Solving PDE-Constrained Control Problems Using Operator Learning

Rakhoon Hwang[*†], [1] **Jae Yong Lee**[*†], [2] **Jin Young Shin**[*], [3] **Hyung Ju Hwang**[‡], [3]

[1]POSTECH Institute of Artificial Intelligence
[2]Center for Artificial Intelligence and Natural Sciences, Korea Institute for Advanced Study
[3]Department of Mathematics, Pohang University of Science and Technology
hrh0125@postech.ac.kr, jaeyong@kias.re.kr, {sjy6006, hjhwang}@postech.ac.kr

Abstract

The modeling and control of complex physical systems are essential in real-world problems. We propose a novel framework that is generally applicable to solving PDE-constrained optimal control problems by introducing surrogate models for PDE solution operators with special regularizers. The procedure of the proposed framework is divided into two phases: solution operator learning for PDE constraints (Phase 1) and searching for optimal control (Phase 2). Once the surrogate model is trained in Phase 1, the optimal control can be inferred in Phase 2 without intensive computations. Our framework can be applied to both data-driven and data-free cases. We demonstrate the successful application of our method to various optimal control problems for different control variables with diverse PDE constraints from the Poisson equation to Burgers' equation.

Introduction

The modeling of physical systems to support decision making and solve the optimal control problem is a key problem in many industrial, economical, and medical applications. Such systems can be described mathematically through partial differential equations (PDEs). In this regard, solving PDE-constrained optimal control problems provides a control law for a complex system governed by PDEs. A PDE-constrained optimal control problem has been successfully used in many applications: shape optimization (Haslinger and Mäkinen 2003; Sokolowski and Zolésio 1992), mathematical finance (Bouchouev and Isakov 1999; Egger and Engl 2005), and flow control (Gunzburger 2002).

As computing power increases and optimization technologies significantly improve, many researchers have studied algorithms and computational methods that are accurate, efficient, and applicable for complex physical systems. In control theory, adjoint methods are some of the most common approaches to handle this problem. Adjoint methods provide an efficient way to compute gradients that appear in optimization problems (Lions 1971; Pironneau 1974; Tröltzsch 2010), and its variants have been applied to many different areas of science. However, adjoint-based iterative schemes, such as shooting methods, suffer from computational costs because of their trial-and-error nature. In addition, iterative schemes are often sensitive to the initial guesses to the solutions.

Deep learning methods have recently derived major techniques for scientific computations, including PDEs or optimization problems (Raissi, Perdikaris, and Karniadakis 2019). In particular, solving a family of parametrized PDEs requires networks to approximate a function-to-function mapping or operator. In (Li et al. 2020a), (Li et al. 2020b), (Lu, Jin, and Karniadakis 2019), (Zhu and Zabaras 2018), and (Zhu et al. 2019), the authors utilized neural networks to learn a mapping from the parameters (e.g. initial or boundary) of a PDE to the corresponding solution. Such models are used as a surrogate model to solve various problems such as uncertainty quantification (Zhu and Zabaras 2018; Zhu et al. 2019) and an inverse problem (Li et al. 2020a).

Recent studies have been conducted to solve PDE-constrained optimal control problems through a deep learning approach. In (Holl, Koltun, and Thuerey 2020), the authors proposed a predictor-corrector scheme for long-term fluid dynamics control, combining neural networks with a differentiable solver. In (Rabault et al. 2019), the authors experimentally showed that active flow control for vortex shedding and a drag reduction can be achieved through model-free reinforcement learning. Model-free methods are often known to require numerous interactions with the environment to search for an optimal policy. A hand-designed reward is necessary for each problem, which usually involves deep prior knowledge of complex physical systems. In addition, this approach requires numerical solvers during every iteration. These make such approaches inapplicable to real-world problems.

We propose an alternative framework to solve PDE-constrained optimal control problems. Our method is divided into two phases: solution operator learning for PDE constraints (Phase 1) and searching for optimal control (Phase 2). During Phase 1, a neural network with a reconstruction structure is trained to approximate the PDE solutions. The optimal control problem can then be solved using the trained network with a reconstruction regularizer during Phase 2. The proposed method has the following four main

[*]These authors contributed equally.
[†]Work performed while at Pohang University of Science and Technology.
[‡]Corresponding author.

contributions compared to the existing approaches in the literature:

Simple but effective regularizer We introduce a novel regularizer to solve PDE-constrained optimal control problems effectively. We employ a reconstruction loss as a regularizer, which enables the control variable to converge correctly in Phase 2.

Application to various PDE-constrained control problems We propose a non-problem-specific methodology for PDE-constrained optimal control problems. Our method is applied to problems with various types of PDEs: elliptic (Poisson, Stokes), hyperbolic (wave), non-linear parabolic (Burgers') equations. In addition, the control variable can be any type, such as the initial condition, the boundary condition, and a parameter in the governing equation.

Computational efficiency Our surrogate model approximates a PDE solution with sufficiently high accuracy to search optimal controls while taking significantly less time for inference. Unlike the adjoint-based iterative methods, which require a heavy computation by PDE solvers for every iteration, our framework is useful when computation resources are limited, or a fast inference is required.

Flexibility Our framework does not depend on the presence or absence of data. In the case that the full data pairs of control-to-state are available, the surrogate model for the PDE solution operator can be trained by a supervised loss during Phase 1. Meanwhile, if the physical systems are described in the form of PDEs, the residual norm of the PDE can be used to train the surrogate model without simulation data.

Related Work

Deep learning and PDEs There are two mainstream deep learning approaches to approximate solutions to the PDEs, i.e., using neural networks directly to parametrize the solution to the PDE and learning operators from the parameters of the PDEs to their solutions. A physics-informed neural network (PINN) was introduced in (Raissi, Perdikaris, and Karniadakis 2019), which learns the neural network parameters to minimize the PDE residuals in the least-squares sense. In (Nabian and Meidani 2018), (Son et al. 2021), and (Weinan and Yu 2018), the authors suggested using a modified residual of the PDEs, and in (Han, Jentzen, and Weinan 2018) and (Sirignano and Spiliopoulos 2018), the authors showed the possibility of solving high-dimensional PDEs. In (Hwang et al. 2020), (Jo et al. 2020), and (Sirignano and Spiliopoulos 2018), the authors prove a theorem on the approximation power of the neural network for an analytic solution to the PDEs. Next, we introduce another approach, operator learning, which is more closely related to our research.

Operator learning Operator learning using neural networks has been studied to approximate a PDE solution operator, which is nonlinear and complex in general. A universal approximation theorem for the operator was introduced in (Chen and Chen 1993). Based on these results, in (Lu, Jin, and Karniadakis 2019), the authors developed a neural network called *DeepONet*. In addition, mesh-based methods using convolutional neural networks (CNN) have been studied in many papers (Bhatnagar et al. 2019; Guo, Li, and Iorio 2016; Khoo, Lu, and Ying 2017; Zhu and Zabaras 2018). These studies used labeled data to train the operator networks. In (Bhatnagar et al. 2019) and (Guo, Li, and Iorio 2016), the authors used a CNN as a surrogate model of a computational fluid dynamics (CFD) solver. The authors showed that the surrogate models have a greater benefit in terms of speed than a CFD solver. In (Khoo, Lu, and Ying 2017) and (Zhu and Zabaras 2018), the authors developed the surrogate model for uncertainty quantification problems. Furthermore, the authors of (Zhu et al. 2019) proposed *physics-constrained surrogate loss*, which can be calculated without labeled data. Li et al. proposed a resolution-invariant neural operator using a graph neural network (Li et al. 2020b) and the fast Fourier transform (Li et al. 2020a).

PDE-constrained control problem The most common approach in control theory is adjoint-based methods which give an efficient way to compute the gradient of forward maps with respect to the parameters (Borrvall and Petersson 2003; Christofides and Chow 2002; Lions 1971; McNamara et al. 2004; Pironneau 1974; Tröltzsch 2010). Several studies have suggested learning-based methods for control problems, such as (de Avila Belbute-Peres et al. 2018), (Hafner et al. 2019), and (Watter et al. 2015). Regarding control problems associated with PDEs, the authors in (Holl, Koltun, and Thuerey 2020) used a differentiable PDE solver to plan optimal trajectories and control fluid dynamics. They experimentally showed that their model enables long-term control with a fast inference time. Flow control problems, including vortex shedding and a drag reduction, were solved using reinforcement learning (Rabault et al. 2019) or Koopman operator theory (Morton et al. 2018). One of the most interesting PDE-constrained optimization problems is an inverse problem, specifying unknown parameters in PDE systems given the observed data. There have recently been attempts to solve the problem by penalizing the parameter space or using a probabilistic approach (Jo et al. 2020; Liang, Lin, and Koltun 2019; Ma et al. 2019; Pilozzi et al. 2018; Ren, Padilla, and Malof 2020). In particular, the approach in (Ren, Padilla, and Malof 2020) is similar to our study in that it learns the forward map first, but does not target the PDE problems. The authors in (Li et al. 2020a) showed that the PDE solution operator approximated by neural networks can be used in a Bayesian inverse problem.

Methodology

In this paper, we aim to solve PDE-constrained control problems. Let M be a reflexive Banach space and U and V be Banach spaces. Formally, a PDE-constrained optimization problem can be written as follows:

$$\min_{u\in U, m\in M} J(u,m) \quad \text{subject to} \quad F(u,m)=0 \qquad (1)$$

where $J : U \times M \to \mathbb{R}$ is an objective function of interest, and $F : U \times M \to V$ is a system of PDEs, which governs the physics of the problem, possibly including initial and boundary conditions. Each space is a space of functions defined in a certain spatial or time domain. Here, u is called a *state* or PDE solution, and m is a *control* input. In the presence of control constraints, the problem is restricted to a set of admissible controls by $M_{ad} \subset M$, which is often assumed to be closed and convex. We remark that the control input can be configured in various forms, such as the values of a source term in a governing equation, or of the initial or boundary conditions. Our goal is to propose a general neural network based framework that is applicable to any type of PDEs and control inputs.

The optimal control of the Poisson equation can be considered a motivating example, a problem of specifying an unknown heat source to achieve a desired temperature profile. In this case, the control input m indicates the values of the source term in the governing equation. The corresponding optimization problem is as follows:

$$\min_{u \in H_0^1(\Omega), m \in L^2(\Omega)} \frac{1}{2} \int_\Omega (u - u_d)^2 \, dx + \frac{\alpha}{2} \int_\Omega m^2 \, dx \tag{2}$$

subject to the Poisson equation with zero Dirichlet boundary conditions

$$\begin{cases} -\Delta u - m = 0 & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{3}$$

where Ω is the domain of interest, the state $u : \Omega \to \mathbb{R}$ is the unknown temperature, $u_d : \Omega \to \mathbb{R}$ is the given desired temperature, α is a penalty parameter, and $m : \Omega \to \mathbb{R}$ is the control function. Note that the penalty term should be distinguished from the regularization, which will be discussed in Section . For practical purpose, m is often restricted to M_{ad}, in which additional inequality constraints $m_a(x) \leq m(x) \leq m_b(x)$ are imposed. It is well-known that this problem is well-posed and has a unique solution (Hinze et al. 2008; Tröltzsch 2010).

We remark from the above example that the PDE solution u can be thought as a function of m with implicit relations $F(u, m) = 0$. In many cases, handling such complex, possibly nonlinear PDE constraints becomes the main difficulty when solving optimal control problems. One possible approach is to use surrogate models for PDE systems, approximating the explicit control-to-state mapping. For a concise notation, we denote the explicit solution expression by $u(m)$ and consider the reduced optimization problem

$$\min_{m \in M_{ad}} \tilde{J}(m) \tag{4}$$

with the reduced objective function $\tilde{J}(m) := J(u(m), m)$. This enables us to convert the constrained optimization problem into an unconstrained problem. We can then apply gradient-based optimization algorithms to obtain a locally optimal solution.

In this study, we approximate the solution operators of PDE constraints through neural networks (Phase 1) and use them to search optimal controls for the given problems (Phase 2) through gradient descent. The two phases are described in detail in Section and Section , and summarized in Figure 1. In Section , error estimates of the optimal controls are discussed under certain assumptions. Further, in Section , we suggest modified architectures that are particularly efficient for time-dependent PDE constraints.

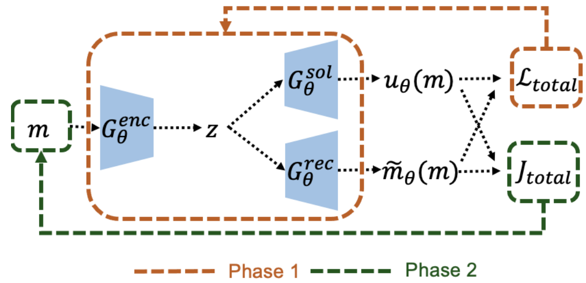

Figure 1: Overview of our autoencoder model. During Phase 1, the parameter θ is updated, and during Phase 2, the control input m is updated.

Phase 1: Solution Operator Learning for PDE Constraints

In the first step, a neural network is trained as a surrogate model for the PDE solution operator. We discretize the spatial domain into a uniform mesh to convert state u and control input m into image-like data. The surrogate model is then trained as an image-to-image regression. We suggest a variant autoencoder specialized for control problems. Our baseline model consists of a single encoder G_θ^{enc} for control input m and two decoders G_θ^{sol}, G_θ^{rec} corresponding to the state $u_\theta(m) = (G_\theta^{sol} \circ G_\theta^{enc})(m)$ and reconstruction $\tilde{m}_\theta(m) = (G_\theta^{rec} \circ G_\theta^{enc})(m)$ where θ is the set of all network parameters (Phase 1 in Figure 1). The reconstruction $\tilde{m}_\theta(m)$ plays an essential role in Phase 2. This will be described in Section . Our method can be applied to the following two scenarios, *data-driven* and *data-free*.

Data-driven scenario In the case that the full data pairs of control-to-state are available, a supervised loss is a natural choice:

$$\mathcal{L}_{sup} = \frac{1}{N} \sum_{i=1}^{N} L(u_\theta(m_i), u_i), \tag{5}$$

where $\{(m_i, u_i)\}_{i=1,\ldots,N}$ is the observed data, and L is a measure of the difference between two vectors. In our experiments, we used the L^2-relative error for L, namely $L(u, \tilde{u}) := \|u - \tilde{u}\|_2 \,/\, \|\tilde{u}\|_2$

Data-free scenario In most real-world scenarios, complete data pairs of the control-to-state cannot be accessed because of expensive simulations. In these situations, one may utilize prior knowledge about the system of interest, which is often described in the form of PDEs. In that case, inspired by (Zhu et al. 2019), the surrogate model can be trained by minimizing the residual norm of the PDE:

$$\mathcal{L}_{res} = \frac{1}{N} \sum_{i=1}^{N} \|F(u_\theta(m_i), m_i)\|^2, \tag{6}$$

where $\|\cdot\|$ is the norm in the Banach space V. Here, we sampled the inputs $\{m_i\}_{i=1,\ldots,N}$ in a set of admissible controls M_{ad}. This loss function imposes the physical law of the PDE constraint $F(u,m) = 0$ to the surrogate model. For example, in the case of the Poisson equation (3), the residual norm of $F(u,m)$ can be expressed as

$$\|F(u,m)\| = \|-\Delta u - m\|_{L^2(\Omega)} + \|u\|_{L^2(\partial\Omega)}. \quad (7)$$

When calculating residual $F(u_\theta(m_i), m_i)$ in $\mathcal{L}_{res}$, the spatial gradients can be approximated efficiently using a convolutional layer with a fixed kernel which consists of the finite difference coefficient, and the boundary condition can be enforced exactly (See Appendix C.1).

Combining Eq. (5) or Eq. (6) with the reconstruction loss, $\mathcal{L}_{rec} = \frac{1}{N}\sum_{i=1}^{N} L(\tilde{m}_\theta(m_i), m_i)$, we used the total loss as $\mathcal{L}_{total} = \mathcal{L}_{sup} + \lambda_1 \mathcal{L}_{rec}$ or $\mathcal{L}_{res} + \lambda_1 \mathcal{L}_{rec}$ where λ_1 is a hyperparameter.

Phase 2: Searching for Optimal Control

After the surrogate model is trained during Phase 1, the learned parameter θ^* is fixed. We cosider m as a learnable parameter and denote the objective function as

$$J_{obj}(m) := J(u_{\theta^*}(m), m). \quad (8)$$

Because the surrogate model is differentiable, the gradient of the objective function with respect to the control input can be directly calculated. Then, J_{obj} can be used as a loss function. If only J_{obj} is minimized, one issue is that control input m may converge to local optima outside the training domain of the surrogate model. This causes performance degradation of the surrogate model. To handle this situation, we employ the reconstruction loss as a regularizer, i.e., $J_{rec}(m) := L(\tilde{m}_{\theta^*}(m), m)$. J_{rec} and $\mathcal{L}_{rec}$ in Phase 1 are similar, but different in that θ is updated during Phase 1 while m is updated during Phase 2 with the fixed θ^*.

Its regularizing effect can be interpreted in perspective of the variational autoencoder (VAE) (Kingma and Welling 2013). For this purpose, we consider the control input m and latent variable z as random vectors with prior density $p(z)$. A graphical model $p(m,z) = p(m|z)p(z)$ is given and induces an inequality given by

$$\begin{aligned} &-\log p(m) \\ &\quad \le \underbrace{-\mathbb{E}_{q(z|m)}\left[\log p(m|z)\right]}_{\text{reconstruction}} + KL\left(q(z|m) | p(z)\right), \end{aligned} \quad (9)$$

where $q(z|m)$ is an approximation of the posterior. $p(m)$ can be thought as the distribution of m sampled during Phase 1 training. In this regard, we expect that minimizing the upper bound in Eq. (9) during Phase 2 keeps the likelihood $p(m)$ large enough, which implies that the updated control input m keeps belonging to the training domain. In our experiments, we use a plain autoencoder, which models $q(z|\tilde{m})$ as a dirac distribution. In this case, the Kullback–Leibler divergence (KL) term is a constant with respect to m and only the reconstruction term remains in the upper bound, which coincides with $J_{rec}(m)$ when L is the L^2-loss and $p(m|z)$ is modeled through a Gaussian distribution. It implies that the role of the reconstruction regularizer in Phase 2 is to hold the control input m in the region where the operator network works well. The experiments in Section show that the regularizer term greatly improves the performance of the optimal control learning, especially in Figure 2.

Consequently, we set the following loss function to train the control problem:

$$J_{total}(m) = J_{obj}(m) + \lambda_2 J_{rec}(m), \quad (10)$$

where λ_2 is a hyperparameter.

Theoretical Connection from Phase 1 to Phase 2

Under some mild assumptions regarding a function space of neural network approximators, we derive the error estimates of the optimal control during Phase 2 in terms of the error that occurred during Phase 1. This provides the theoretical connection between the two separate phases. Although the following discussion is focused on our motivating example, i.e., a tracking-type problem with the Poisson equation, it can be adapted to other problems in a similar fashion.

In Eq. (3), we denote an exact solution operator by S, which satisfies $F(Sm, m) = 0$, and an approximate solution operator by S_h. In addition, let m^* be an exact optimal solution, and m_h^* be the optimal solution inferred by our method. One of the main assumptions necessary to present our proposition is the Lipschitz continuity of the surrogate model S_h. Such an assumption has been well addressed in recent papers such as (Fazlyab et al. 2019) amd (Virmaux and Scaman 2018). We then derive the following proposition:

Proposition 1. *If $S_h : M \to U$ is Lipschitz continuous and approximated with error $\|S - S_h\|_2 < \epsilon$, then the L^2 error for optimal control is estimated as*

$$\|m^* - m_h^*\|_2 < C\alpha^{-1}(1 + \alpha^{-1/2})\, \|u_d\|_2\, \epsilon$$

for a constant C.

The observation u_d and the penalty parameter α are the prescribed values. Meanwhile, $\|S - S_h\|_2$, which is defined by $\|S - S_h\|_2 := \sup_{m \in L^2(\Omega),\ \|m\| \le 1} \|(S - S_h)m\|_2$, can be thought of as a measurement of approximation and generalization of the operator learning. A sketch of proof is as follows: First we define the discrete version of the given optimization problem, which attains an optimal solution m_h^*. Then, by subtracting and modifying the first-order optimality condition for each problem, we can derive a proper upper bound for the error $\|m^* - m_h^*\|_2$. The detailed statements are given in Appendix A.

The proposition provides the error estimates for the approximated optimal control input m_h^* obtained using our method in Phase 2. If the surrogate model S_h is trained to minimize the operator norm $\|S - S_h\|_2$ within the given error tolerance ϵ, then the error for optimal control $\|m^* - m_h^*\|_2$ can be estimated as in the proposition. This provides the connection between the two separate phases.

Application to Time-dependent PDEs

We explain how our model can be extended and applied to time-dependent PDEs. We consider the time-dependent PDEs, in which the system can be written as

$$\frac{\partial}{\partial t} u(t,\cdot) = F(u(t,\cdot), m(t,\cdot)) \quad (11)$$

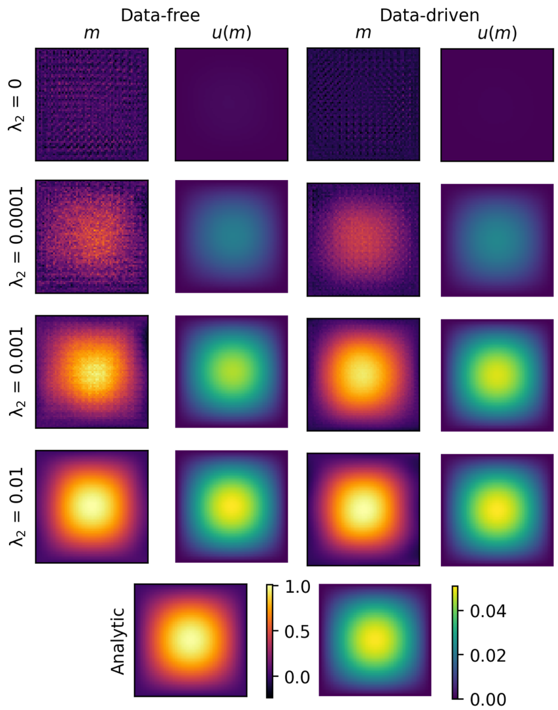

Figure 2: Control results according to different λ_2. First two columns are the results from data-free setting. The remaining columns are the results from data-driven setting. The last row shows the analytic optimal m^* and u^*.

for $t \in [0, T]$. We discretize the time-dependent PDE as follows:

$$u^{t+\Delta t} = \mathcal{F}(u^t, m^t) \tag{12}$$

where $u^t(\cdot) := u(t, \cdot) \in U$ and $m^t(\cdot) := m(t, \cdot) \in M$, in which $t = 0, \Delta t, ..., (n-1)\Delta t$ with $T = n\Delta t$. Recently, the authors of (Lusch, Kutz, and Brunton 2018) employed a deep learning approach to discover representations of Koopman eigenfunctions from data. One of the key ideas in (Lusch, Kutz, and Brunton 2018) is that the time evolution of the eigenfunctions proceeds on the latent space between encoder and decoder. Our method can be extended for the time-dependent PDEs with inspiration from the idea in (Lusch, Kutz, and Brunton 2018).

We use two autoencoders H_Θ and G_Θ for the state u^t and control input m^t. The transition network T_Θ predicts the next time latent state $v^{t+\Delta t}$ from the latent state $v^t = H^{enc}(u^t)$ under the influence of the latent variable $g^t = G^{enc}(m^t)$. Therefore, the model propagates the state u^t to the next time state $u^{t+\Delta t}$ under the influence of the control input m^t. The model can be used repeatedly n times to generate a desired state u^T from the given initial state u^0 and the given control inputs m^0, $m^{\Delta t}$, ..., $m^{(n-1)\Delta t}$. The loss functions for Phases 1 and 2, and other details of the extended model are given in Appendix B.

Experiment

In this section, we evaluate our method to handle a variety of PDE-constrained optimal control problems through four different examples. We first target the source control of the Poisson equation to illustrate the basic idea of our methodology. Two cases for operator learning, data-driven and data-free, will be considered to confirm the flexibility of our method. Next, we study the boundary control of the Stokes equation and the inverse design of the wave equation. Finally, the modified model architecture proposed in Section will be verified using a nonlinear time-dependent PDE, Burgers' equation. For each experiment, we compare our method with the adjoint-based iterative method that is typically used for solving the optimal control problem effectively.

In this section, we focus on the results of optimal control during Phase 2. For the results of Phase 1, we reported the values of relative errors on test data in each experiment below. The visual results of the trained solution operator during Phase 1 are described in each subsection of Appendix C. It shows that the surrogate models for the PDE solution operator in each control problem are well approximated with small relative errors, which is sufficient for use in Phase 2. Data generation and other details are given in Appendix C.

Source control of the Poisson equation The Poisson equation is an elliptic PDE, also referred to a steady state heat equation. We consider the optimal control problem with control objective function (2) subject to the Poisson equation under the Dirichlet boundary condition, which is described in equation (3). This is a fundamental PDE-constrained control problem. The problem aims to control the source term m to make $u(m)$ similar to u_d with L^2 penalization. During our experiment, $(x, y) \in \Omega = [0, 1] \times [0, 1]$, $\alpha = 10^{-6}$, and $u_d = \frac{1}{2\pi^2} \sin \pi x \sin \pi y$. In this case, the analytic optimal control m^* and the corresponding state u^* are given by

$$m^* = \frac{1}{1 + 4\alpha\pi^4} \sin \pi x \sin \pi y, \quad u^* = \frac{1}{2\pi^2} m^*. \tag{13}$$

Figure 7 in Appendix C.1 shows the training results for the solution operator during Phase 1. The solution operators are well approximated for both cases, data-driven (supervised loss, Eq. (5)) and data-free (residual loss, Eq. (6)). The relative errors for test data are 0.0016 and 0.0080, respectively.

To verify the regularization effect of J_{rec}, we observe the change in optimal control obtained during Phase 2 depending on the regularizer coefficient λ_2. As shown in Figure 2, with $\lambda_2 = 0$, i.e., no regularizer, the obtained optimal control is irregular and fails to converge to the analytic optimal m^*. As λ_2 increases, the optimal control becomes smoother and closer to m^*. This shows that the regularizer term greatly improves the performance of the optimal control learning. It makes the control input m remain in the training domain where the operator network works well. In both cases, data-free and data-driven, the optimal control m from our method is sufficiently close to analytic optimal control m^* when $\lambda_2 = 0.01$. This phenomenon agrees with our expectation of the regularization effect discussed in Section .

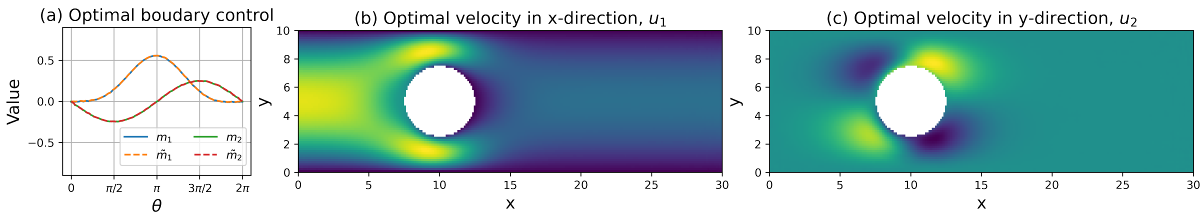

Figure 3: Illustration of the boundary control of the Stokes equation. (a) The control inputs m_1 and m_2 which corresond to the Dirichlet boundary values on the circle. The solid lines indicate the optimal control obtained by our method, and the dashed lines are the reconstructed control. (b), (c) The corresponding velocity fields u_1 and u_2 that minimize the drag energy.

Boundary control of the stationary Stokes equation Consider the drag minimization problem of the two dimensional stationary Stokes equation:

$$\min_{u,p,m} \frac{1}{2} \int_{\Omega} \nabla u \cdot \nabla u \, dxdy + \frac{\alpha}{2} \int_{\partial\Omega_{circle}} m^2 \, ds \tag{14}$$

subject to

$$\begin{cases} -\Delta u + \nabla p = 0 & \text{in } \Omega \\ \text{div } u = 0 & \text{in } \Omega \end{cases} \tag{15}$$

with boundary conditions (BCs)

$$\begin{cases} u = m & \text{on } \partial\Omega_{circle} \\ u = f & \text{on } \partial\Omega_{in}, \end{cases} \begin{cases} u = 0 & \text{on } \partial\Omega_{walls} \\ p = 0 & \text{on } \partial\Omega_{out}, \end{cases} \tag{16}$$

where Ω is a rectangular domain with a circular obstacle inside (Each component is described in detail in Figure 8 in Appendix C.2.). $u = [u_1, u_2]$ is the velocity, p is the pressure, and $m = [m_1, m_2]$ is the control input, which corresponds to the Dirichlet BC on the circle. The inflow BC is given as $f(y) = y(10 - y)/25$. This problem is interpreted as minimizing the drag from the flow by actively controlling the in/outflow on the circle boundary.

In Phase 1, the solution operator is well approximated with the relative error 0.0042 for test data. For the results of Phase 2, the left plot in Figure 3 describes the obtained optimal control through our method. m_1 and m_2 are the x- and y-components of the optimal control, respectively, which are represented as functions of angle θ with respect to the center of the Ω_{circle}. The right plot describes the corresponding velocity fields $[u_1, u_2]$ evaluated by the surrogate model. Figure 4 summarizes the comparison of the inference time for optimal control (Phase 2) by varying the size of mesh when our method and the adjoint method are used. The mesh size in the x-axis means the maximum diameter of meshes used in each method. We remark that the complexity of the adjoint method increases much faster than our method as the mesh size increases. This is because the adjoint method requires heavy computation to obtain the exact gradient value of the objective function with respect to the control parameter.

Inverse design of nonlinear wave equation Given a target function $u_d(x)$ the inverse design problem aims to find the initial conditions that yield a solution $u(T, x) = u_d(x)$. The optimization problem is given by

$$\min_{u,m} \frac{1}{2} \int_{\Omega} \left(u(T, x) - u_d(x)\right)^2 \, dx \tag{17}$$

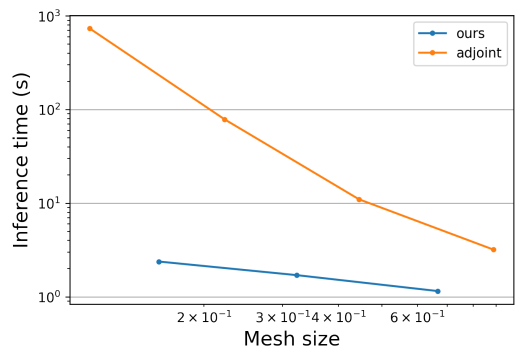

Figure 4: Comparison of the required inference time for the boundary control of the Stokes equation, using our method and the adjoint method. The results are plotted in the log-log scale. This shows that our method achieves better computational complexity than the adjoint method.

subject to the following nonlinear wave equation

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} - a^2 u_{xx} + f(u) = 0 & \text{in } [0, T] \times \Omega \\ u(t, 0) = u(t, L) = 0 & \text{on } [0, T] \times \partial\Omega \\ u(0, x) = 0, u_t(0, x) = m(x) & \text{in } \{t = 0\} \times \Omega \end{cases} \tag{18}$$

where $\Omega = [0, L]$, and nonlinear source term $f(u) = u + u^3$. Here the initial condition $m(x)$ corresponds to control input. We choose $L = 1$, $a = 1/3$, and $T = 5$. If the wave equation has a linear source term, the problem can be easily solved backward in time (time-reversibility). In our problem, however, such an approach fails owing to time-irreversible property caused by nonlinearity $f(u)$. Our method uses the surrogate model for direct mapping from initial to target state.

In Phase 1, the solution operator is well approximated with the relative error 0.0065 for test data. For the results of Phase 2, Table 1 summarizes the results of the optimal control when using our method and the adjoint method. The objective function values of the two methods are comparable, but in terms of the computation time, our model significantly outperforms the adjoint method. This is because the surrogate model can infer fast the solution at $t = T$ for different control inputs, whereas the adjoint method needs to compute the forward and backward computations for each time step to reach the target time. In this regard, our method

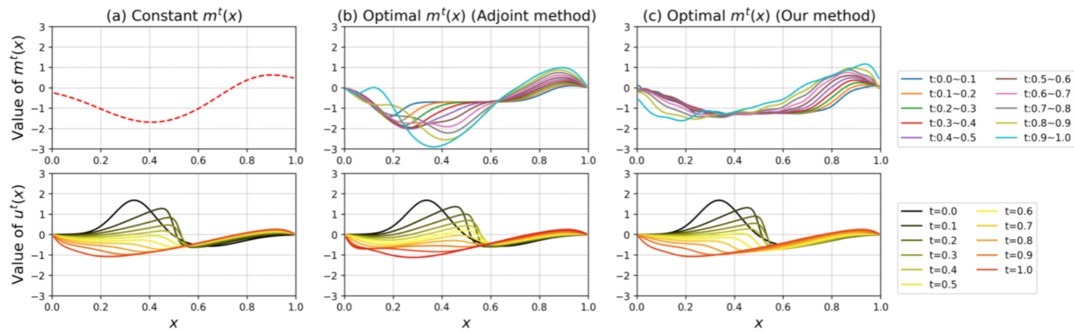

Figure 5: The external force and trajectories with the external force using Burgers' equation. The three columns are (a) a constant external force, (b) an optimal external force using the adjoint method, (c) an optimal external force using our method. The first row shows the time-discretized optimal control inputs m^t $(t = 0, \Delta t, ..., (n-1)\Delta t)$. The second row shows the time evolution of Burgers' equation with the external force in the first row.

can be applied robustly even when the target time becomes longer.

Force control of Burgers' equation We use the extended model explained in Section for the control problem to Burgers' equation. It describes the interaction between the effects of nonlinear convection and diffusion. Burgers' equation leads the shock wave phenomenon when the viscosity parameter has a small value. With the Dirichlet boundary condition, the 1D Burgers' equation with external force $m(t,x)$ reads as

$$\begin{cases} \frac{\partial u}{\partial t} = -u \cdot \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} + m(t,x) & \text{in } [0,T] \times \Omega \\ u(t,x) = 0 & \text{on } [0,T] \times \partial\Omega \\ u(0,x) = u^0(x) & \text{in } \{t=0\} \times \Omega \end{cases} \tag{19}$$

where ν is a viscosity parameter, and $u_0(x)$ is an initial condition. During the experiment, we consider the control problem minimizing

$$\min_{u,m} \frac{1}{2} \int_\Omega |u(T,x) - u_d(x)|^2 \, dx + \frac{\alpha}{2} \int_{\Omega \times [0,T]} |m(t,x)|^2 \, dx dt \tag{20}$$

subject to Burgers' equation (19), given a target $u_d(x)$. In this case, we control an external force m^t. We set $\alpha = 0.01$, $\Omega = [0,1]$, $T = 1$, and $\Delta t = 0.1$. Also, we set the viscosity parameter $\nu = 0.01$ to generate a shock wave.

The solution operator is well trained during Phase 2 with a relative error 0.0020 for test data. Figure 5 and Table 2 show the result of our method compared to the adjoint method for optimal control (Phase 2). A time step size for the adjoint method is chosen as $\Delta t = 0.01$ since the method does not converge when the time step size is set to the coarse grid ($\Delta t = 0.1$) under our setting. In Figure 5, the trajectory in the second and third columns are scattered with less force than the constant external force in the first column. A quantitative comparison of our method to the adjoint method is shown in Table 2. Our framework takes less time compared to the adjoint method, while the objective function values are similar. Our surrogate model can mimic the time evolution of Burgers' equation in the coarse time grid. It makes our method infer faster than the adjoint method for the control optimization problem of time-dependent PDEs.

	Objective	Time (s)
Ours	0.014 ± 0.005	0.210 ± 0.017
Adjoint	0.012 ± 0.005	473.909 ± 43.622

Table 1: Optimal control results of the wave equation, repeated for 50 samples.

	Objective	Time (s)
Ours	0.002 ± 0.002	7.714 ± 1.600
Adjoint	0.004 ± 0.003	12.965 ± 4.306

Table 2: Optimal control results of Burgers' equation, repeated for 100 samples.

Conclusion

We presented a general framework for solving PDE-constrained optimal control problems. We designed the surrogate models for PDE solution operators with a reconstruction structure. It allowed our model to solve the optimal control efficiently by adopting the reconstruction loss as a regularizer. The experimental results demonstrated that the proposed method has a significant gain in time complexity compared to the adjoint method. Also, our framework can be applied flexibly for both data-driven and data-free control problems.

Although our proposed method can achieve many benefits, it cannot completely replace the existing numerical methods. We believe that the two approaches can be complementary. The numerical method has an advantage in terms of accuracy, and our method is computationally efficient. In general, numerical methods slow down exponentially as the number of dimensions increases. We believe that a deep learning method can alleviate this issue.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2017R1E1A1A03070105, NRF-2019R1A5A1028324) and by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government(MSIP) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

References

Alnæs, M.; Blechta, J.; Hake, J.; Johansson, A.; Kehlet, B.; Logg, A.; Richardson, C.; Ring, J.; Rognes, M. E.; and Wells, G. N. 2015. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100).

Berg, J.; and Nyström, K. 2018. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317: 28–41.

Bhatnagar, S.; Afshar, Y.; Pan, S.; Duraisamy, K.; and Kaushik, S. 2019. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2): 525–545.

Borrvall, T.; and Petersson, J. 2003. Topology optimization of fluids in Stokes flow. *International journal for numerical methods in fluids*, 41(1): 77–107.

Bouchouev, I.; and Isakov, V. 1999. Uniqueness, stability and numerical methods for the inverse problem that arises in financial markets. *Inverse problems*, 15(3): R95.

Chen, T.; and Chen, H. 1993. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Transactions on Neural Networks*, 4(6): 910–918.

Christofides, P. D.; and Chow, J. 2002. Nonlinear and robust control of PDE systems: Methods and applications to transport-reaction processes. *Appl. Mech. Rev.*, 55(2): B29–B30.

de Avila Belbute-Peres, F.; Smith, K.; Allen, K.; Tenenbaum, J.; and Kolter, J. Z. 2018. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31: 7178–7189.

Egger, H.; and Engl, H. W. 2005. Tikhonov regularization applied to the inverse problem of option pricing: convergence analysis and rates. *Inverse Problems*, 21(3): 1027.

Fazlyab, M.; Robey, A.; Hassani, H.; Morari, M.; and Pappas, G. J. 2019. Efficient and accurate estimation of lipschitz constants for deep neural networks. *arXiv preprint arXiv:1906.04893*.

Gunzburger, M. D. 2002. *Perspectives in flow control and optimization*. SIAM.

Guo, X.; Li, W.; and Iorio, F. 2016. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 481–490.

Hafner, D.; Lillicrap, T.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, 2555–2565. PMLR.

Han, J.; Jentzen, A.; and Weinan, E. 2018. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510.

Haslinger, J.; and Mäkinen, R. A. 2003. *Introduction to shape optimization: theory, approximation, and computation*. SIAM.

Hinze, M.; Pinnau, R.; Ulbrich, M.; and Ulbrich, S. 2008. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media.

Holl, P.; Koltun, V.; and Thuerey, N. 2020. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*.

Hwang, H. J.; Jang, J. W.; Jo, H.; and Lee, J. Y. 2020. Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach. *Journal of Computational Physics*, 419: 109665.

Jo, H.; Son, H.; Hwang, H. J.; and Kim, E. H. 2020. Deep neural network approach to forward-inverse problems. *Networks & Heterogeneous Media*, 15(2): 247.

Khoo, Y.; Lu, J.; and Ying, L. 2017. Solving parametric PDE problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020a. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020b. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*.

Liang, J.; Lin, M.; and Koltun, V. 2019. Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32.

Lions, J.-L. 1971. *Optimal control of systems governed by partial differential equations*. Die Grundlehren der mathematischen Wissenschaften, Band 170. Springer-Verlag, New York-Berlin. Translated from the French by S. K. Mitter.

Lu, L.; Jin, P.; and Karniadakis, G. E. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Lusch, B.; Kutz, J. N.; and Brunton, S. L. 2018. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1): 1–10.

Ma, W.; Cheng, F.; Xu, Y.; Wen, Q.; and Liu, Y. 2019. Probabilistic representation and inverse design of metamaterials based on a deep generative model with semi-supervised learning strategy. *Advanced Materials*, 31(35): 1901111.

McNamara, A.; Treuille, A.; Popović, Z.; and Stam, J. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3): 449–456.

Mitusch, S. K.; Funke, S. W.; and Dokken, J. S. 2019. dolfin-adjoint 2018.1: automated adjoints for FEniCS and Firedrake. *Journal of Open Source Software*, 4(38): 1292.

Morton, J.; Jameson, A.; Kochenderfer, M. J.; and Witherden, F. 2018. Deep Dynamical Modeling and Control of Unsteady Fluid Flows. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Nabian, M. A.; and Meidani, H. 2018. A deep neural network surrogate for high-dimensional random partial differential equations. *arXiv preprint arXiv:1806.02957*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Pilozzi, L.; Farrelly, F. A.; Marcucci, G.; and Conti, C. 2018. Machine learning inverse problem for topological photonics. *Communications Physics*, 1(1): 1–7.

Pironneau, O. 1974. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(1): 97–110.

Rabault, J.; Kuchta, M.; Jensen, A.; Réglade, U.; and Cerardi, N. 2019. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of fluid mechanics*, 865: 281–302.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707.

Ren, S.; Padilla, W.; and Malof, J. 2020. Benchmarking deep inverse models over time, and the neural-adjoint method. *arXiv preprint arXiv:2009.12919*.

Sirignano, J.; and Spiliopoulos, K. 2018. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375: 1339–1364.

Sokolowski, J.; and Zolésio, J.-P. 1992. Introduction to shape optimization. In *Introduction to Shape Optimization*, 5–12. Springer.

Son, H.; Jang, J. W.; Han, W. J.; and Hwang, H. J. 2021. Sobolev Training for the Neural Network Solutions of PDEs. *arXiv preprint arXiv:2101.08932*.

Tröltzsch, F. 2010. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc.

Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Watter, M.; Springenberg, J.; Boedecker, J.; and Riedmiller, M. 2015. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Weinan, E.; and Yu, B. 2018. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1): 1–12.

Zhu, Y.; and Zabaras, N. 2018. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366: 415–447.

Zhu, Y.; Zabaras, N.; Koutsourelakis, P.-S.; and Perdikaris, P. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394: 56–81.