\Omega (u - u_d)^2 dxdt + \frac{\alpha}{2} \int_0^T \int_\Omega v^2 dxdt, \tag{3.16}$$

$$\text{subject to} \quad \begin{cases} \partial_t u - \Delta u - v = f, & \text{in } D, \\ u = 0, & \text{on } \partial\Omega, \\ u(x,0) = sin(\pi x), & \text{in } \Omega, \end{cases} \tag{3.17}$$

where $\Omega = (0, 1)$, $D = \Omega \times (0, T]$, $u : D \to \mathbb{R}$ is the unknown term satisfying (3.17), $u_d : D \to \mathbb{R}$ is the given desired temperature distribution, v is the unknown control function, f is the source term in D, and $\alpha \geq 0$ is a regularization parameter. Here, we set $u_d(x,t) = \sin(\pi x)(t+1) + \frac{1}{2}e^t \sin(\pi x) - \frac{1}{2}(e^t - e)\sin(\pi x)$, $\alpha = 1$, $f(x,t) = \sin(\pi x) + \pi^2(t+1)(\sin(\pi x) + \frac{1}{2}(e^t - e)\sin(\pi x)$, $\varphi(x) = sin(\pi x)$. When $v(x,t) = -\frac{1}{2}(e^t - e)\sin(\pi x)$, $u(x,t) = (t+1)\sin(\pi x)$, the $J(u,v)$ gets the global minimum.

The optimal control problems (3.16) and (3.17) can be transformed into the optimality system,

$$\begin{cases} \partial_t u - \Delta u - v = f, & \text{in } D, \\ \partial_t p + \Delta p + u = u_d, & \text{in } D, \\ \alpha v + p = 0, & \text{in } D, \\ u(x,0) = sin(\pi x), & \text{in } \Omega, \\ p(x,T) = 0, & \text{in } \Omega, \\ u(x,t) = 0, & \text{on } \partial\Omega \times (0,T], \\ p(x,t) = 0, & \text{on } \partial\Omega \times (0,T]. \end{cases} \tag{3.18}$$

For the PDEs system (3.18), we use MIMOONets to solve it. The operator G^1 and G^2 can be learned from the source term f, u_d and φ. Their representations are as follows

$$\begin{cases} G^1_\theta(f, u_d, \varphi) = \sum\limits_{k=1}^{p} b1_k^f b1_k^{u_d} b1_k^{\varphi} tr_k, \\ G^2_\theta(f, u_d, \varphi) = \sum\limits_{k=1}^{p} b2_k^f b2_k^{u_d} b2_k^{\varphi} tr_k. \end{cases} \tag{3.19}$$

Here, we select 100 sensors points for input functions f, u_d and φ. Three branch networks are three separate 5-layer FNN with 300 neurons per hidden layer and output layer. The trunk network is a 5-layer FNN with 300 neurons per hidden layer and 150 neurons for output layer. Relu or Tanh is used as the activation function.

We use the same network structure of the MIMOONets as the network structure physics-informed MIMOONets where the activation function is Tanh. The corresponding loss function for the deep MIMOONets is expressed by

$$\mathcal{L}_{data}(\theta) = \frac{1}{N}\sum_{k=1}^{N}(\mid u(x_k,t_k) - G^1_\theta(f,u_d,\varphi,\psi)(x_k,t_k) \mid^2 + \mid p(x_k,t_k) - G^2_\theta(f,u_d,\varphi,\psi)(x_k,t_k) \mid^2). \tag{3.20}$$

The deep physics-informed MIMOONets loss function is the following form

$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{data}(\theta) + \lambda_1 \mathcal{L}_{physics}(\theta), \tag{3.21}$$

where

$$\mathcal{L}_{physics}(\theta) = \lambda_2 \mathcal{L}_{pde1}(\theta) + \lambda_3 \mathcal{L}_{pde2}(\theta) + \lambda_4 \mathcal{L}_{IC}(\theta) + \lambda_5 \mathcal{L}_{TC}(\theta) + \lambda_6 \mathcal{L}_{BC}(\theta), \tag{3.22}$$

and

$$\begin{cases} \mathcal{L}_{pde1}(\theta) = \frac{1}{N_f}\sum\limits_{k=1}^{N_f} \mid \frac{\partial G_\theta^1(f,u_d,\varphi)(x_k,t_k)}{\partial t} - \frac{\partial^2 G_\theta^1(f,u_d,\varphi)(x_k,t_k)}{\partial x} - G_\theta^2(f,u_d,\varphi)(x_k,t_k) - f(x_k,t_k) \mid^2, \\ \mathcal{L}_{pde2}(\theta) = \frac{1}{N_{u_d}}\sum\limits_{k=1}^{N_{u_d}} \mid \frac{\partial G_\theta^2(f,u_d,\varphi)(x_k,t_k)}{\partial t} + \frac{\partial^2 G_\theta^2(f,u_d,\varphi)(x_k,t_k)}{\partial x^2} + G_\theta^1(f,u_d,\varphi)(x_k,t_k) - u_d(x_k,t_k) \mid^2, \\ \mathcal{L}_{IC}(\theta) = \frac{1}{N_{BC}}\sum\limits_{i=1}^{N_{BC}} \mid G_\theta^1(f,u_d,\varphi)(x_i) - \varphi(x_i,t_i) \mid^2, \\ \mathcal{L}_{TC}(\theta) = \frac{1}{N_{IC}}\sum\limits_{i=1}^{N_{TC}} \mid G_\theta^2(f,u_d,\varphi)(x_i,t_i) \mid^2, \\ \mathcal{L}_{BC}(\theta) = \frac{1}{N_{BC}}\sum\limits_{i=1}^{N_{BC}} (\mid G_\theta^1(f,u_d,\varphi)(x_i,t_i) \mid^2 + \mid G_\theta^2(f,u_d,\varphi)(x_i,t_i) \mid^2). \end{cases}$$

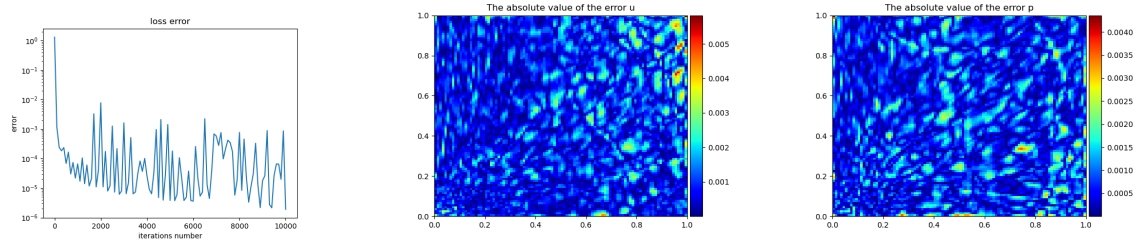
To calculate the value of the loss function, we take the random sample of initial data points $N = 10,000$, the numbers of the residual training points $N_f = N_{u_d} = 10,000$, the number N_{IC} and N_{TC} of the initial and termination condition points $x \in \Omega$ are 100, and the number N_{BC} of the boundary condition is 100. Then, we use the Adam optimizer to train the deep MIMOONets and physics-informed MIMOONets ($\lambda_i = 1, i = 0, 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$) by minimizing the loss of Eqs (3.20) and (3.21). The learning rate is 0.002. The experimental results are shown in Tables 5 and 6, and Figures 6–8.

Table 5. MIMOONets for parabolic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Relu	(0.11 ± 0.04)%	(0.24 ± 0.05)%
10,000	Tanh	(0.60 ± 0.10)%	(1.60 ± 0.13)%
40,000	Relu	(0.054 ± 0.006)%	(0.10 ± 0.04)%
40,000	Tanh	(0.40 ± 0.05)%	(0.88 ± 0.10)%

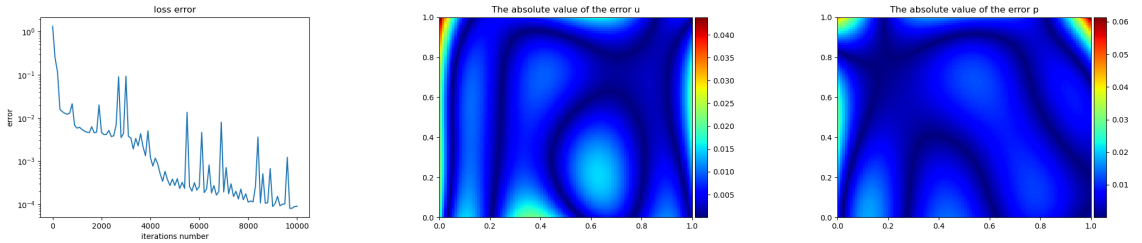
Table 6. Physics-informed MIMOONets for parabolic problem.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Tanh	(0.52 ± 0.12)%	(4.9 ± 0.27)%
40,000	Tanh	(0.32 ± 0.08)%	(4.7 ± 0.21)%



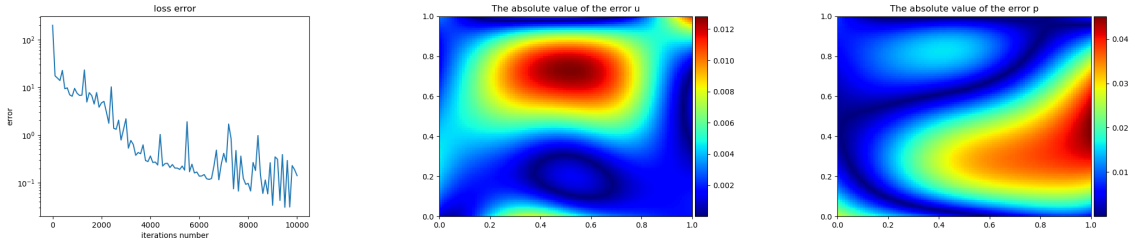
(a) (b) (c)

Figure 6. MIMONet iterations 10,000 times with Rule for parabolic-constraint optimal control problem. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 7. MIMONet iterations 10,000 times with tanh for parabolic-constraint optimal control problem. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 8. Physics-informed MIMONet iterations 10,000 times for parabolic-constraint optimal control problem ($\lambda_i = 1, i = 0, 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.

We found that both the deep MIMOONets method and the deep physics-informed MIMOONets method are effective for parabolic optimal control problem, and the activation function Relu converges faster than Tanh with the MIMOONets method. We also found that the physics-informed MIMOONets

not only attains comparable accuracy to the original MIMOONets but also satisfies the underlying PDEs constraint.

3.4. Hyperbolic optimal control problem

Consider the following hyperbolic optimal control problem [13]:

$$\min J(u,v) = \frac{1}{2}\int_0^T \int_\Omega (u-u_d)^2 dxdt + \frac{\alpha}{2}\int_0^T \int_\Omega v^2 dxdt, \tag{3.23}$$

$$\text{subject to} \quad \begin{cases} u_{tt} - \Delta u - v = f, & \text{in } D, \\ u = 0, & \text{on } \partial\Omega \times [0,T], \\ u(x,0) = \varphi(x), u_t(x,0) = \psi(x), & \text{in } \Omega, \\ a \le v \le b, & \text{in } D, a, b \in \mathbb{R}, \end{cases} \tag{3.24}$$

where $\Omega = [0,1]^2$, $D = \Omega \times (0,T]$. In the experiment, we take $u_d(x,t) = \sin(\pi x_1)\sin(\pi x_2)(e^t + 2 + 2\pi^2(t-T)^2)$, $T = 1, \alpha = 1, a = 0.2, b = 0.8$, $f(x,t) = (1+2\pi^2)e^t \sin(\pi x_1)\sin(\pi x_2) - \max\{a, \min\{b, (t-T)^2 \sin(\pi x_1)\sin(\pi x_2)\}\}$, $\varphi(x) = \sin(\pi x_1)\sin(\pi x_2)$, $\psi(x) = \sin(\pi x_1)\sin(\pi x_2)$. The exact solution $u(x,t) = e^t \sin(\pi x_1)\sin(\pi x_2)$, $v(x,t) = \max\{a, \min\{b, (t-T)^2 \sin(\pi x_1)\sin(\pi x_2)\}\}$.

Based on (3.23) and (3.24), the following optimality system can be obtained,

$$\begin{cases} u_{tt} - \Delta u - v = f, & \text{in } D, \\ p_{tt} - \Delta p - u = -u_d, & \text{in } D, \\ v = \max\{a, \min\{b, -\frac{p}{\alpha}\}\}, & \text{in } D, \\ u(x,0) = \varphi(x), u_t(x,0) = \psi(x), & \text{in } \Omega, \\ u(x,t) = 0, & \text{on } \partial\Omega \times [0,T], \\ p(x,T) = 0, p_t(x,T) = 0, & \text{in } \Omega, \\ p(x,t) = 0, & \text{on } \partial\Omega \times [0,T]. \end{cases} \tag{3.25}$$

The solution operator of system (3.25) can be represented by G^1 and G^2 as follows:

$$\begin{cases} G^1_\theta(f, u_d, \varphi, \psi) = \sum\limits_{k=1}^{p} b1_k^f b1_k^{u_d} b1_k^\varphi b1_k^\psi tr_k, \\ G^2_\theta(f, u_d, \varphi, \psi) = \sum\limits_{k=1}^{p} b2_k^f b2_k^{u_d} b2_k^\varphi b2_k^\psi tr_k. \end{cases} \tag{3.26}$$

In the experiment, we use 100 sensor points as input functions f, u_d, φ, and ψ. Each of the four branch networks consists of an independent 5-layer FNN, with 300 neurons in each hidden layer and output layer. The trunk network is a 5-layer FNN with 300 neurons per hidden layer and 150 neurons for the output layer. The activation function used is either Relu or Tanh.

We employ the same network structure as the physics-informed MIMOONets, using Tanh as the activation function. The corresponding loss function for the deep MIMOONets is expressed as follows:

$$\begin{aligned} \mathcal{L}_{data}(\theta) \quad = \quad & \frac{1}{N}\sum_{k=1}^{N} (\mid u(x_k, x_2^k, t_k) - G^1_\theta(f, u_d, \varphi)(x_1^k, x_2^k, t_k) \mid^2 \\ & + \mid p(x_1^k, x_2^k, t_k) - G^2_\theta(f, u_d, \varphi)(x_1^k, x_2^k, t_k) \mid^2). \end{aligned} \tag{3.27}$$

The deep physics-informed MIMOONets loss function is the following form

$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{data}(\theta) + \lambda_1 \mathcal{L}_{physics}(\theta), \tag{3.28}$$

where

$$\mathcal{L}_{physics}(\theta) = \lambda_2 \mathcal{L}_{pde1}(\theta) + \lambda_3 \mathcal{L}_{pde2}(\theta) + \lambda_4 \mathcal{L}_{IC}(\theta) + \lambda_5 \mathcal{L}_{TC}(\theta) + \lambda_6 \mathcal{L}_{BC}(\theta), \tag{3.29}$$

and

$$\begin{cases} \mathcal{L}_{pde1}(\theta) = \frac{1}{N_f} \sum\limits_{k=1}^{N_f} \mid \frac{\partial^2 G^1_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k)}{\partial t^2} - \frac{\partial^2 G^1_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k}{\partial x_1^2} - \frac{\partial^2 G^1_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k}{\partial x_2^2} \\ \qquad - \max\{a, \min\{b, -\frac{G^2_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k)}{\alpha}\}\} - f(x_1^k, x_2^k, t^k) \mid^2, \\ \mathcal{L}_{pde2}(\theta) = \frac{1}{N_{u_d}} \sum\limits_{k=1}^{N_{u_d}} \mid \frac{\partial^2 G^2_\theta(f,u_d,\varphi)(x_1^k,x_2^k,t^k))}{\partial t^2} + \frac{\partial^2 G^2_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k))}{\partial x_1^2} + \frac{\partial^2 G^2_\theta(f,u_d,\varphi,\psi)(x_1^k,x_2^k,t^k))}{\partial x_2^2} \\ \qquad + G^1_\theta(f, u_d, \varphi, \psi)(x_1^k, x_2^k, t^k) - u_d(x_1^k, x_2^k, t^k) \mid^2, \\ \mathcal{L}_{IC}(\theta) = \frac{1}{N_{BC}} \sum\limits_{i=1}^{N_{BC}} (\mid G^1_\theta(f, u_d, \varphi, \psi)(x_1^i, x_2^i, t^i) - \varphi(x_1^i, x_2^i) \mid^2 \\ \qquad + \mid \frac{\partial G^1_\theta(f,u_d,\varphi,\psi)(x_1^i,x_2^i,t^i)}{\partial t} - \psi(x_1^i, x_2^i) \mid^2), \\ \mathcal{L}_{TC}(\theta) = \frac{1}{N_{IC}} \sum\limits_{i=1}^{N_{TC}} (\mid G^2_\theta(f, u_d, \varphi, \psi)(x_1^i, x_2^i, t^i) \mid^2 + \mid \frac{\partial G^2_\theta(f,u_d,\varphi,\psi)(x_1^i,x_2^i,t^i)}{\partial t} \mid^2). \\ \mathcal{L}_{BC}(\theta) = \frac{1}{N_{BC}} \sum\limits_{i=1}^{N_{BC}} (\mid G^1_\theta(f, u_d, \varphi, \psi)(x_1^i, x_2^i, t^i) \mid^2 + \mid G^2_\theta(f, u_d, \varphi, \psi)(x_1^i, x_2^i, t^i) \mid^2). \end{cases} \tag{3.30}$$

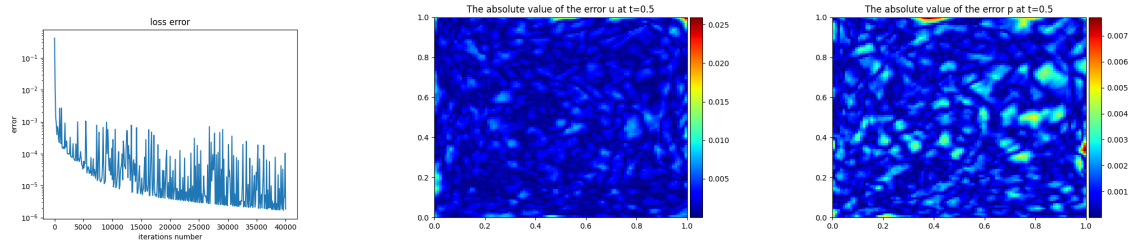
In the experiment, uniformly random samples are taken with $N = 10,000$ initial data points, and the number of residual training points is $N_f = N_{u_d} = 125,000$. The sample size for initial condition points N_{IC} and termination condition points N_{TC} in Ω is 10,000, and the number of points on the spatiotemporal boundary $\partial\Omega \times [0, T]$ is $N_{BC} = 400 \times 100$. For the sampling of sensor points for input functions, besides the source function $f(x_1, x_2, t)$, 100 sensor points are uniformly randomly sampled. However, for the source function $f(x_1, x_2, t)$ in the spatiotemporal space D, Latin hypercube sampling is used to sample 100 sensor points. Subsequently, we use the Adam optimizer to train the deep MIMOONets and physics-informed MIMOONets ($\lambda_0 = 0, \lambda_i = 1, i = 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$) by minimizing the loss of Eqs (3.27) and (3.28). The learning rate is set to 0.0001. The experimental results are shown in Tables 7 and 8, and Figures 9–11.

Table 7. MIMOONets for hyperbolic control problem at t = 0.5.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Relu	(0.48 ± 0.08)%	(1.89 ± 0.09)%
10,000	Tanh	(3.85 ± 0.30)%	(21.20 ± 1.10)%
40,000	Relu	(0.32 ± 0.06)%	(1.09 ± 0.30)%
40,000	Tanh	(0.72 ± 0.08)%	(1.90 ± 0.50)%

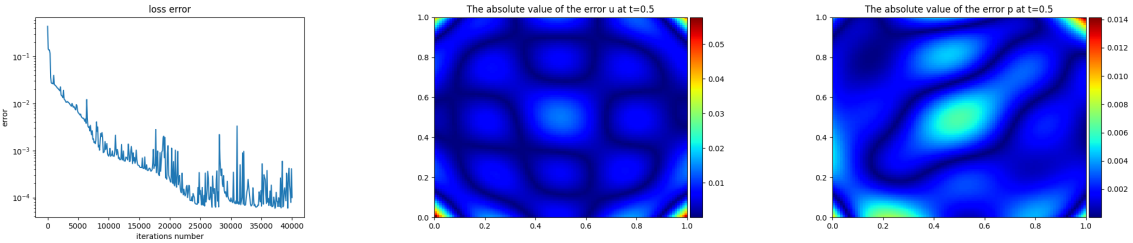
Table 8. Physics-informed MIMOONets for hyperbolic control problem at t = 0.5 without data.

Iterations	Activation function	Relative L^2 error of u	Relative L^2 error of p
10,000	Tanh	(6.25 ± 0.80)%	(13.57 ± 1.20)%
40,000	Tanh	(1.72 ± 0.30)%	(5.2 ± 0.31)%



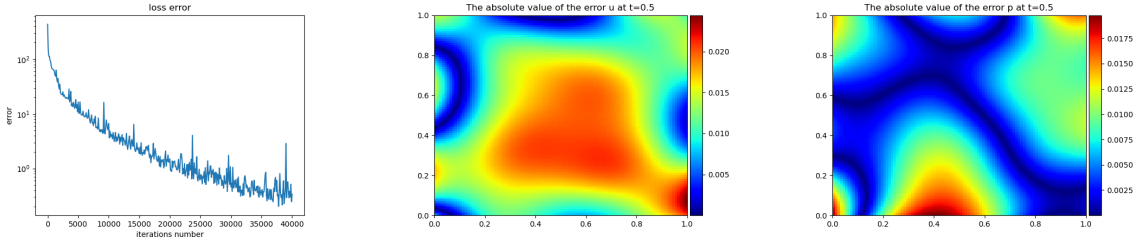
(a) (b) (c)

Figure 9. MIMONet iterations 40,000 times with Rule for hyperbolic problem ($t = 0.5$). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 10. MIMONet iterations 40,000 times with tanh for hyperbolic problem ($t = 0.5$). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.



(a) (b) (c)

Figure 11. Physics-informed MIMONet iterations 40,000 times for hyperbolic problem ($\lambda_0 = 0, \lambda_i = 1, i = 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p.

For hyperbolic optimal control problems, both the deep MIMOONets method and the deep physics-informed MIMOONets method are effective. In the experiments, the MIMOONets method adopts a data-driven approach, and the Relu activation function converges faster than Tanh. As for the physics-informed MIMOONets method, we adopt a non-data-driven mode and conduct experiments by considering the PDE as a constraint. The experimental results are excellent.

4. Conclusions

In this work, a novel deep learning framework is presented, which enables the construction of fast surrogates for solving PDE-constrained optimization problems using MIMOONets and physics-informed MIMOONets. The MIMOONets frameworks (MIMOONets and physics-informed MIMOONets) offer flexibility and faster implementation compared to other traditional methods. Compared with MIMOONets, the physics-informed MIMOONets require little paired input-output data, and are more efficient and cost-effective.

Although our methods (MIMOONets and physics-informed MIMOONets) were initially designed for solving PDE-constrained optimization problems, they can also be extended to multi-equation-coupled problems, including hyperbolic and parabolic optimal control problems [31], Cahn-Hilliard-Navier-Stokes equation [32], and other scenarios. Moreover, they can be effectively employed for real-time prediction in optimal systems governed by PDEs.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Granted No. 11961008) and Funding from the Scientific Research Fund Project of Guizhou Education University (Granted No. 2024ZD007).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. G. Fabbri, Heat transfer optimization in corrugated wall channels, *Int. J. Heat Mass Transfer*, **43** (2000), 4299–4310. https://doi.org/10.1016/S0017-9310(00)00054-5
2. G. Cornuéjols, J. Peña, R. Tütüncü, *Optimization Methods in Finance*, 2^{nd}, Cambridge University Press, New York, 2018.
3. J. C. De los Reyes, C. B. Schönlieb, Image denoising: learning the noise model via nonsmooth PDE-constrained optimization, *Inverse Probl. Imaging*, **7** (2013), 1183–1214. https://doi.org/10.3934/ipi.2013.7.1183

4. J. Sokolowski, J. P. Zolésio, *Introduction to Shape Optimization*, Springer-Verlag, Berlin, 1992. https://doi.org/10.1007/978-3-642-58106-9_1

5. J. Haslinger, R. A. E. Mäkinen, *Introduction to Shape Optimization: Theory, Approximation, and Computation*, SIAM, Philadelphia, 2003. https://doi.org/10.1137/1.9780898718690

6. R. M. Hicks, P. A. Henne, Wing design by numerical optimization, *J. Aircr.*, **15** (1978), 407–412. https://doi.org/10.2514/3.58379

7. P. D. Frank, G. R. Shubin, A comparison of optimization-based approaches for a model computational aerodynamics design problem, *J. Comput. Phys.*, **98** (1992), 74–89. https://doi.org/10.1016/0021-9991(92)90174-W

8. J. Ng, S. Dubljevic, Optimal boundary control of a diffusion-convection-reaction PDE model with time-dependent spatial domain: Czochralski crystal growth process, *Chem. Eng. Sci.*, **67** (2012), 111–119. https://doi.org/10.1016/j.ces.2011.06.050

9. S. P. Chakrabarty, F. B. Hanson, Optimal control of drug delivery to brain tumors for a distributed parameters model, in *Proceedings of the 2005, American Control Conference*, **2** (2005), 973–978. https://doi.org/10.1109/ACC.2005.1470086

10. W. B. Liu , N. N. Yan, *Adaptive Finite Element Methods for Optimal Control Governed by PDEs*, Science Press, Beijing, 2008.

11. Y. P. Chen, F. L. Huang, N. Yi, W. B. Liu, A Legendre Galerkin spectral method for optimal control problems governed by Stokes equations, *SIAM J. Numer. Anal.*, **49** (2011), 1625–1648. https://doi.org/10.1137/080726057

12. A. Borzì, V. Schulz, *Computational Optimization of Systems Governed by Partial Differential Equations*, SIAM, Philadelphia, 2011.

13. X. Luo, A priori error estimates of Crank-Nicolson finite volume element method for a hyperbolic optimal control problem, *Numer. Methods Partial Differ. Equations*, **32** (2016), 1331–1356. https://doi.org/10.1002/num.22052

14. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations, preprint, arXiv:1711.10561.

15. S. Wang, H. Zhang, X. Jiang, Physics-informed neural network algorithm for solving forward and inverse problems of variable-order space-fractional advection-diffusion equations, *Neurocomputing*, **535** (2023), 64–82. https://doi.org/10.1016/j.neucom.2023.03.032

16. J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, *J. Comput. Phys.*, **375** (2018), 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029

17. W. N. E, B. Yu, The deep Ritz method: a deep-learning based numerical algorithm for solving variational problems, *Commun. Math. Stat.*, **6** (2018), 1–12. https://doi.org/10.1007/s40304-018-0127-z

18. Y. L. Liao, P. B. Ming, Deep Nitsche method: deep Ritz method with essential boundary conditions, *Commun. Comput. Phys.*, **29** (2021), 1365–1384. https://doi.org/10.4208/cicp.OA-2020-0219

19. L. Lu, P. Z. Jin, G. F. Pang, Z. Q. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.*, **3** (2021), 218–229. https://doi.org/10.1038/s42256-021-00302-5

20. C. Moya, S. Zhang, G. Lin, M. Yue, DeepONet-grid-UQ: a trustworthy deep operator framework for predicting the power grid's post-fault trajectories, *Neurocomputing*, **535** (2023), 166–182. https://doi.org/10.1016/j.neucom.2023.03.015

21. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, et al., Neural operator: graph kernel network for partial differential equations, preprint, arXiv:2003.03485.

22. Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, et al., Fourier neural operator for parametric partial differential equations, preprint, arXiv:2010.08895.

23. S. F. Wang, H. W. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets, *Sci. Adv.*, **7** (2021), eabi8605. https://doi.org/10.1126/sciadv.abi8605

24. P. Jin, S. Meng, L. Lu, MIONet: learning multiple-input operators via tensor product, *SIAM J. Sci. Comput.*, **44** (2022), A3490–A3514. https://doi.org/10.1137/22M1477751

25. C. J. García-Cervera, M. Kessler, F. Periago, Control of partial differential equations via physics-informed neural networks, *J. Optim. Theory Appl.*, **196** (2023), 391–414. https://doi.org/10.1007/s10957-022-02100-4

26. S. Mowlavi, S. Nabib, Optimal control of PDEs using physics-informed neural networks, *J. Comput. Phys.*, **473** (2023), 111731. https://doi.org/10.1016/j.jcp.2022.111731

27. J. Barry-Straume, A. Sarsha, A. A. Popov, A. Sandu, Physics-informed neural networks for PDE-constrained optimization and control, preprint, arXiv:2205.03377.

28. S. F. Wang, M. A. Bhouri, P. Perdikaris, Fast PDE-constrained optimization via self-supervised operator learning, preprint, arXiv:2110.13297.

29. J.L. Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, Springer-Verlag, Berlin, 1971.

30. T. P. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE Trans. Neural Networks*, **6** (1995), 911–917. https://doi.org/10.1109/72.392253

31. I. Lasiecka, *Mathematical Control Theory of Coupled PDEs*, SIAM, Philadelphia, 2001. https://doi.org/10.1137/1.9780898717099

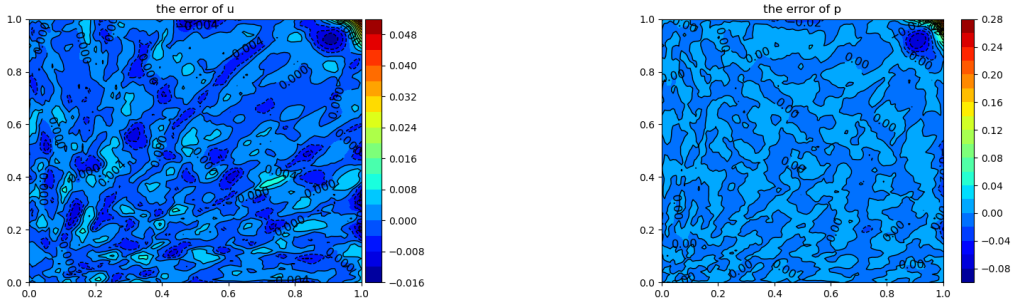
32. A. Miranville, *The Cahn-Hilliard Equation: Recent Advances and Applications*, SIAM, Philadelphia, 2019. https://doi.org/10.1137/1.9781611975925

Appendix

A. Supplementary visualizations of elliptic optimal control problem

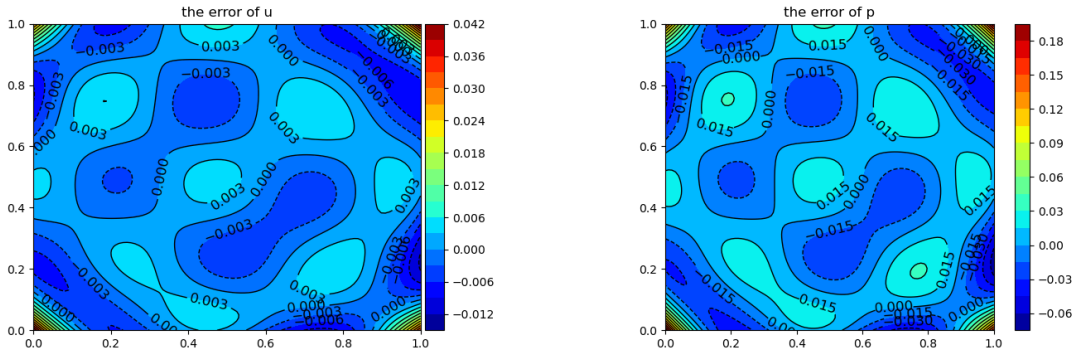
We present some numerical images for solving elliptic optimal control problems (3.1) and (3.2) using deep MIMOONets and physics-informed MIMOONets framework. The data-driven results are

shown in Figures A.1–A.6. When there is no data, we can also use deep physics-informed MIMOONets to solve the problem. The experimental results are shown in Figure A.7.



(a) (b)

Figure A.1. MIMONet iterations 10,000 times with Relu for elliptic constraint optimal control problem. (a) The error of u: $u - G_\theta^1(f, u_d)$. (b) The error of p: $p - G_\theta^2(f, u_d)$.



(a) (b)

Figure A.2. MIMONet iterations 10,000 times with Tanh for elliptic constraint optimal control problem. (a) The error of u : $u - G_\theta^1(f, u_d)$. (b) The error of p: $p - G_\theta^2(f, u_d)$.

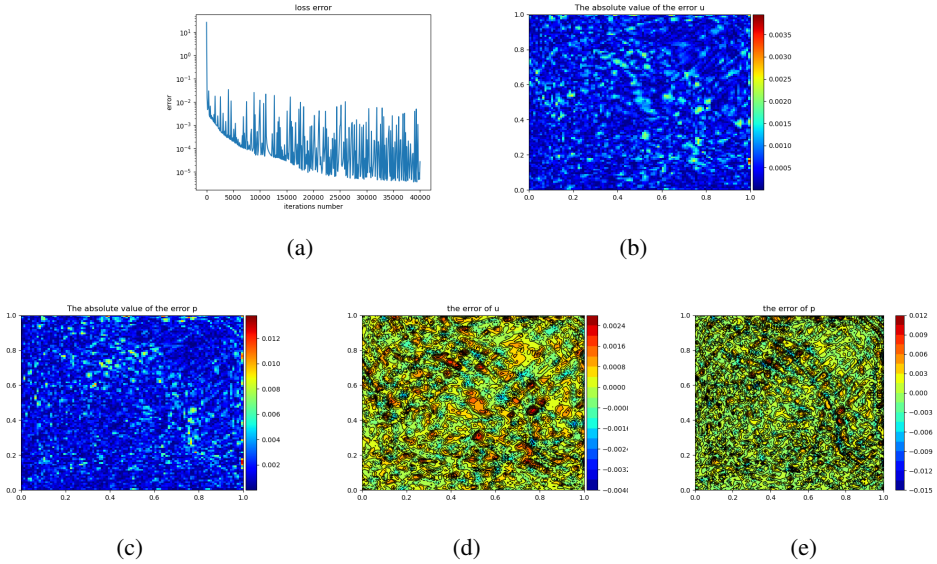


Figure A.3. MIMONet iterations 40,000 times with Relu for elliptic constraint optimal control problem. (a) Train loss error. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d)$. (e) The error of p: $p - G_\theta^2(f, u_d)$.

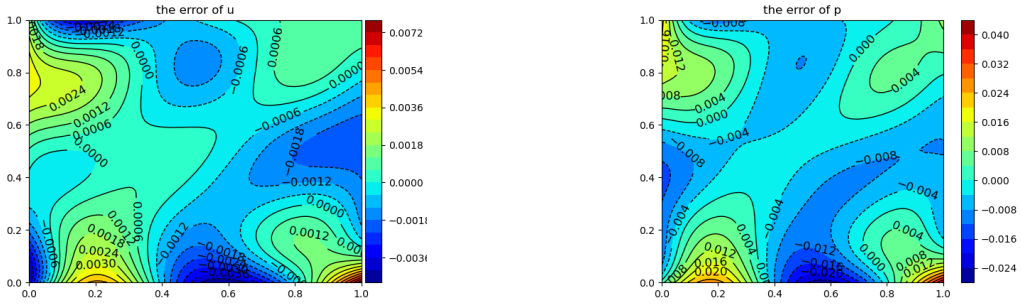


Figure A.4. Physics-informed MIMONet iterations 10,000 times for elliptic constraint optimal control problem ($\lambda_0 = \lambda_1 = \lambda_2 = 1, \lambda_3 = \lambda_4 = 100$). (a) The error of u: $u - G_\theta^1(f, u_d)$. (b) The error of p: $p - G_\theta^2(f, u_d)$.

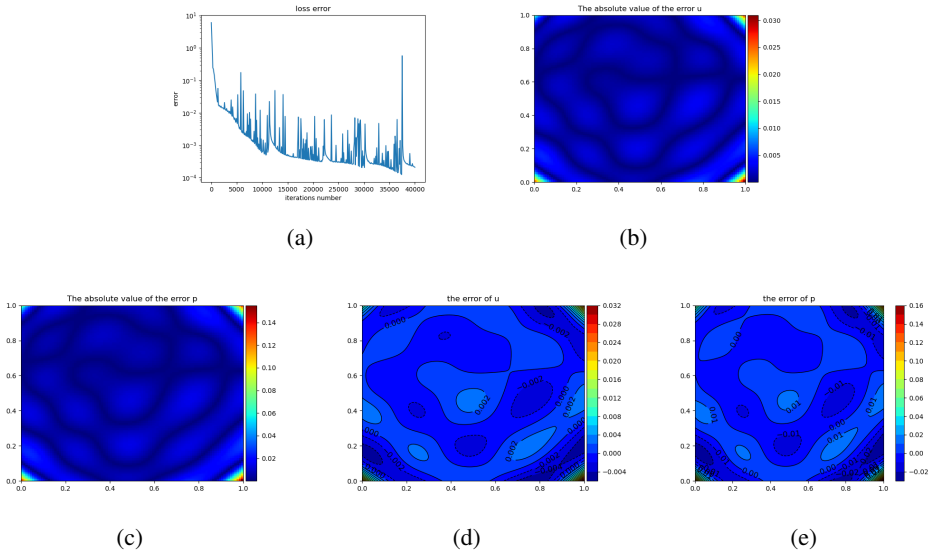


Figure A.5. MIMONet iterations 40,000 times with Tanh for elliptic-constraint optimal control problem. (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d)$. (e) The error of p: $p - G_\theta^2(f, u_d)$.

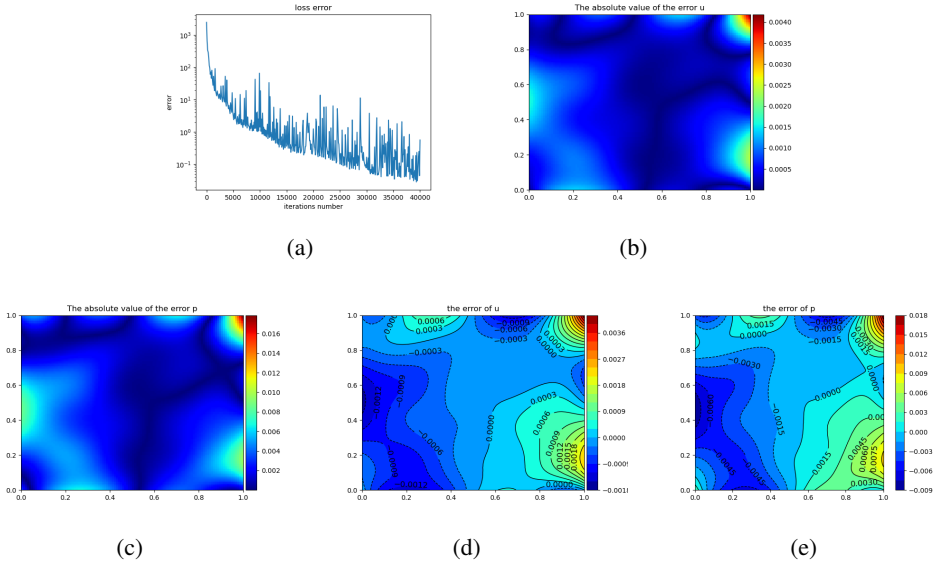


Figure A.6. Physics-informed MIMONet iterations 40,000 times with Tanh for elliptic constraint optimal control problem ($\lambda_0 = \lambda_1 = \lambda_2 = 1, \lambda_3 = \lambda_4 = 100$). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d)$. (e) The error of p: $p - G_\theta^2(f, u_d)$.

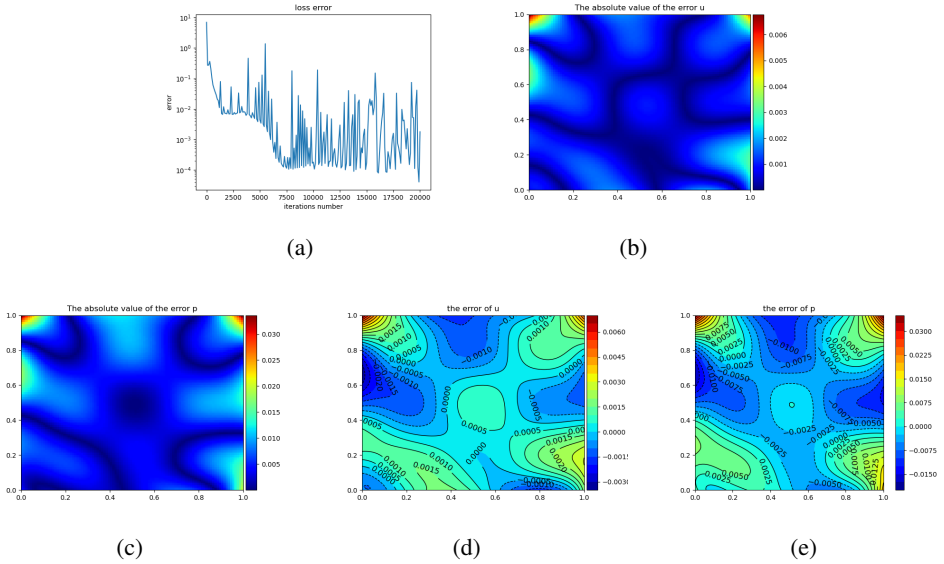
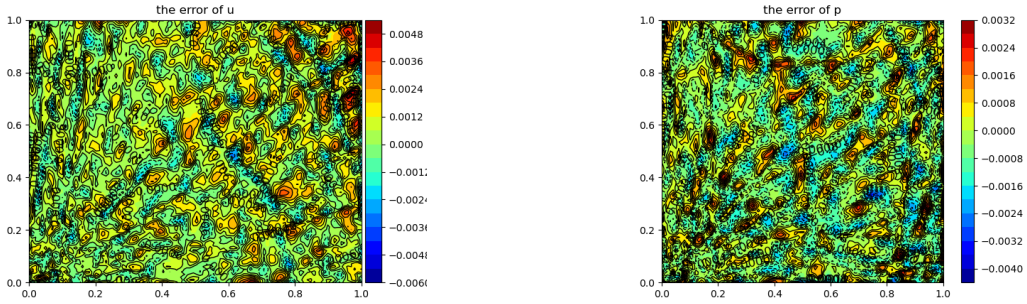


Figure A.7. Physics-informed MIMONet iterations 40,000 times for elliptic-constraint optimal control problem ($\lambda_0 = 0, \lambda_1 = \lambda_2 = 1, \lambda_3 = \lambda_4 = 100$, and there is no initial data). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G^1_\theta(f, u_d)$. (e) The error of p: $p - G^2_\theta(f, u_d)$.

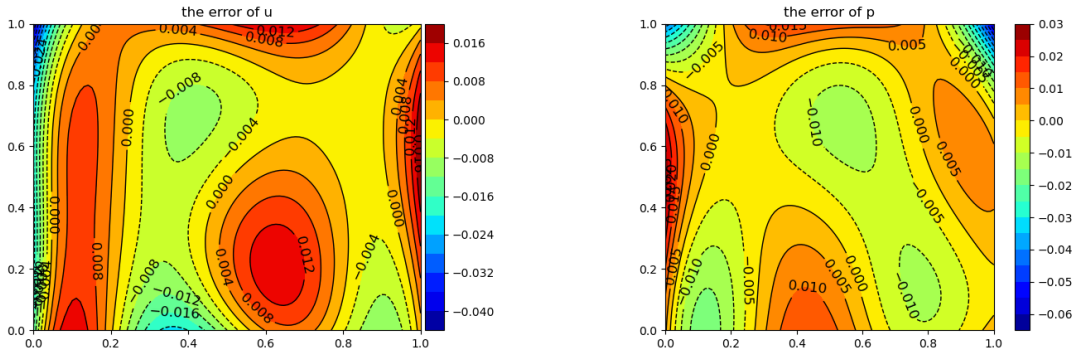
B. Supplementary visualizations of parabolic optimal control problem

We present some numerical images for solving parabolic optimal control problems using deep MIMOONets and physically informed MIMOONets frameworks. The data-driven results are shown in Figures B.1– B.6. When there is no initial data, we can also use MIMOONets based on depth physical information to solve this problem, and the experimental results are shown in Figure B.7.



(a) (b)

Figure B.1. MIMONet iterations 10,000 times with Relu for parabolic-constraint optimal control problem. (a) The error of u: $u - G_\theta^1(f, u_d, \varphi)$. (b) The error of p: $p - G_\theta^2(f, u_d, \varphi)$.



(a) (b)

Figure B.2. MIMONet iterations 10,000 times with Tanh for parabolic-constraint optimal control problem. (a) The error of u: $u - G_\theta^1(f, u_d, \varphi)$. (b) The error of p: $p - G_\theta^2(f, u_d, \varphi)$.

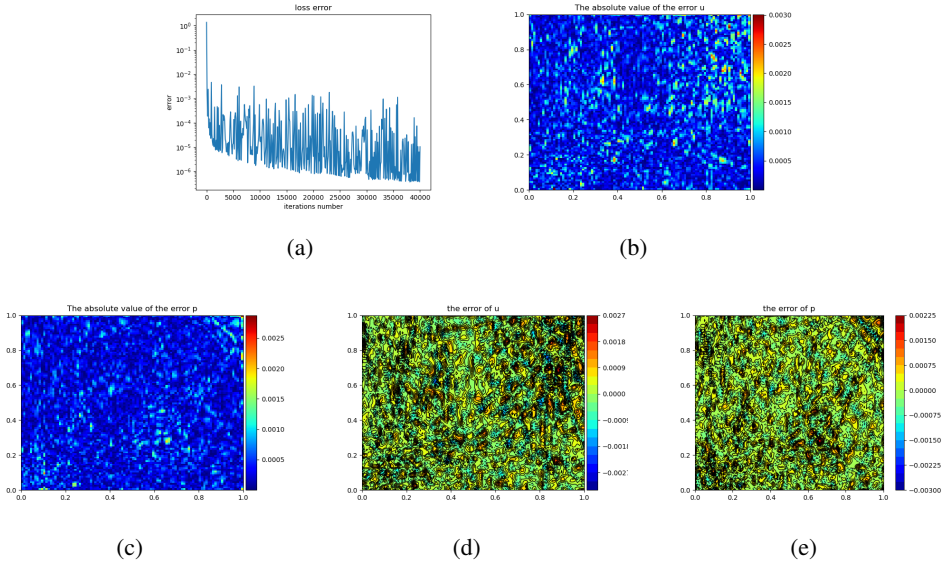


Figure B.3. MIMONet iterations 40,000 times with Relu for parabolic-constraint optimal control problem. (a) Train loss. (b) The absolute value of the error u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d, \varphi)$. (e) The error of p: $p - G_\theta^2(f, u_d, \varphi)$.

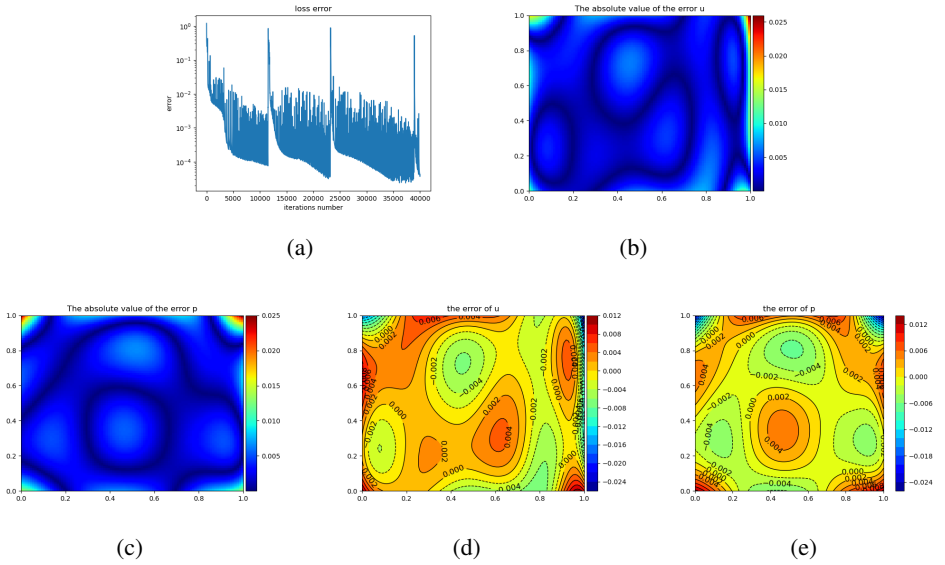
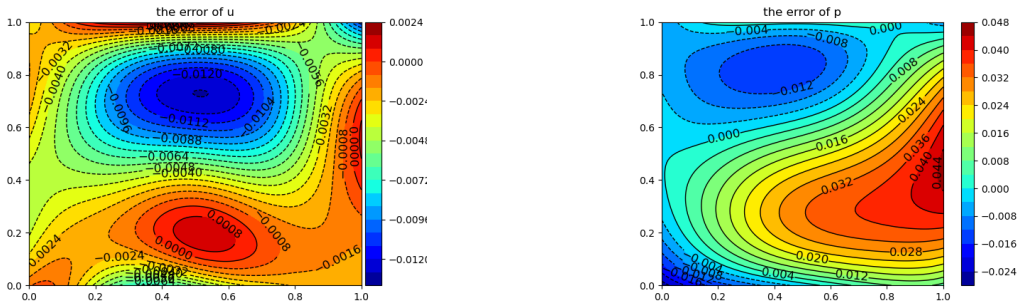
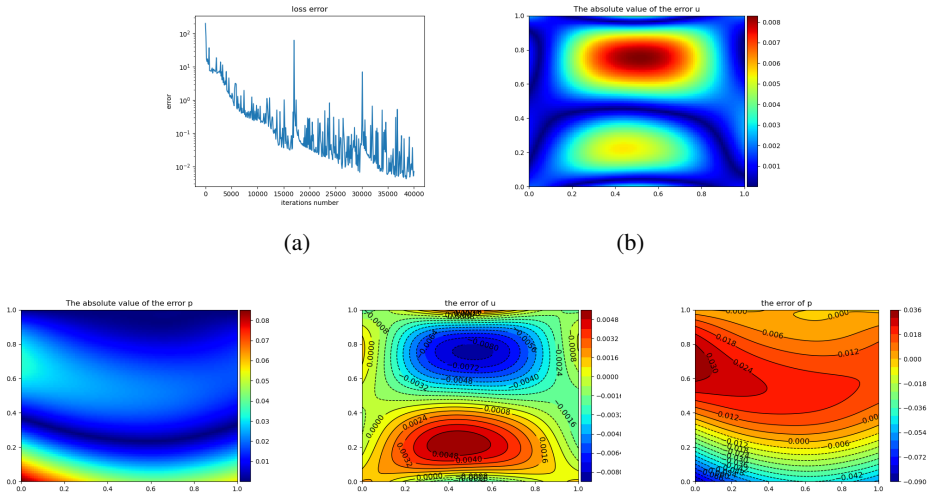


Figure B.4. MIMONet iterations 40,000 times with Tanh for parabolic-constraint optimal control problem. (a) Train loss. (b) The absolute value of the error u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d, \varphi)$. (e) The error of p: $p - G_\theta^2(f, u_d, \varphi)$.



(a) (b)

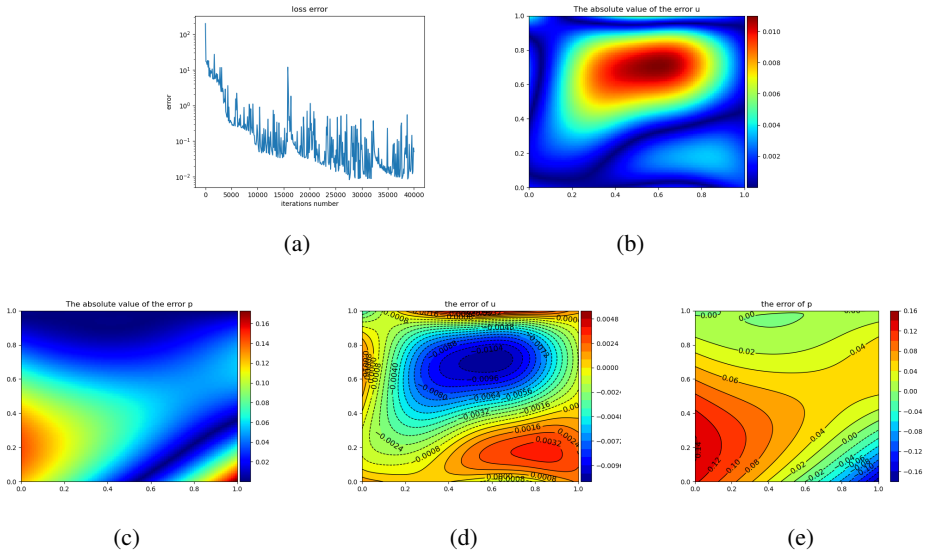
Figure B.5. Physics-informed MIMONet iterations 10,000 times for parabolic-constraint optimal control problem ($\lambda_i = 1, i = 0, 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$, there is no initial data). (a) The error of u: $u - G^1_\theta(f, u_d, \varphi)$. (b) The error of p: $p - G^2_\theta(f, u_d, \varphi)$.



(a) (b)

(c) (d) (e)

Figure B.6. Physics-informed MIMONet iterations 40,000 times for parabolic-constraint optimal control problem ($\lambda_i = 1, i = 0, 1, 2, 3; \lambda_i = 100, i = 4, 5, 6$, there is no initial data). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G^1_\theta(f, u_d, \varphi)$. (e) The error of p: $p - G^2_\theta(f, u_d, \varphi)$.



(a) (b)

(c) (d) (e)

Figure B.7. Physics-informed MIMONet iterations 40,000 times for parabolic-constraint optimal control problem ($\lambda_0 = 0; \lambda_i = 1, i = 1, 2, 3; \lambda_j = 100, j = 4, 5, 6$, there is no initial data). (a) Train loss. (b) The absolute value of the error of u. (c) The absolute value of the error of p. (d) The error of u: $u - G_\theta^1(f, u_d, \varphi)$. (e) The error of p: $p - G_\theta^2(f, u_d, \varphi)$.