Deep Multi-Input and Multi-Output Operator Networks Method for Optimal Control of Pdes

<table>
<thead>
<tr>
<th>Item Type</th>
<th>Preprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Yong, Jinjun; Luo, Xianbing; Sun, Shuyu</td>
</tr>
<tr>
<td>Eprint version</td>
<td>Pre-print</td>
</tr>
<tr>
<td>DOI</td>
<td>10.2139/ssrn.4353535</td>
</tr>
<tr>
<td>Publisher</td>
<td>Elsevier BV</td>
</tr>
<tr>
<td>Rights</td>
<td>This is a preprint version of a paper and has not been peer reviewed. Archived with thanks to Elsevier BV.</td>
</tr>
<tr>
<td>Download date</td>
<td>2023-11-01 05:42:26</td>
</tr>
<tr>
<td>Link to Item</td>
<td><a href="http://hdl.handle.net/10754/690163">http://hdl.handle.net/10754/690163</a></td>
</tr>
</tbody>
</table>
Deep Multi-input and Multi-output Operator Networks Method for Optimal Control of PDEs

Jinjun Yong\textsuperscript{a,b}, Xianbing Luo\textsuperscript{a,*}, Shuyu Sun\textsuperscript{c}

\textsuperscript{a}School of Mathematics and Statistics, Guizhou University, Guiyang, 550025, China
\textsuperscript{b}School of Mathematics And Big Data, Guizhou Education University, Guiyang, 550018, China
\textsuperscript{c}Computational Transport Phenomena Laboratory, Division of Physical Science and Engineering, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia

Abstract

Deep operator networks are widely used to solve optimal control problem due to the expressive capability of the approximate nonlinear operators. Generally, an optimal control problem is equivalent to a system of partial differential equations problem. Based on this, we propose a deep multi-input and multi-output operator neural network (MIMOONet) method. We apply the proposed MIMOONet and physically informed MIMOONet to solve the optimal control problems. It works successfully for solving the elliptic (linear and semi-linear) and parabolic optimal control problems.

Keywords: PDE Optimal control; Multi-input; Multi-output; Operator neural networks; Physics-informed neural networks

AMS subject classifications: 35R11, 35R20, 35R60, 65Mxx

1. Introduction

The optimal control problem has been successfully used 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]. A large number of numerical methods have been successfully used to solve PDE constrained optimal control problems, such as finite elements method, finite differences method, finite volume method, spectral methods and mesh less methods (see, e.g., [10–12]). Despite their prominence, the optimal control problem is notoriously difficult to solve, especially when the problem is nonlinear. The numerical methods for PDE optimal control needs to numerically approximate partial differential equation. Recently, deep learning is a popular method for solving partial differential equation, especially nonlinear equations.

Deep learning for solving PDEs has received a lot of attention, such as physics informed neural networks (PINN) [13], deep Galerkin method [14], deep Ritz method [15], deep Nitsche method [16].

This work is partially supported by the National Natural Science Foundation of China (11961008).

*Corresponding author.

Email addresses: yongjinjun0126.com (Jinjun Yong), xbluo1@gzu.edu.cn;luoxb121863 (Xianbing Luo), shuyu.sun@kaust.edu.sa (Shuyu Sun)
and deep operator networks (DeepONets) [17]. PINNs can be used to solve PDEs with specific initial and boundary conditions, as well as loading or source terms. But it requires expensive optimization during inference. Therefore, the PDEs with operating conditions and real-time inference can not be solved by PINNs. The neural operator full-fills this well. Various versions of the operator networks have been published, e.g., graph neural operator networks [18], Fourier neural operator (FNO) [19], physics-informed neural operators (PINO) [20] and deep multiple input operator network (DeepMIONet) [21].

Recently, neural networks have been used to tackle PDE-constrained optimization since neural networks may have some advantages over traditional solvers. It is a mesh-free scheme, which is helpful to deal with engineering problems in the background of complex geometric regions. Simulation of these control problems with complex geometric regions using traditional numerical methods usually requires high-quality grids and time-consuming human intervention in the preprocessing phase before the actual simulation. Many researchers are interested in using neural networks to replace traditional numerical methods. A methodology and a set of guidelines for solving optimal control problems with PINNs are proposed in [22, 23]. Barry-Straume et al use a two-stage framework to solve PDE-constrained optimization problems [24]. Wang et al use physics-informed deep operator networks (DeepONets) framework to learn the solution operator of parametric PDEs, which builds a surrogate for solving PDE-constrained optimization problems [25].

In this paper, to solve PDE-constrained optimal control problems with some data, we propose a MIMOONet. For the PDE optimal problem, the governing PDE is fully known and the goal is to find a control variable that minimizes the cost objective function. Firstly, the PDE-constrained optimal control problem is transformed into a PDE system by the adjoint method. Secondly, the PDE system is solved by MIMOONets. Furthermore, we consider a physical system that can be characterized by PDEs and give physics-informed MIMOONets to solve the PDE-constrained optimal control problem.

The remainder of this paper is organized as follows. In Sect.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 Sect.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, in Sect.4, summarizes the results, potential pitfalls, 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:

$$u^*, v^* = \arg \min_{u \in S, v \in U} J(u, v),$$

subject to $F(u, v) = 0$,

where $J : S \times U \to \mathbb{R}$ is a cost function, $F : S \times U \to V$ is a system of PDEs subject to initial and boundary conditions, $u$ and $v$ are state variable and control variable. Assume that the problem (1) exists unique solutions $u$ and $v$. In the subsequent, the MIMOONets method is presented under this assumption.

2.1. Optimality system

In this subsection, we transform PDE-constrained optimization problem into optimality system. Applying the Lagrangian method to the following problem [26],

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

subject to

$$\begin{cases}
F[u(x, t); v(x, t)] = 0, & x \in \Omega, t \in [0, T], \\
B[u(x, t)] = 0, & x \in \partial \Omega, t \in [0, T], \\
I[u(x, 0)] = 0, & x \in \Omega,
\end{cases}$$

where $x$ and $t$ denote space and time respectively, the domain $\Omega \subseteq \mathbb{R}^d$, $\partial \Omega$ is the boundary of the domain $\Omega$, $B$ and $I$ are boundary conditions and initial condition respectively. We construct the Lagrangian function of the problem (2) as follow:

$$L(u, v, p_1, p_2, p_3) = J(u, v) - \int_0^T \int_{\Omega} p_1 F(u, v) dx dt - \int_0^T \int_{\partial \Omega} p_2 B(u) ds dt - \int_{\Omega} p_3 I(u) dx.$$  \hspace{1cm} (4)

Here, $p_1, p_2, p_3$ are Lagrange multipliers 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 $p = (p_1, p_2, p_3)$ to satisfy the optimality conditions. Therefore, the problem (2)-(3) is equivalent to the following unconstrained problem

$$\begin{pmatrix} (u^*, v^*, p^*) = \arg \min_{u \in S, v \in U, p} L(u, v, p). \end{pmatrix}$$  \hspace{1cm} (5)

Then, the directional derivative of $L$ with respect to $u$ is disappear at the optimal point, that is

$$D_uL(u^*, v^*, p^*)\delta u = \lim_{\varepsilon \to 0} \frac{L(u^* + \varepsilon \delta u, v^*, p^*) - L(u^*, v^*, p^*)}{\varepsilon} = 0, \quad \forall \delta u \in S.$$  \hspace{1cm} (6)

For the control variable $v$ and Lagrange multipliers $p$, we have

$$D_vL(u^*, v^*, p^*)\delta v = 0, \quad \forall \delta v \in U,$$  \hspace{1cm} (7)

and

$$D_pL(u^*, v^*, p^*)\delta p = 0, \quad \forall \delta p,$$  \hspace{1cm} (8)

respectively. Therefore, the problem (2)-(3) can be written in the following way:

$$\begin{cases}
D_uL(u, v, p)\delta u = 0, \\
D_vL(u, v, p)\delta v = 0, \\
D_pL(u, v, p)\delta p = 0.
\end{cases}$$  \hspace{1cm} (9)
There exists various methods for obtaining the solution of the system (9). In this work, we use the MMOONet to solve the system.

2.2. Multiple-input operators networks

DeepONets provides a learning framework which can learn abstract nonlinear operators in infinite-dimensional function spaces. DeepONets is proposed by Lu et al in [17], which is defined in Theorem 2.1 and inspired by the universal approximation theorem of operators [27]. The network framework of the DeepONets is composed of trunk network and branch network. Trunk network provides the basis functions by encoding the information related to the space-time coordinates of the output function. Branch network encodes the input function to provide the coefficients at fixed sensor points.

**Theorem 2.1.** Suppose that \( X \) is a Banach space, \( K_1 \subset X \), \( K_2 \subset \mathbb{R}^d \) are two compact sets in \( X \) and \( \mathbb{R}^d \), respectively. \( V \) is a compact set in \( C(K_1) \), \( G : V \to C(K_2) \) is a nonlinear continuous operator, \( \sigma \) is a continuous nonpolynomial function. Then for \( \forall \varepsilon > 0 \), there exist \( n, p, m \in \mathbb{N} \), constants \( c^k_i, a^k_{ij}, \theta^k_i, \zeta^k \in \mathbb{R} \), \( w_k \subset \mathbb{R}^d \), \( x_j \in K_1 \), \( i = 1, 2, \ldots, n \), \( k = 1, \ldots, p \), \( j = 1, \ldots, m \), such that

\[
| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c^k_i \sigma(\sum_{j=1}^{m} a^k_{ij} u(x_j) + \theta^k_i) \sigma(w_k \cdot y + \zeta^k) | < \varepsilon,
\]

(10)

for any \( u \in V \) and \( y \in K_2 \).

The input of the DeepONets is a single function which is defined on a Banach space. However, multiple input functions should be considered in practical problems. The DeepMIONet is proposed in [21], and defined through the tensor product of Banach spaces by:

\[
G_{\theta}(u, v) = \sum_{i=1}^{p} b^u_i b^v_i tr_i,
\]

(11)

where \( b^u_i \) and \( b^v_i \) 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 is show in Figure 1, which has two different branch networks.

2.3. Multi-input and multi-output operators networks

In this subsection, we focus on solving the PDEs system using neural operator network. DeepONets and MIONet can be used to solving a single PDE. But for a PDEs system, two solution operators at least must be generated, and two output operators are needed for a network. To improve MIONets, we propose MIMOONets, which can be used to solve the PDEs system. The network framework of the MIMOONets is composed of a trunk network and multiple branches network. Trunk network provides the basis functions of the solution operators, and the multiple branches network provide more groups coefficients of solution operators at fixed sensor points.
Figure 1: Architectures of MIONet for $G_{\theta}(u, v)(x, t)$: The branch network 1 takes $u$ as input functional (employs an 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).

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

\[
\begin{align*}
L_1[u(x); v(x); w(x)] &= f(x), \\ L_2[u(x); v(x); w(x)] &= g(x), \\ L_3[u(x); v(x); w(x)] &= h(x), \\
B_1[u(x)] &= \varphi(x), \\ B_2[v(x)] &= \psi(x), \\ B_3[w(x)] &= \phi(x),
\end{align*}
\]

where $u$, $v$ and $w$ are functions, $L_1$, $L_2$ and $L_3$ are the differential operators, $\varphi(x)$, $\psi(x)$ and $\phi(x)$ are the boundary conditions of $u$, $v$ and $w$, respectively. Let $G^1$, $G^2$ and $G^3$ are the solutions of the operator taking input functions $f$, $g$, $h$, $\varphi$, $\psi$ and $\phi$. Then $G^1(f, g, h, \varphi, \psi, \phi)(y)$, $G^2(f, g, h, \varphi, \psi, \phi)(y)$ and $G^3(f, g, h, \varphi, \psi, \phi)(y)$ are the corresponding output functions. Given any point $y \in D$, the output $G^1(f, g, h, \varphi, \psi, \phi)(y)$, $G^2(f, g, h, \varphi, \psi, \phi)(y)$ and $G^3(f, g, h, \varphi, \psi, \phi)(y)$ are real numbers. According to the results of reference [21] and [27], the solutions $u, v, w$ can be expressed as

\[
\begin{align*}
u &= G^1(f, g, h, \varphi, \psi, \phi)(y), \\
v &= G^2(f, g, h, \varphi, \psi, \phi)(y), \\
w &= G^3(f, g, h, \varphi, \psi, \phi)(y).
\end{align*}
\]  

(12)

Then, the solution of problem (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 operators can be described by

\[
\begin{align*}
G^1_{\theta}(f, g) &= \sum_{k=1}^{P} b_{1k}^{f} b_{1k}^{g} tr_k, \\
G^2_{\theta}(f, g) &= \sum_{k=1}^{P} b_{2k}^{f} b_{2k}^{g} tr_k,
\end{align*}
\]  

(13)

where the definitions of $b_{1k}^{f}$, $b_{2k}^{f}$, $b_{1k}^{g}$, $b_{2k}^{g}$ and $tr_k$ can be seen the illustration of (11). To reduce
the generalized error, we may add a bias $b_1, b_0 \in \mathbb{R}$ in the last stage:

$$
\begin{aligned}
G_1^\theta(f, g) &= \sum_{k=1}^P b_1^1 f_k^1 b_1^2 tr_k + b_1^0, \\
G_2^\theta(f, g) &= \sum_{k=1}^P b_2^1 f_k^1 b_2^2 tr_k + b_2^0.
\end{aligned}
$$

(14)

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

$$
\mathcal{L}(\theta) = \frac{1}{N} \sum_{k=1}^N (|u(x_k) - G_1^\theta(f, g)(x_k)|^2 + |v(x_k) - G_2^\theta(f, g)(x_k)|^2).
$$

(15)

We can now learning parameter \(\theta\) by minimizing the loss function (15) using 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).](image)

### 2.4. Physics-informed MIMOONets

A large number of paired data sets is required to solve PDEs system using the MIMOONets. However, data acquisition is expensive in many engineering applications and physical systems. Under the condition of sparse data, it is very important to introduce PINNs to train the MIMOONets by integrating known differential equation with label data in the loss function. We use automatic differentiation of the outputs of the MIMOONets respect to their input coordinates, and adopt an appropriate regularization mechanism to make the target output functions satisfy the PDE constraints.

For simplicity, we consider the problem (without causing confusion, we still use the preceding symbols)

$$
\begin{aligned}
\mathcal{L}_1[u(x); v(x)] &= f(x), & x \in D, \\
\mathcal{L}_2[u(x); v(x)] &= g(x), & x \in D, \\
B_1[u(x)] &= \varphi(x), & x \in \partial D, \\
B_2[v(x)] &= \psi(x), & x \in \partial D.
\end{aligned}
$$

(16)
The solutions $u, v$ can be expressed as

$$
\begin{align*}
    u &= G^1(f, g, h, \varphi, \psi, \phi)(y), \\
    v &= G^2(f, g, h, \varphi, \psi, \phi)(y).
\end{align*}
$$

For the problem (16), the loss function of a physics-informed MIMOONets is defined as follows

$$
L(\theta) = L_{\text{data}}(\theta) + L_{\text{physics}}(\theta),
$$

where

$$
\begin{align*}
    L_{\text{data}}(\theta) &= \frac{1}{N} \sum_{k=1}^{N} \left( |u(x_k) - G^1_\theta(f, g, \varphi, \psi)(x_k)|^2 + |v(x_k) - G^2_\theta(f, g, \varphi, \psi)(x_k)|^2 \right), \\
    L_{\text{physics}}(\theta) &= L_{\text{pde}1}(\theta) + L_{\text{pde}2}(\theta) + L_{\text{BC}1}(\theta) + L_{\text{BC}2}(\theta),
\end{align*}
$$

and

$$
\begin{align*}
    L_{\text{pde}1}(\theta) &= \frac{1}{N_f} \sum_{k=1}^{N_f} \left| L_1[G^1_\theta(f, g, \varphi, \psi)(x_k); G^2_\theta(f, g, \varphi, \psi)(x_k)] - f(x_k) \right|^2, \\
    L_{\text{pde}2}(\theta) &= \frac{1}{N_g} \sum_{k=1}^{N_g} \left| L_2[G^1_\theta(f, g, \varphi, \psi)(x_k); G^2_\theta(f, g, \varphi, \psi)(x_k)] - g(x_k) \right|^2, \\
    L_{\text{BC}1}(\theta) &= \frac{1}{N_{\varphi}} \sum_{k=1}^{N_{\varphi}} \left| B_1[G^1_\theta(f, g, \varphi, \psi)(x_k)] - \varphi(x_k) \right|^2, \\
    L_{\text{BC}2}(\theta) &= \frac{1}{N_{\psi}} \sum_{k=1}^{N_{\psi}} \left| B_2[G^2_\theta(f, g, \varphi, \psi)(x_k)] - \psi(x_k) \right|^2,
\end{align*}
$$

where $N$ is the number of initial data points. $N_f, N_g$ are the number of sample from the computational domain $\Omega$ for the PDEs. $N_{\varphi}$ and $N_{\psi}$ are the number of the boundary points for $u$ and $v$.

The loss function (18) is minimized by learning the parameters $\theta$ of the deep neural network. Sometimes in order to improve the accuracy of the solution or increase the convergence rate, we can introduce penalty parameters.

2.5. PDE-constrained optimization with physics-informed MIMOONets

For a given PDE-constrained optimal control problem such as (2), we use physics-informed MIMOONets to solve the optimization problems. The corresponding steps are as follows: Firstly, we turn PDE-constrained optimization problems (2) to optimality system (9), which consists of adjoint equation, state equation and optimality condition; Secondly, it is solved by physics-informed MIMOONets for optimality system.

3. Main results

In the following, to demonstrate the effectiveness of MIMOONs, we give numerical examples of elliptic, semi linear elliptic and parabolic control problems. Data driven MIMOONs or physically informed MIMOONs is used for numerical simulation.

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

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

(21)
subject to \[
\begin{cases}
-\Delta u - v = f, & \text{in } \Omega, \\
u = 0, & \text{on } \partial \Omega,
\end{cases}
\] (22)

where \(\Omega\) is a bounded domain, \(u : \Omega \to \mathbb{R}\) is the unknown temperature satisfying (22), \(u_d : \Omega \to \mathbb{R}\) is the given desired temperature, \(v\) is the unknown control function, \(f\) is source term in \(\Omega\), \(\alpha \geq 0\) 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 problem (21)-(22) can be transformed into the optimality system as follows

\[
\begin{cases}
-\Delta u - v = f, & \text{in } \Omega, \\
\Delta p + u = u_d, & \text{in } \Omega, \\
\alpha v + p = 0, & \text{in } \Omega, \\
u = 0, & \text{on } \partial \Omega, \\
p = 0, & \text{on } \partial \Omega.
\end{cases}
\] (23)

We use an MIMOONets to solve the PDEs system (23). The solution operator \(G^1\) and \(G^2\) of \(f\) and \(u_d\) can be represented as follows

\[
\begin{align*}
G^1_0(f, u_d) &= \sum_{k=1}^{p} b_1^k b_1^u d x_k, \\
G^2_0(f, u_d) &= \sum_{k=1}^{p} b_2^k b_2^u d x_k,
\end{align*}
\] (24)

where the branch network 1 and the branch network 2 are two separate 5-layer fully connected neural network (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

\[
L_{\text{data}}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \left( |u(x_k, y_k) - G^1_0(f, u_d)(x_k, y_k)|^2 + |p(x_k, y_k) - G^2_0(f, u_d)(x_k, y_k)|^2 \right).
\] (25)

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

\[
L(\theta) = L_{\text{data}}(\theta) + L_{\text{physics}}(\theta),
\] (26)

where

\[
L_{\text{physics}}(\theta) = L_{\text{pde1}}(\theta) + L_{\text{pde2}}(\theta) + L_{\text{BC1}}(\theta) + L_{\text{BC2}}(\theta),
\] (27)

and

\[
\begin{align*}
L_{\text{pde1}}(\theta) &= \frac{1}{N_d} \sum_{k=1}^{N_d} \left| \frac{\partial^2 G^1_0(f, u_d)(x_k, y_k)}{\partial x^2} - \frac{\partial^2 G^1_0(f, u_d)(x_k, y_k)}{\partial y^2} + G^2_0(f, u_d)(x_k, y_k) - f(x_k, y_k) \right|^2, \\
L_{\text{pde2}}(\theta) &= \frac{1}{N_d} \sum_{k=1}^{N_d} \left| \frac{\partial^2 G^2_0(f, u_d)(x_k, y_k)}{\partial x^2} + \frac{\partial^2 G^2_0(f, u_d)(x_k, y_k)}{\partial y^2} + G^1_0(f, u_d)(x_k, y_k) - u_d(x_k, y_k) \right|^2, \\
L_{\text{BC1}}(\theta) &= \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |G^1_0(f, u_d)(x_i, y_i)|^2, \\
L_{\text{BC2}}(\theta) &= \frac{1}{N_{\text{BC}}} \sum_{i=1}^{N_{\text{BC}}} |G^2_0(f, u_d)(x_i, y_i)|^2,
\end{align*}
\] (28)
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 $\Omega$, respectively. $N_{BC}$ is the number of the boundary points on the $\partial \Omega$.

To evaluate the loss, we randomly sample $N = 10000$ training points $(x_k, y_k) \in \Omega$, $N_f = N_u_d = 10000$ 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>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of $u$</th>
<th>Relative $L^2$ error of $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Relu</td>
<td>$(0.62 \pm 0.08)%$</td>
<td>$(0.39 \pm 0.05)%$</td>
</tr>
<tr>
<td>10000</td>
<td>Tanh</td>
<td>$(1.00 \pm 0.15)%$</td>
<td>$(0.98 \pm 0.12)%$</td>
</tr>
<tr>
<td>40000</td>
<td>Relu</td>
<td>$(0.14 \pm 0.03)%$</td>
<td>$(0.11 \pm 0.03)%$</td>
</tr>
<tr>
<td>40000</td>
<td>Tanh</td>
<td>$(0.45 \pm 0.05)%$</td>
<td>$(0.45 \pm 0.06)%$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of $u$</th>
<th>Relative $L^2$ error of $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Tanh</td>
<td>$(0.26 \pm 0.04)%$</td>
<td>$(0.28 \pm 0.05)%$</td>
</tr>
<tr>
<td>40000</td>
<td>Tanh</td>
<td>$(0.13 \pm 0.04)%$</td>
<td>$(0.14 \pm 0.05)%$</td>
</tr>
</tbody>
</table>

We use the Adam optimizer to train the networks, and take the corresponding learning rate 0.002. At the same time, we can choose different parameters to improve the accuracy of deep physics-informed MIMOONets method for the follow formal

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

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.
Figure 4: MIMONet iterations 10000 times with tanh. (a) Train loss. (b) The absolute value of the error of $u$. (c) The absolute value of the error of $p$.

Figure 5: Physics-informed MIMONet iterations 10000 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 $p$.

Figure 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. That 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

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

subject to

$$\begin{cases} -\Delta u + u^3 = v, & \text{in } \Omega, \\ u = 0, & \text{on } \partial\Omega. \end{cases}$$
The problem (30)-(29) leads to the following optimality system

\[
\begin{aligned}
-\Delta u + u^3 - v &= 0, \quad \text{in } \Omega, \\
\Delta p - 3u^2p + u &= u_d, \quad \text{in } \Omega, \\
\alpha v + p &= 0, \quad \text{in } \Omega, \\
u &= 0, \quad \text{on } \partial \Omega, \\
p &= 0, \quad \text{on } \partial \Omega.
\end{aligned}
\]  

(31)

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

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of $u$</th>
<th>Relative $L^2$ error of $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>Relu</td>
<td>(0.62± 0.08)%</td>
<td>(0.076± 0.011)%</td>
</tr>
<tr>
<td>100000</td>
<td>Tanh</td>
<td>(6.35± 0.35)%</td>
<td>(5.58± 0.22)%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of $u$</th>
<th>Relative $L^2$ error of $p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>Tanh</td>
<td>(2.40± 0.15)%</td>
<td>(1.60± 0.12)%</td>
</tr>
</tbody>
</table>

We use a multi-output operators network (MOONet) to solve the PDEs system (31). The corresponding solution operators $G_1$ and $G_2$ are shown below

\[
\begin{aligned}
G_1^k(u_d) &= \sum_{k=1}^{p} b^1_k u_d tr_k, \\
G_2^k(u_d) &= \sum_{k=1}^{p} b^2_k u_d tr_k,
\end{aligned}
\]

(32)

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 5 hidden layers with 300 neurons in each layer and 150 neurons in the output layer. The loss function for known data is similar to (25). Taken into account the physics informed MOONet and $\alpha = 1$, the corresponding loss function is expressed as follows

\[
\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{\text{data}}(\theta) + \lambda_1 \mathcal{L}_{\text{physics}}(\theta),
\]

(33)

where

\[
\mathcal{L}_{\text{physics}}(\theta) = \lambda_2 \mathcal{L}_{pde1}(\theta) + \lambda_3 \mathcal{L}_{pde2}(\theta) + \lambda_4 \mathcal{L}_{BC1}(\theta) + \lambda_5 \mathcal{L}_{BC2}(\theta),
\]

(34)

and

\[
\begin{aligned}
\mathcal{L}_{pde1}(\theta) &= \frac{1}{N_u} \sum_{k=1}^{N_u} \left| \frac{\partial^2 G_1^k(u_d)(x_k, y_k)}{\partial y^2} + \frac{\partial G_1^k(u_d)(x_k, y_k)}{\partial y} + (G_1^k(u_d)(x_k, y_k))^3 + G_2^k(u_d)(x_k, y_k) \right|^2, \\
\mathcal{L}_{pde2}(\theta) &= \frac{1}{N_u} \sum_{k=1}^{N_u} \left| \frac{\partial^2 G_2^k(f, u_d)(x_k, y_k)}{\partial y^2} + \frac{\partial G_2^k(f, u_d)(x_k, y_k)}{\partial y} - 3(G_1^k(u_d)(x_k, y_k))^2 G_2^k(u_d)(x_k, y_k) \right|^2, \\
\mathcal{L}_{BC1}(\theta) &= \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| G_1^i(f, u_d)(x_i, y_i) \right|^2, \\
\mathcal{L}_{BC2}(\theta) &= \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| G_2^i(f, u_d)(x_i, y_i) \right|^2.
\end{aligned}
\]

(35)
When we use physics informed MOONet, we choose \( N = N_1 = N_2 = 10000, N_{BC} = 400, \lambda_0 = \lambda_1 = \lambda_2 = 1, \lambda_3 = \lambda_4 = 300 \). Not using physics informed MOONet, we take \( N = 10000, \lambda_0 = 1 \), the others are 0. The sample data are collected by finite difference and sequential quadratic programming. The experimental results are shown in Table 3 and Table 4.

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. Optimal control of parabolic problem

We consider the following parabolic optimal control problem,

\[
\begin{align*}
\min J(u,v) &= \frac{1}{2} \int_0^T \int_{\Omega} (u - u_d)^2 \, dx \, dt + \frac{\alpha}{2} \int_0^T \int_{\Omega} v^2 \, dx \, dt, \\
\text{subject to} & \begin{cases} \\
\partial_t u - \Delta u - v = f, & \text{in } D, \\
u = 0, & \text{on } \partial D, \\
u(x,0) = \sin(\pi x), & \text{in } \Omega,
\end{cases}
\end{align*}
\]

(36)

where \( \Omega = [0,1] \), \( D = \Omega \times [0,T] \), \( u : D \to \mathbb{R} \) is the unknown term satisfying (37), \( u_d : D \to \mathbb{R} \) is the given desired temperature distribution, \( v \) is the unknown control function, \( f \) is source term in \( D \), \( \alpha \geq 0 \) is a regularization parameter. Here, we set \( D = [0,1] \), \( u_d(x,t) = \sin(\pi x)(t + 1) + \frac{1}{2}e^t\sin(\pi x) - \frac{1}{2}(e^t - e)^\pi \sin(\pi x) \), \( \alpha = 1 \), \( f(x,t) = \sin(\pi x) + \pi^2(t + 1)(\sin(\pi x) + \frac{1}{2}(e^t - e)\sin(\pi x)) \).

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 problem (36)-(37) can be transformed into the optimality system,

\[
\begin{align*}
\partial_t u - \Delta u - v &= f, & \text{in } D, \\
\partial_t p + \Delta p + u &= u_d, & \text{in } D, \\
\alpha v + p &= 0, & \text{in } D, \\
\varphi(x) &= u(x,0) = \sin(\pi x), & \text{in } \Omega, \\
u(x,t) &= 0, & \text{on } \partial D, \\
p(x,T) &= 0, & \text{in } \Omega.
\end{align*}
\]

(38)

For the PDEs system (38), we use MIMOOONets to solve it. The operator \( G^1 \) and \( G^2 \) can be learn from the source term \( f, u_d \) and \( \varphi \). Their representations are as follows

\[
\begin{align*}
G^1_0(f,u_d,\varphi) &= \sum_{k=1}^p b_{1k}^{1}b_{1k}^{u_d}b_{1k}^{\varphi} tr_k, \\
G^2_0(f,u_d,\varphi) &= \sum_{k=1}^p b_{2k}^{1}b_{2k}^{u_d}b_{2k}^{\varphi} tr_k.
\end{align*}
\]

(39)

Here, we select 100 sensors points for input functions \( f, u_d \) and \( \varphi \). 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

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \left( |u(x_k, t_k) - G^1_\theta(f, u_d, \varphi)(x_k, t_k)|^2 + |p(x_k, t_k) - G^2_\theta(f, u_d, \varphi)(x_k, t_k)|^2 \right). \tag{40}$$

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

$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{\text{data}}(\theta) + \lambda_1 \mathcal{L}_{\text{physics}}(\theta), \tag{41}$$

where

$$\mathcal{L}_{\text{physics}}(\theta) = \lambda_2 \mathcal{L}_{\text{pde1}}(\theta) + \lambda_3 \mathcal{L}_{\text{pde2}}(\theta) + \lambda_4 \mathcal{L}_{\text{IC}}(\theta) + \lambda_5 \mathcal{L}_{\text{TC}}(\theta) + \lambda_6 \mathcal{L}_{\text{BC}}(\theta), \tag{42}$$

and

$$\mathcal{L}_{\text{pde1}}(\theta) = \frac{1}{N_f} \sum_{k=1}^{N_f} \left| \frac{\partial G^1_\theta(f, u_d, \varphi)(x_k, t_k)}{\partial x} \right|^2 + \left| \frac{\partial^2 G^1_\theta(f, u_d, \varphi)(x_k, t_k)}{\partial x \partial t} \right|^2 + \| \mathbf{G}^1_\theta(f, u_d, \varphi)(x_k, t_k) - \mathbf{f}(x_k, t_k) \|^2,$$

$$\mathcal{L}_{\text{pde2}}(\theta) = \frac{1}{N_{u_d}} \sum_{k=1}^{N_{u_d}} \left| \frac{\partial G^2_\theta(f, u_d, \varphi)(x_k, t_k)}{\partial x} \right|^2 + \left| \frac{\partial^2 G^2_\theta(f, u_d, \varphi)(x_k, t_k)}{\partial x \partial t} \right|^2 + \| \mathbf{G}^2_\theta(f, u_d, \varphi)(x_k, t_k) - u_d(x_k, t_k) \|^2,$$

$$\mathcal{L}_{\text{IC}}(\theta) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left| G^1_\theta(f, u_d, \varphi)(x_i, t_i) - \varphi(x_i, t_i) \right|^2,$$

$$\mathcal{L}_{\text{TC}}(\theta) = \frac{1}{N_{TC}} \sum_{i=1}^{N_{TC}} \left| G^2_\theta(f, u_d, \varphi)(x_i, t_i) - \varphi(x_i, t_i) \right|^2,$$

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| G^3_\theta(f, u_d, \varphi)(x_i, t_i) \right|^2. \tag{43}$$

To calculate the value of the loss function, we take the random sample of initial data points $N = 10000$, the numbers of the residual training points $N_f = N_{u_d} = 10000$, 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 equation (40) and (41). The learning rate is 0.002. The experimental results are shown in Table 5, Table 6, and Figure 6-8.

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of u</th>
<th>Relative $L^2$ error of p</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Relu</td>
<td>(0.11± 0.04)%</td>
<td>(0.24± 0.05)%</td>
</tr>
<tr>
<td>10000</td>
<td>Tanh</td>
<td>(0.60± 0.10)%</td>
<td>(1.60± 0.13)%</td>
</tr>
<tr>
<td>40000</td>
<td>Relu</td>
<td>(0.054± 0.006)%</td>
<td>(0.10± 0.04)%</td>
</tr>
<tr>
<td>40000</td>
<td>Tanh</td>
<td>(0.40± 0.05)%</td>
<td>(0.88± 0.10)%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Activation function</th>
<th>Relative $L^2$ error of u</th>
<th>Relative $L^2$ error of p</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>Tanh</td>
<td>(0.52± 0.12)%</td>
<td>(4.9± 0.27)%</td>
</tr>
<tr>
<td>40000</td>
<td>Tanh</td>
<td>(0.32± 0.08)%</td>
<td>(4.7± 0.21)%</td>
</tr>
</tbody>
</table>
We find that both the deep MIMOONets method and the deep physics-informed MIMOONets method are effective for parabolic optimal control problem. That the activation function Relu converges faster than Tanh with the MIMOONets method. We also find that the physics-informed MIMOONets not only attains comparable accuracy to the original MIMOONets, but also satisfies the underlying PDEs constraint.

4. Discussion

This work provides a novel deep learning framework that enables the construction of fast surrogates for solving the PDEs optimal control problem using MIMOONets and physics-informed MIMOONets. We first use the formal Lagrange method to transform the optimal control problem into an optimal system consisting of state equations and adjoint equations. Then, we apply deep MIMOONets or physics-informed MIMOONets to solve the optimal system. In contrast to previous approaches, the proposed physics-informed MIMOONets surrogates can be trained by a little paired input-output data, which requires the evaluation of expensive simulators or experiments. Their effectiveness is tested by tackling linear elliptic, semi-linear elliptic and parabolic PDE-constrained...
Figure 8: Physics-informed MIMONet iterations 10000 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 $p$.

Optimization problems. The merit of the MIMOONets framework is its flexibility and faster of implementation, compared with the traditional method. The optimal control problem can be obtained more easily by a wider audience.

Furthermore, our MIMOONet method is very suitable for solving differential equations. This work can be used to solve complex system design and control problems in practical applications, especially for fluid mechanics problems.

Although our proposed method has many advantages, it cannot reach the precision of the existing numerical methods in many cases. It needs further study for complex problems. The proposed method can be extended to more practical applications, to make it possible to accurately solve more complex partial differential equations, such as multi-scale solutions, multi-phase flows, large space-time domains, etc. In general, with the increase of dimensionality, the speed of traditional numerical methods increases exponentially. However, the deep learning method can solve these problems to some extent. We are still in the early stages of using machine learning to solve complex problems. Therefore, the design of a more effective neural network architecture is the key to release the full potential of MIMOONets with physical information for real-time solving more realistic PDE constrained optimization problems.

References


Appendix A. Supplementary Visualizations of Elliptic Optimal Control Problem

We present some numerical images for solving elliptic optimal control problem (21)-(22) using deep MIMOONets and physics-informed MIMOONets framework. The data-driven results are shown in Figure A.9-Figure A.14. 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.15.

Figure A.9: MIMONet iterations 10000 times with Relu for elliptic constraint optimal control problem. (a) The error of $u$: $u - G^1_0(f, u_d)$. (b) The error of $p$: $p - G^2_0(f, u_d)$.

Figure A.10: MIMONet iterations 10000 times with Tanh for elliptic constraint optimal control problem. (a) The error of $u$: $u - G^1_0(f, u_d)$. (b) The error of $p$: $p - G^2_0(f, u_d)$.
Figure A.11: MIMONet iterations 40000 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 $u$: $u - G_1^1(f, u_d)$. (e) The error of $p$: $p - G_2^1(f, u_d)$.

Figure A.12: Physics-informed MIMONet iterations 10000 times for elliptic constraint optimal control problem ($\lambda_0 = \lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 100$). (a) The error of $u$: $u - G_1^3(f, u_d)$. (b) The error of $p$: $p - G_2^3(f, u_d)$.
Figure A.13: MIMONet iterations 40000 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 $u$: $u - G_1^u(f, u_d)$. (e) The error of $p$: $p - G_2^p(f, u_d)$.

Figure A.14: Physics-informed MIMONet iterations 40000 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 $u$: $u - G_1^u(f, u_d)$. (e) The error of $p$: $p - G_2^p(f, u_d)$.
Appendix 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 Figure B.16-Figure B.21. 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 the Figure B.22.
Figure B.17: MIMONet iterations 10000 times with Tanh for Parabolic-constraint optimal control problem. (a) The error of $u$: $u - G^1_\theta(f, u_d, \varphi)$. (b) The error of $p$: $p - G^2_\theta(f, u_d, \varphi)$.

Figure B.18: MIMONet iterations 40000 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 $u$: $u - G^1_\theta(f, u_d, \varphi)$. (e) The error of $p$: $p - G^2_\theta(f, u_d, \varphi)$. 
Figure B.19: MIMONet iterations 40000 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 $u$: $u - G^1_θ(f, u_d, ϕ)$. (e) The error of $p$: $p - G^2_θ(f, u_d, ϕ)$.

Figure B.20: Physics-informed MIMONet iterations 10000 times for Parabolic-constraint optimal control problem ($λ_i = 1, i = 0, 1, 2, 3; λ_j = 100, j = 4, 5, 6$, there is no initial data). (a) The error of $u$: $u - G^1_θ(f, u_d, ϕ)$. (b) The error of $p$: $p - G^2_θ(f, u_d, ϕ)$. 

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4353535
Figure B.21: Physics-informed MIMONet iterations 40000 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 $u$: $u - G_1^1(f, u_d, \varphi)$. (e) The error of $p$: $p - G_2^2(f, u_d, \varphi)$.

Figure B.22: Physics-informed MIMONet iterations 40000 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 $u$: $u - G_1^1(f, u_d, \varphi)$. (e) The error of $p$: $p - G_2^2(f, u_d, \varphi)$.