# Operator Learning for Nonlinear Adaptive Control

**Luke Bhan** LBHAN@UCSD.EDU
*Department of Electrical and Computer Engineering, University of California, San Diego*

**Yuanyuan Shi** YYSHI@ENG.UCSD.EDU
*Department of Electrical and Computer Engineering, University of California, San Diego*

**Miroslav Krstic** KRSTIC@UCSD.EDU
*Department of Mechanical and Aerospace Engineering, University of California, San Diego*

**Editors:** N. Matni, M. Morari, G. J. Pappas

## Abstract

In this work, we propose an operator learning framework for accelerating nonlinear adaptive control. We define three operator mappings in adaptive control-the parameter identifier operator, the controller gain operator, and the control operator. We introduce neural operators for learning both the parameter identification mapping and the gain function mapping to produce the control action at each step. Through the formalization of neural operators, we are able to learn these mappings for a wide set of different system parameter values without retraining. Empirically, we test our controller on two experiments ranging from an aircraft system (a nonlinear ODE) to a first-order hyperbolic PDE system. We demonstrate that the accuracy of both the gain function and parameter approximation can reach the magnitude of $10^{-4}$ with speedups around 98% compared to numerical solvers. Furthermore, we empirically demonstrate that despite error propagation, closed-loop stability guarantees are maintained when substituting neural operator approximations.

**Keywords:** Neural Operators, Adaptive Control

## 1. Introduction

Over the past thirty years, a series of theoretical adaptive controllers have been developed for stabilization of complex dynamical systems. This includes model-based adaptive controllers for nonlinear ODEs Krstic et al. (1995), PDEs Smyshlyaev and Krstić (2018); Anfinsen (2019), and delay systems Zhu and Krstic (2020). Often, these controllers provide well-defined stability and robustness guarantees from a theoretical perspective. However, in practice, many adaptive controllers require solving a series of challenging integro-differential equations. Therefore, the computational cost for complex nonlinear PDE systems can easily become intractable. Thus, a need exists to develop adaptive control techniques that are feasible to compute in real-time without sacrificing the stability and performance guarantees of the original controller.

In this work, we design an operator learning framework for accelerating adaptive control that arises in both nonlinear ODE and PDE systems. To guarantee the stabilization of the closed-loop system, we generate supervised training data from adaptive controllers designed by the *backstepping* method. To accelerate the computational efficiency, we use *neural operators* for model learning. These operator learning techniques have shown superior performance in learning mappings between infinite-dimensional spaces of functions Lu et al. (2021), Li et al. (2020). Specially, we learn the functional mapping from the system measurements, control inputs, and initial parameter estimates to the parameter estimates and gain function values. Pairing an adaptive controller of nonlinear systems, with neural operator approximations of the parameter identifier and controller gains enables the user to capitalize on the mathematical rigor of the controller and the computational performance of operator learning. This allows us to achieve the best of both worlds. Our contributions in this paper are summarized as follows.

**Contributions.** Theoretically, we formulate the operator learning problem for nonlinear adaptive control. We define three operator mappings in adaptive control, the parameter identifier operator $\mathbb{I}$, the controller gain operator $\mathbb{G}$, and the control operator $\mathbb{C}$ in Section 3.1. Empirically, we demonstrate our controller on two experiments including a simulation of a aircraft wing-rock model (a nonlinear ODE system) and a challenging first-order hyperbolic PDE system. In these experiments, we show accuracy on the magnitude of $10^{-4}$ and a speedup of around $98\%$ compared to numerical solvers. Furthermore, we showcase that stability guarantees of the closed-loop system are preserved when inserting the neural operator approximated parameter identifier and gain function.

The proposed operator learning paradigm for parameter identification is intrinsically different from existing ML methods for system ID Ljung et al. (2020); Pillonetto et al. (2014); Andersson et al. (2019); Gedon et al. (2021); Dalla Libera and Pillonetto (2022); Raissi et al. (2019). Existing methods perform system ID with training instances generated from one single system, and require re-training from scratch when the system parameters change. We conduct training, instead, with data generated from ground-truth systems with a range of different system parameters and initial conditions. Thus, once the neural operator is learned, we apply it in real-time without re-training. Whenever the system parameters change significantly (due to a drift in the environment, aging, and wear), the neural operator identifier will "automatically" update the parameter estimates, empowered by the adaptive control scheme.

## 2. Background and Related Works

### 2.1. A Brief Introduction to Adaptive Control

In general, an adaptive controller consists of two main components: a parameter update law and a control law. Parameter identification estimates the unknown parameters of a system based on the measured output. This estimation is then fed into the controller, which injects input signals into the system (typically at a boundary point) in order to achieve some desired trajectory. As an illustrative example, consider a first-order hyperbolic PDE system with output recirculation,

$$u_t(x,t) = u_x(x,t) + \theta(x)u(0,t)\,, \tag{1a}$$

$$Y(t) = u(0,t)\,, \quad \text{(measured output)} \tag{1b}$$

$$u(1,t) = U(t)\,, \quad \text{(control input)} \tag{1c}$$

where $\theta(x)$ are unknown, continuous functions, and for all $0 \le x \le 1$, $\theta(x)$ is bounded - $|\theta(x)| \le M_\theta$ for constant $M_\theta > 0$. In this example, lets define the goal to regulate $u(x,t)$ to 0 for all $x \in [0,1]$ using the measurement $Y(t) = u(0,t)$ and the boundary control $U(t) = u(1,t)$. The following theorem presents a paired adaptive controller and parameter update law from Bernard and Krstic (2014) that guarantees stabilization of the closed-loop system.

**Theorem 1 (Theorem 5, Bernard and Krstic (2014))** *Consider the plant* (1) *with controller,*

$$U(t) = \int_{t-1}^{t} \hat{\kappa}(t-\tau, t)U(\tau)d\tau + \int_{t-1}^{t} \left[\int_{t-\tau}^{1} \hat{\kappa}(\mu, t)\hat{\theta}(1-\mu+t-\tau, t)d\mu\right] Y(\tau)d\tau \tag{2}$$

*the parameter update law for $\hat{\theta}$ is generated by,*

$$\hat{\theta}_t(x,t) = \gamma(x)\frac{Y(t-x)\left[Y(t) - U(t-1) - \int_{t-1}^{t} \hat{\theta}(t-\tau,t)Y(\tau)d\tau\right]}{1+\int_{t-1}^{t} Y^2(\tau)d\tau} \tag{3}$$

*where $\gamma(x)$ is the estimation gain. $\hat{\kappa}$, the controller gain function, is obtained from $\hat{\theta}$ by real-time solution of the following equation,*

$$\hat{\kappa}(x,t) = -\hat{\theta}(x,t) + \int_{0}^{x} \hat{\kappa}(y,t)\hat{\theta}(x-y,t)dy \tag{4}$$

*For any initial condition, $\hat{\theta}(\cdot, 0) \in C^1(0,1)$, $Y(0), U(0)$, the solution $(u, \hat{\theta})$ and the control input $U$ are bounded for all $x \in [0,1]$, $t \geq 0$ and $\lim_{t\to\infty} u(x,t) = 0, \forall x \in [0,1]\,, \lim_{t\to\infty} U(t) = 0$.*

Notice that the parameter estimator (3) contains dynamics because it involves delayed input $U(t)$ and output $Y(t)$. More importantly, the mapping from the measurement $Y(t)$ to the parameter estimator in (3) are highly nonlinear and therefore require significant computational time to calculate. The task of operator learning will be to capture these nonlinear mappings, thus accelerating the computation of adaptive control for real-time applications.

## 2.2. Neural Operators

Recently, machine learning (ML) methods have shown promise in solving complex algebraic and differential equations. Earlier works on data-driven solvers borrow off-the-shelf ML models designed for computer vision and language processing. Examples include multi-layer perceptions (MLPs) Levine and Stuart (2021), convolution neural networks (CNNs) Bhatnagar et al. (2019), and graph neural networks Alet et al. (2019). Another line of work, called Physics-Informed Neural Networks (PINNs) (see Raissi et al. (2019) and references within), add physics constraint loss to neural networks. PINNs take advantage of the auto-differentiation of neural networks and can serve as generic solvers for PDEs. However, PINNs need to be re-trained for new boundary and initial conditions, thus not fitting our goal of accelerating adaptive control in real-time.

In Lu et al. (2019); Li et al. (2020), neural operators are designed for solving partial differential equations and dynamical systems. Compared to other ML methods, neural operators provide two unique advantages. Firstly, from a theoretical perspective, neural operators formalize the learning problem as a mapping of function spaces. This contrasts standard neural networks that target learning the map between finite-dimensional vector spaces. Neural operators enable us to learn mappings for an entire set of system parameters instead of requiring retraining when system parameters change. Secondly, from an empirical perspective, Lu et al. (2021), Shi et al. (2022) demonstrated that neural operators achieve superior accuracy in emulating challenging functions compared to standard deep learning approaches. We provide a short review of two popular neural operator architectures, DeepONet Lu et al. (2019) and Fourier Neural Operators Li et al. (2020).

**Definition 2** *A DeepONet operator is defined as two neural networks in the form,*

$$\mathcal{H}_{\mathbb{N}}(\mathbf{u}_m) := \sum_{k=1}^{p} \underbrace{g^{\mathcal{N}}(\mathbf{u}_m; \Theta^{(k)})}_{\textit{branch}} \underbrace{f^{\mathcal{N}}(y; \theta^{(k)})}_{\textit{trunk}} \tag{5}$$

*where there exists positive integers $p$ and $m$, such that $k = 1, ..., p$, neural networks defined as $f^{\mathcal{N}}(\cdot; \theta^{(k)})$, $g^{\mathcal{N}}(\cdot; \Theta^{(k)})$ and measurements $\mathbf{u}_m = (u(x_1), u(x_2), ..., u(x_m))^\top$ for $x_j \in K_1$.*

From a high-level perspective, one can think of a DeepONet as two concatenated neural networks where the branch network takes in the system measurements and the trunk network takes in the system's domain. In practice, one can choose these neural networks in any form such as MLPs, CNNs, or even recurrent neural networks (RNNs) depending on the type of data stream.

**Definition 3** *A Fourier neural operator (FNO) is defined as the composition of linear integral operator $\mathcal{K}$ with pointwise non-linear activation function $\sigma$ in the following form,*

$$\mathcal{H}_{\mathbb{N}}(\mathbf{u}_m) := \mathcal{Q} \circ \sigma(W_L + \mathcal{K}_L) \circ \cdots \sigma(W_1 + \mathcal{K}_1) \circ \mathcal{P} \tag{6}$$

*where $\mathcal{P}$, $\mathcal{Q}$ and $W_l \in \mathbb{R}^{d_{l+1} \times d_l}$ are matrices, and the integral operator $\mathcal{K}$ is implemented with a Fourier transform,*

$$\mathcal{F}_l := \mathcal{F}^{-1}\Big(K_l \cdot (\mathcal{F}\mathbf{u}_m^{l-1})\Big)(x) \qquad \forall x \in D. \tag{7}$$

*where $\mathcal{F}$ denotes the Fourier transform and $\mathcal{F}^{-1}$ denotes the inverse Fourier transform, and $K_l$ denotes the linear transformation matrix applied to function in the Fourier domain.*

## 3. Operator Learning for Adaptive Control

### 3.1. Operator Learning Formulation

Our goal is to accurately capture the nonlinear mappings in adaptive control. To do this, we construct simulations with a large set of functional initial conditions $u(x, 0)$ and system functional parameters $\theta(x)$. Both will be unknown in the implementation of the neural observers. In this work, we will formally introduce three operators in the adaptive control framework. To the authors' knowledge, this is the first time these operators are formulated and learned across different parameter functions. We define the three core operators as:

**Parameter identifier operator $\mathbb{I}$:**

$$\mathbb{I} : \Big(\hat{\theta}_0(x), x \in [0, 1]; Y(\tau), \tau \in [t-1, t]; U(\tau), \tau \in [t-1, t]\Big) \mapsto \Big(\hat{\theta}(x, t), x \in [0, 1]\Big) \tag{8}$$

**Gain operator $\mathbb{G}$:**

$$\mathbb{G} : \Big(\hat{\theta}(x, t), x \in [0, 1]\Big) \mapsto (\hat{\kappa}(x, t), x \in [0, 1]) \tag{9}$$

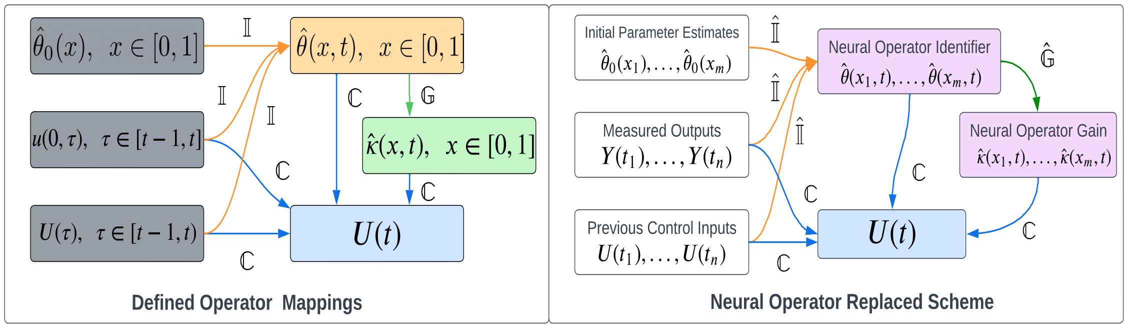
$\hat\theta_0(x), \ x \in [0,1]$
$\hat\theta(x,t), \ x \in [0,1]$
$u(0,\tau), \ \tau \in [t-1,t]$
$\hat\kappa(x,t), \ x \in [0,1]$
$U(\tau), \ \tau \in [t-1,t)$
$U(t)$
$\mathbb{I}$
$\mathbb{C}$
$\mathbb{G}$
Defined Operator Mappings
Initial Parameter Estimates
$\hat\theta_0(x_1),\dots,\hat\theta_0(x_m)$
Neural Operator Identifier
$\hat\theta(x_1,t),\dots,\hat\theta(x_m,t)$
Measured Outputs
$Y(t_1),\dots,Y(t_n)$
Neural Operator Gain
$\hat\kappa(x_1,t),\dots,\hat\kappa(x_m,t)$
Previous Control Inputs
$U(t_1),\dots,U(t_n)$
$\hat{\mathbb{I}}$
$\hat{\mathbb{G}}$
Neural Operator Replaced Scheme

Figure 1: Diagram of the proposed operator learning for adaptive control framework.

**Controller operator $\mathbb{C}$:**

$$\mathbb{C} : \Big(U(\tau), \tau \in [t-1,t); Y(\tau), \tau \in [t-1,t]; \hat\theta(x,t), x \in [0,1]; \hat\kappa(x,t), x \in [0,1]\Big) \mapsto U(t) \tag{10}$$

We focus on learning the first two operators: the parameter identifier operator $\mathbb{I}$, and the control gain operator $\mathbb{G}$, while directly solving for the controller input $U(t)$. To learn the neural operator approximation of the parameter estimation and gain mappings, we follow the standard supervised learning setup.

*1) Operator Learning for Parameter Identifier* $\mathbb{I}$: To learn a neural operator for the parameter identifier operator, we need first to generate a training dataset containing input-output pairs produced by solving the parameter update law, e.g., (3) for the hyperbolic PDE system. Consider a time interval $[t-1,t]$ with measurement interval $\Delta t$, and contains $n$ intervals: $t_1 = t-1, t_2 = t-1+\Delta t, ..., t_n = t$. For the spatial domain $[0,1]$ with discretization interval $\Delta x$, and contains $m$ intervals: $x_1 = 0, x_2 = \Delta x, ..., x_m = 1$. Denote $\Delta t, \Delta x$ to be the the temporal and spatial intervals between two measurements, and $n, m$ to denote the total number of temporal and spatial measurements. For the neural operator design, we learn a mapping $\hat{\mathbb{I}}$ that maps the initial parameter estimates $[\hat\theta_0(x_1), \hat\theta_0(x_2), ..., \hat\theta_0(x_m)]$, the boundary measurements $[Y(t_1), Y(t_2), ..., Y(t_n)]$ and the control inputs $[U(t_1), U(t_2), ..., U(t_n)]$, to the estimates $[\hat\theta(x_1,t), ..., \hat\theta(x_m,t)]$. Then we train the neural operator model $\hat{\mathbb{I}}$ (parameterized as DeepONet or FNO) to minimize the difference between the parameter estimates produced by neural operators $[\hat\theta_{\hat{\mathbb{I}}}(x_1,t), ..., \hat\theta_{\hat{\mathbb{I}}}(x_m,t)]$ and $[\hat\theta(x_1,t), ..., \hat\theta(x_m,t)]$. In all cases, we minimize the $L_2$ loss between the estimates produced by the neural operators and that from the original parameter estimator.

*2) Operator Learning for Gain Function* $\mathbb{G}$: To learn the gain function mapping, we generate the training dataset containing the input-output pairs produced by solving the control gain equations, e.g., (4) for the hyperbolic PDE system. For the neural operator design, we learn a mapping $\hat{\mathbb{G}}$ that maps the parameter estimates $[\hat\theta(x_1,t), ..., \hat\theta(x_m,t)]$ to the control gains $[\hat\kappa(x_1,t), ..., \hat\kappa(x_m,t)]$. Similarly, the learning goal be to minimize the difference between the neural operator approximated gain function $[\hat\kappa_{\hat G}(x_1,t), ..., \hat\kappa_{\hat G}(x_m,t)]$ and $[\hat\kappa(x_1,t), ..., \hat\kappa(x_m,t)]$.

An overview of the operator learning for the adaptive control framework and the input and output information flow is shown in Figure 3.1. The neural operator approximations of the parameter estimates and gain functions will be fed into the controller operator (10), together with the boundary measurements and previous control inputs, to produce the final control action $U(t)$. Once such solution operators are learned, they can directly evaluate any new input queries with different system parameter $\theta(x)$ or initial conditions $u(\cdot,0)$ without solving the differential and integral equations.

As a result, the data-driven solver can be orders of magnitude faster compared to conventional numerical solvers, making it more amenable to real-time applications.

**Remark 4 (Alternative operator learning formulation)** There are alternative ways of formulating operator mappings associated with adaptive control, besides the formulations in (8)-(10). For example, for learning the controller gain operator $\mathbb{G}$, rather than using the intermediate prediction $\hat{\theta}(x,t)$, and then mapping to $\hat{\kappa}(x,t)$, one can learn a direct composition operator as follows,

$$\mathbb{G}(\mathbb{I}) : \left(\hat{\theta}_0(x), x \in [0,1]; Y(\tau), \tau \in [t-1,t]; U(\tau), \tau \in [t-1,t]\right) \mapsto (\hat{\kappa}(x,t), x \in [0,1])$$

Similarly for the controller operator $\mathbb{C}$, instead of using the intermediate predictions $\hat{\theta}(x,t)$ and $\hat{\kappa}(x,t)$, one can learn an end-to-end composition control operator,

$$\mathbb{C}(\mathbb{I},\mathbb{G}) : \left(\hat{\theta}_0(x), x \in [0,1]; Y(\tau), \tau \in [t-1,t]; U(\tau), \tau \in [t-1,t]\right) \mapsto U(t)$$

Our framework, learning the direct operator mappings (8)-(10), features better explainability and flexibility as the intermediate parameter estimation $\hat{\theta}_{\hat{\mathbb{I}}}(x,t)$ and gain function $\hat{\kappa}_{\hat{\mathbb{G}}}(x,t)$ can be used for monitoring the model performance. The alternative framework, in learning the composed operators feature simplicity in directly learning the composition of mappings and can potentially provide further acceleration. An interesting future work direction would be comparing these two operator learning frameworks in terms of accuracy, speedup, and closed-loop performance.

## 3.2. Approximator Error of Neural Operators

In this section, we highlight that the neural operator has a bounded approximation error for a variety of operators. We present this as an introductory step for formally retaining the rigorous guarantees of the original parameter estimator and controller with a neural operator approximation. In Lu et al. (2021), it has been shown that the DeepONet satisfies the universal approximation of continuous operators (below) as long as the branch and trunk neural networks satisfy the universal approximation theorem of continuous functions on compact sets.

**Definition 5 (Universal approximation of continuous operators)** *Let $K_1 \subset \mathbb{R}^{d_1}, K_2 \subset \mathbb{R}^d$ be two compact sets. Let $V$ be a compact set in $C(K_1)$. Assume that $\mathcal{H} : V \mapsto C(K_2)$ is a continuous operator. If for any $\epsilon > 0$, there exists a neural operator $\hat{\mathcal{H}}$ belonging to the class of neural operators such that the following holds:*

$$|\mathcal{H}(u)(y) - \hat{\mathcal{H}}(\mathbf{u}_m)| < \epsilon \qquad \forall u \in V, y \in K_2 \tag{11}$$

*where $\mathbf{u}_m = (u(x_1), u(x_2), ..., u(x_m))^\top$ for $x_j \in K_1$, we say that this class of neural operators has universal approximation ability.*

A similar result has been shown for FNO in Kovachki et al. (2021), proving the universal approximation property of FNO for continuous operators. Besides the universal approximation ability of neural operators, one can also bound the size of the neural operators to ensure a desired accuracy of the approximation. See Deng et al. (2021). From a theoretical standpoint, this bound requires billions of parameters. However, in practice as evidenced by the numerical results, one can see stable parameter and gain approximations are achieved with a much smaller sized neural operator. This result is the basis for the theoretical foundation of neural operators for adaptive control and the key reason why using infinite dimensions is worthwhile. Since we are able to bound the operator error, we can then treat the NN as a disturbance effect on the adaptive controller and analyze the resulting stability properties of the system. We anticipate this theoretical formulation will be a important direction for our future work.

## 4. Experiments

In this section, we present and analyze the performance of the proposed operator learning for adaptive control framework experimentally. We demonstrate the performance comparisons in two benchmark control tasks: (i) a wing-rock model of aircraft (for demonstrating the parameter identifier operator learning combined with the controller mapping) and (ii) first-order hyperbolic PDE in Section 2.1 for demonstrating all parts working together. The code for our experiments is available on Github. Additionally all experimental details, hyperparameters, model architectures, and a further experiment with a simple linear ODE system is available in the supplemental.

### 4.1. General Experimental Details

For all experiments, we follow a similar formulation for generating a dataset. We first solve the system with the analytical mappings for 1000 instances while randomizing the system parameters and initial conditions that changes the systems behavior. We chose a spatial and time step size of $\Delta x = \Delta t = 0.01$ unless otherwise noted as in the first-order hyperbolic PDE example. We then split the data into 900/100 training testing sets and train all learning models on the same sets. As baselines, we consider two common recurrent neural network architectures - the GRU and LSTM. We chose these due to the temporal nature of our mappings. Additionally, we include two neural operator structures, DeepONet and FNO, as there are cases where they perform differently from accuracy and computation speed perspective Lu et al. (2022). We provide network architectures and example datasets in the supplemental. Lastly, all experiments are run on an Nvidia RTX 3090.

### 4.2. Wing-Rock Model of an Aircraft

Now, we demonstrate our approach in an adaptive control scheme for a real-world wing rock model. We consider the model from Monahemi and Krstić (1996) defined as the following:

$$\ddot{\phi} = \theta_1 + \theta_2\phi + \theta_3\dot{\phi} + \theta_4|\phi|\dot{\phi} + \theta_5|\dot{\phi}|\dot{\phi} \tag{12}$$

where $\phi$ is the angle, $\dot{\phi}$ is the angular velocity and $\theta_n$ is a constant parameter of the system. We can rewrite the above equations in a parametric form and introduce the control variable $u \in \mathbb{R}$.

$$\dot{\phi} = p \, , \dot{p} = \varphi^{\mathrm{T}}(\phi, p)\theta + u \tag{13}$$

with $\theta = [\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5]^{\mathrm{T}}$ and $\varphi(\phi, p) = [1,\ \phi,\ p,\ |\phi|p,\ |p|p]^{\mathrm{T}}$. After rewriting Eqn. (12) in a parametric form, we can then use the least-squares estimator from Krstic (2009).

$$\hat{\theta} = \alpha + \Gamma \int_0^p \varphi(\phi, \sigma) d\sigma \, , \dot{\Gamma} = -\Gamma\varphi\varphi^T\Gamma \, , \dot{\alpha} = -\Gamma\varphi\varphi^T\alpha - \Gamma p \int_0^p \frac{\partial\varphi(\phi, \sigma)}{\partial\phi} d\sigma - \Gamma\varphi u \tag{14}$$

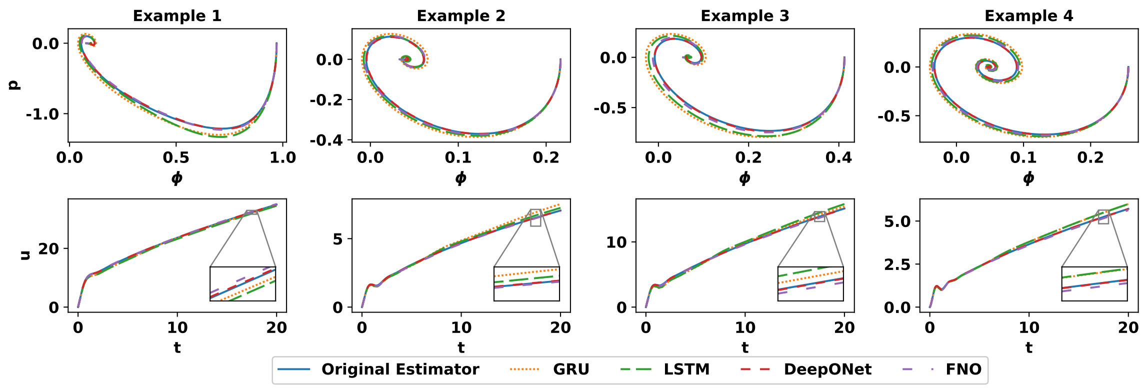

Figure 2: System control results with the replacement of the original parameter estimator with different types of ML estimators. The top row demonstrates system solutions (x-axis: $\phi$, y-axis: $p = \dot{\phi}$) with target (0, 0) and the bottom showcases the control input $u(t)$.

| Model | Parameter Size | Training Time (mins) | Average Relative Error Training Data | Average Relative Error Testing Data | Average Calculation Time CPU (sec) | Average Calculation Time GPU(sec) |
|---|---|---|---|---|---|---|
| GRU | 2194705 | 87.33 | 7.20E-4 | 3.25E-4 | 1.85E-1 | 1.36E-2 |
| LSTM | 2924855 | 98.49 | 6.58E-4 | 2.94E-4 | 2.39E-1 | 1.06E-2 |
| DeepONet | **298848** | 36.78 | **7.61E-5** | **5.98E-5** | 6.0E-2 | 5.5E-3 |
| FNO | 311669 | **9.41** | 2.33E-4 | 1.06E-4 | **4.3E-3** | **3.6E-3** |

Table 1: Summary results for the wing-rock system. Green indicates the best value.

with the control law of the form:

$$u = \dot{y_r} + \left(K - \frac{\lambda}{2} e_2^T P\right) \tilde{x} - \varphi^T \hat{\theta} \tag{15}$$

where $x_r(t) = [y_r(t), \dot{y}_r(t)]^T$ is the vector of the reference trajectory function $y_r(t)$ and its derivative, K is chosen to satisfy a set of Hurwitz criteria (See Krstic (2009), Section 2.2), and e is defined as $[0, 1]$. We solve the identifier operator in Eq (8) as we replace the estimation $\hat{\theta}$ in the control scheme by a neural operator. To build a dataset, we generate 1000 solutions, in which every element of $\theta$ is distributed according to $\theta \sim [U(0,1), U(-24,-28), U(0.5,1), U(-2,-4), U(-0.01,0)]$, where $U(a,b)$ denotes the uniform distribution between interval $(a,b)$. Additionally, we generate $\phi_0 = U(0,1)$, $\dot{\phi}_0 = 0$ and $\theta_0 = \theta * U(0,2)$. We then split the dataset into 900 training solutions and 100 testing solutions to learn the following mapping:

$$[\hat{\theta}_0, \phi(0), \dot{\phi}(0), \phi(\Delta t), \dot{\phi}(\Delta t), \phi(2\Delta t), \dot{\phi}(2\Delta t), ..., \phi(t=1)] \mapsto [\hat{\theta}(0), \hat{\theta}(\Delta t), \hat{\theta}(2\Delta t), ..., \hat{\theta}(t=1)].$$

We showcase results for training and speedups in Table 1. We can see that the operators perform the best from an accuracy perspective and the FNO performs the best from a speed perspective. As seen in Table 1, all the approaches are able to learn the estimator to quite a high degree of accuracy. Furthermore, we can see the error propagation in 4 examples of the control in Figure 2. In all cases, the aircraft is able to stabilize to the reference speed and angle. We emphasize this as we are able to

| **Model** | **Parameter Size** | **Training Time (mins)** | **Average Relative Error Testing Data For Solution $u$** | **Average Relative Error Testing Data For Control $U(t)$** | **Average Calculation Time CPU (sec)** | **Average Calculation Time GPU(sec)** |
|---|---|---|---|---|---|---|
| GRU | 1608051 | 12.23 | 8.96E-4 | 3.20E-3 | 0.89 | **0.057** |
| LSTM | 2135551 | 12.19 | 8.12E-4 | 3.19E-3 | 1.23 | 0.059 |
| DeepONet | 671152 | 18.83 | 9.9E-4 | 3.79E-3 | 0.875 | 0.065 |
| FNO | **534931** | **11.55** | **4.46E-4** | **1.52E-3** | **0.242** | 0.237 |

Table 2: Summary results for the first-order hyperbolic PDE system. Green indicates the best value. The original scheme takes a total of 3.23 seconds (CPU time) of calculation for the parameter estimator and controller gain function combined.

approximate the estimator mapping in not just an open loop, but closed feedback control with high success. From a speed perspective, it is worth noting that the FNO does not have any internal state across time as opposed to the RNNs. Therefore, FNO is able to perform quite well on both the CPU and GPU. The original control scheme takes 0.038 CPU (sec) for the parameter estimation, and the FNO-accelerated parameter estimator provides a speedup of 90%. Additionally, we can see that the other approaches do not perform as well as the original estimator unless they are run on the GPU. However, this is the most common implementation for neural network applications and therefore it is reasonable to assume that in practice, these control schemes are computed on the GPU.

## 4.3. General first-order hyperbolic PDE

Secondly, we utilize the control form for a hyperbolic PDE first introduced in Section 2.1 with $\gamma(x) = 1$. We parameterize $\theta(x)$ as the Chebyshev polynomial $\theta(x) = 3\cos(\delta \arccos(x))$. We again solve 1000 instances of this problem from $t \in [0, 15]$ and $x \in [0, 1]$ where we vary $\delta \sim U(1, 12)$, $u(x, 0) = U(50, 100)$. We keep $\theta_0$ constant at 0 and emphasize that $\delta$ creates a wide array of different PDE shapes making this problem extraordinary challenging compared to learning something easier like $\theta(x) = \sin(x)$ (Example solution shapes in the supplemental). For the dataset, we subsample at a rate of $\Delta t = 0.05$ and $\Delta x = 0.02$ for our dataset but solve the control loop at step sizes of $\Delta t = 0.01$ and $\Delta x = 0.01$.

We replace both the gain function $\mathbb{G}$ and parameter estimator $\mathbb{I}$ with neural operators. We showcase both the final control errors and the solution errors for the testing set in Table 2. We see that the FNO approach performs the best in terms of accuracy. In addition, by including the learning portion of the gain function, we obtain much faster speedups across all methods. Furthermore, Fig 3 shows the parameter estimation and gain function learning results in one text example. The accuracy across all methods is very strong as there is no distinguishable difference in the first two rows of Figure 3. We can see that the control errors are much larger than the previous wing-rock systems. However, in all learning accelerated methods, we can see that the solution $u(x, t)$ stabilizes. Again, we emphasize that not only can we approximate both the mappings $\mathbb{I}$ and $\mathbb{G}$ in an open loop, but the error propagation is small enough for a closed feedback system to stabilize.

## 5. Conclusion and Future Works

In this work, we have presented a new adaptive control scheme augmented with neural operators. Effectively, we demonstrate not only the open loop approximation results of learning of two chal-

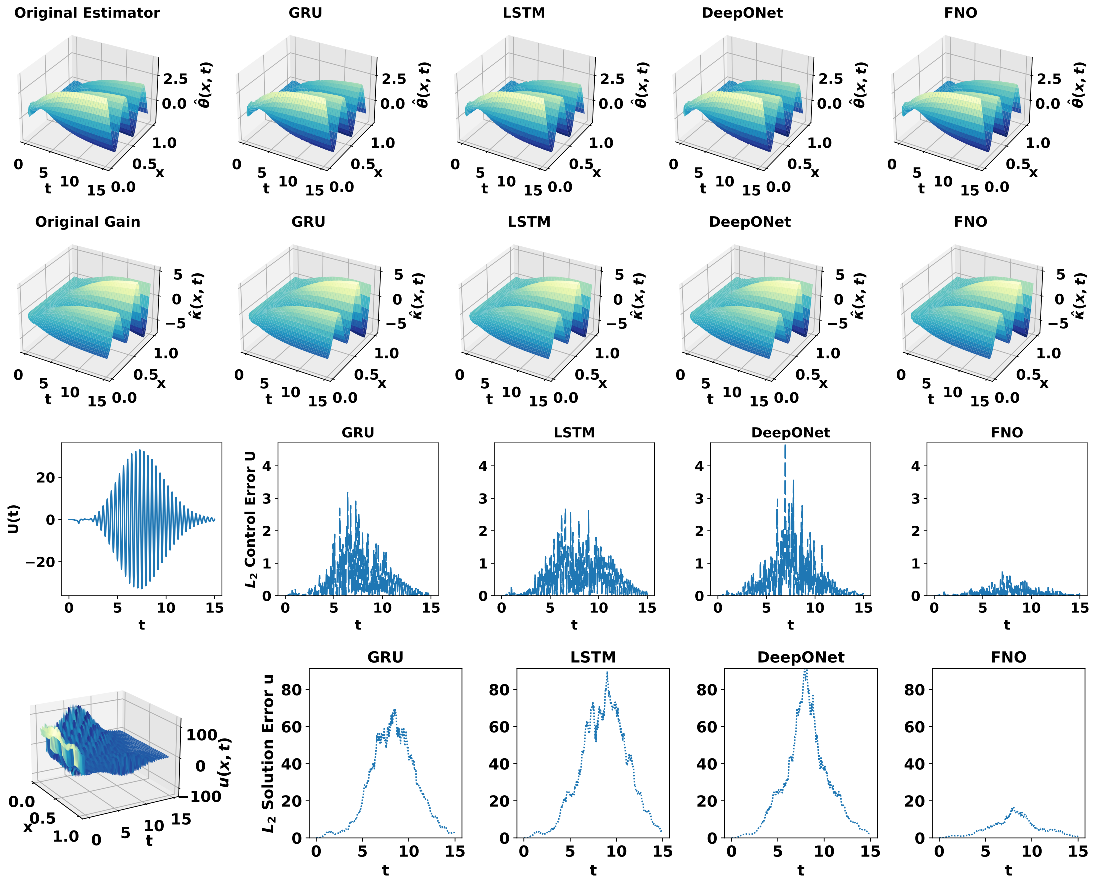

Figure 3: One example of the hyperbolic PDE system. The top two rows display the learned estimator and gain mappings respectively. The third row displays the control function (left) and the residual $\mathcal{L}_2$ errors for each method. The bottom row displays the solution (left) and the solution $\mathcal{L}_2$ error (summed over x) for each method in the closed feedback loop.

lenging parameter estimation and controller gain operators, but stabilize the nonlinear systems in a feedback control loop. We demonstrate a high accuracy around a magnitude of $10^{-4}$ on both nonlinear systems while demonstrating speedups up to 98%. Additionally, by formalizing the problem as an operator learning problem, we are able to avoid retraining when the system parameters change significantly. This work serves as an exciting direction to developing learning-augmented control and there are many future directions to be explored. For example, one important direction is to explore the approximation error of parameter and gain function operators from a theoretical perspective in the closed-loop control systems. For example, Lu et al. (2019); Kovachki et al. (2021) have developed theoretical operator approximation guarantees for both FNO and DeepONet that can be used to analyze the stability of both ODE and PDE systems under the neural operator approximated gain and estimation laws. Additionally, improving the operator architectures to take advantage of certain control forms such as the convolution-like gain function in the hyperbolic PDE is certainly worthwhile. Lastly, it is worth exploring whether learning the entire control function instead of the individual components can provide similar accuracy with improved speedups.

## Acknowledgments

We thank Professor Paulina Bernard for her code on the hyperbolic PDE system, and Jie Feng, Zongyi Li for interesting conversations surrounding hyperparameters and learning techniques.

## References

Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pages 212–222. PMLR, 2019.

Carl Andersson, Antônio H. Ribeiro, Koen Tiels, Niklas Wahlström, and Thomas B. Schön. Deep convolutional networks in system identification. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3670–3676, 2019. doi: 10.1109/CDC40024.2019.9030219.

Henrik Anfinsen. *Adaptive control of hyperbolic PDEs*. Springer, Cham, Switzerland, 2019. ISBN 978-3-030-05879-1.

Pauline Bernard and Miroslav Krstic. Adaptive output-feedback stabilization of non-local hyperbolic pdes. *Automatica*, 50(10):2692–2699, 2014. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2014.09.001. URL `https://www.sciencedirect.com/science/article/pii/S0005109814003550`.

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

Alberto Dalla Libera and Gianluigi Pillonetto. Deep prediction networks. *Neurocomputing*, 469:321–329, 2022. ISSN 0925-2312.

Beichuan Deng, Yeonjong Shin, Lu Lu, Zhongqiang Zhang, and George Em Karniadakis. Convergence rate of deeponets for learning operators arising from advection-diffusion equations, 2021. URL `https://arxiv.org/abs/2102.10621`.

Daniel Gedon, Niklas Wahlström, Thomas B. Schön, and Lennart Ljung. Deep state space models for nonlinear system identification. *IFAC-PapersOnLine*, 54(7):481–486, 2021. ISSN 2405-8963. 19th IFAC Symposium on System Identification SYSID 2021.

Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22:Art–No, 2021.

M. Krstic, I. Kanellakopoulos, and P. V. Kokotovic. *Nonlinear and Adaptive Control Design*. Wiley, New York, NY, 1995.

Miroslav Krstic. On using least-squares updates without regressor filtering in identification and adaptive control of nonlinear systems. *Automatica*, 45(3):731–735, 2009. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2008.09.024. URL `https://www.sciencedirect.com/science/article/pii/S0005109808004949`.

Matthew E Levine and Andrew M Stuart. A framework for machine learning of model error in dynamical systems. *arXiv:2107.06658*, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2020. URL `https://arxiv.org/abs/2010.08895`.

Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B. Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020. ISSN 2405-8963. 21st IFAC World Congress.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. doi: 10.1038/s42256-021-00302-5. URL `https://doi.org/10.1038%2Fs42256-021-00302-5`.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, apr 2022. doi: 10.1016/j.cma.2022.114778. URL `https://doi.org/10.1016%2Fj.cma.2022.114778`.

Mogen M. Monahemi and Miroslav Krstić. Control of wing rock motion using adaptive feedback linearization. *Journal of Guidance Control and Dynamics*, 19:905–912, 1996.

Gianluigi Pillonetto, Francesco Dinuzzo, Tianshi Chen, Giuseppe De Nicolao, and Lennart Ljung. Kernel methods in system identification, machine learning and function estimation: A survey. *Automatica*, 50(3):657–682, 2014. ISSN 0005-1098.

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

Yuanyuan Shi, Zongyi Li, Huan Yu, Drew Steeves, Anima Anandkumar, and Miroslav Krstic. Machine learning accelerated pde backstepping observers. *61st IEEE Conference on Decision and Control (CDC), 2022*, 2022.

A. Smyshlyaev and M. Krstić. *Boundary Control of PDEs: A Book on Backstepping Designs*. SIAM, 2018.

Y. Zhu and M. Krstic. *Delay-Adaptive Linear Control*. Princeton University Press, 2020.